Computer Communications and Networks

Monowar H. Bhuyan
Dhruba K. Bhattacharyya
Jugal K. Kalita

Network Traffic Anomaly Detection and Prevention

Concepts, Techniques, and Tools

Springer

Computer Communications and Networks

Series editor
A.J. Sammes
Centre for Forensic Computing
Cranfield University, Shrivenham Campus
Swindon, UK

The **Computer Communications and Networks** series is a range of textbooks, monographs and handbooks. It sets out to provide students, researchers, and non-specialists alike with a sure grounding in current knowledge, together with comprehensible access to the latest developments in computer communications and networking.

Emphasis is placed on clear and explanatory styles that support a tutorial approach, so that even the most complex of topics is presented in a lucid and intelligible manner.

More information about this series at http://www.springer.com/series/4198

Monowar H. Bhuyan • Dhruba K. Bhattacharyya
Jugal K. Kalita

Network Traffic Anomaly Detection and Prevention

Concepts, Techniques, and Tools

Springer

Monowar H. Bhuyan
Kaziranga University
Jorhat, India

Dhruba K. Bhattacharyya
Tezpur University
Napaam, India

Jugal K. Kalita
University of Colorado
Colorado Springs
CO, USA

ISSN 1617-7975 ISSN 2197-8433 (electronic)
Computer Communications and Networks
ISBN 978-3-319-65186-6 ISBN 978-3-319-65188-0 (eBook)
DOI 10.1007/978-3-319-65188-0

Library of Congress Control Number: 2017949881

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

This book is dedicated to my beloved son Zeeshan and wife Rehena, without whose love, encouragement, support, and inspiration I would never have made it this far. I also wish to dedicate this book to my parents for their continuous love and support all the way since the beginning.

Monowar H. Bhuyan

Dedicated to my deceased father T. N. Bhattacharyya.

Dhruba K. Bhattacharyya

Dedicated to my deceased parents Benudhar Kalita (Deuta) and Nirala Kalita (Maa) and my daughter Ananya.

Jugal K. Kalita

Foreword

Modern society depends on the continuous availability of the networked computer systems and useful services they offer through the Internet. Almost every activity in modern society is mediated through the Internet's services. However, with the increasing use of the Internet and proliferation of Internet-based technologies, computer systems are being targeted by attackers to cause malfunction in particular services or transactions without physical interference into systems. Attacks on a computer system are activities that feed through the Internet to the target, causing detached services and media used by services, and compromise security in terms of confidentiality, availability, and integrity. Providing for these three characteristics is the main goal of a secure and stable computer system.

Data mining techniques are used to extract valid, novel, and potentially useful and meaningful patterns from large datasets with respect to a domain of interest. Network traffic is comprised of a collection of packets or NetFlow records that are described in terms of properties such as the duration of each packet or record, protocol type, and number of data bytes transferred from a source to a destination. As a packet or record is generated, or during its travel, an attacker may attempt to compromise that packet or record by inserting or modifying the header or content. It is crucial to ensure useful services in the presence of attacks. However, no one can make a networked computer system completely attack-free. This book introduces fundamental concepts of network traffic anomalies and mechanisms to detect them in both offline and online modes. The network security research community lacks real-time network traffic datasets to evaluate a newly designed mechanism or system. This book introduces a systematic approach to generate real-time network traffic datasets. It also focuses on different techniques and systems such as statistical, classification, knowledge base, cluster and outlier detection, soft computing, and combination learners to counter large-scale network attacks. This book also discusses intrusion prevention mechanisms that attempt to block the entry of attackers into a networked system. This book also discuss how to generate alarms for further diagnosis even after an attacker has intruded. In addition, this book presents tools for both attackers and defenders, with view to motivating the

development of new defensive tools. Analysis of network attack tools is important to understand attack behavior in terms of real-time parameters. Of course, network defense tools can be used to protect networks from attacks. Evaluation of a detection mechanism or system is crucial before deployment in real networks. This book provides a good discussion of evaluation measures commonly used by the network security community. Finally, it contains discussion of current issues and challenges that are yet to be overcome by network traffic security solution providers.

Jorhat, India Monowar H. Bhuyan
Napaam, India Dhruba K. Bhattacharyya
Colorado Springs, USA Jugal K. Kalita

Preface

With rapid developments in network technologies and widespread use of Internet services, the volume of worldwide network traffic is growing rapidly day by day. This continued growth of network traffic increases the number of anomalies that arise due to misconfiguration of network devices and port scans in preparation of future attacks, viruses, and worms that consume resources and bandwidth, making network services unavailable. These anomalies generate a large amount of network as well as Internet traffic. Therefore, detection and diagnosis of such anomalies are crucial tasks for network operators to ensure that resources and services are available to those who need them. Data mining is used in domains such as the business world, biomedical sciences, physical sciences, and engineering to make new discoveries from the large amounts of data that are being collected continuously. In the last few decades, many data-driven anomaly detection techniques have been developed to thwart anomalies, but most methods have limitations that deter their use in real environments. Because legitimate traffic needs to travel over the network, quickly and accurately, identifying anomalies in network-wide traffic is really important and demands development of effective and efficient detection techniques.

Unlike common network security books, this book focuses on network traffic anomaly detection and prevention with details of hands-on experience in generating real-life network intrusion datasets. Anomalies are patterns of interest to network defenders, who want to extract them from large-scale network traffic. Data mining techniques are useful in identifying anomalous patterns from large datasets. This book discusses the basic concepts of networks and causes that lead to network traffic anomalies. We present a network attack taxonomy based on attack behavior. This book also discusses generic architectures of anomaly-based network intrusion detection systems that use supervised, semi-supervised, and unsupervised machine learning techniques with details of each component. This book also discusses datasets that are needed by the research community in anomaly-based network intrusion detection systems, noting that the community lacks real-life and up-to-date network intrusion datasets. The book presents a step-by-step hands-on approach to generate real-life network intrusion datasets. Researchers may use the steps

provided either to generate a new real-life dataset or use them for evaluating anomaly detection techniques and systems. A network anomaly detection system generates an alert when it finds an anomaly in the network. This book introduces the basic concept of alerts and presents alert management techniques. It also includes a discussion of network anomaly prevention techniques, followed by concepts of network intrusion prevention. Practical tools are necessary to capture, monitor, and analyze network traffic for detection and prevention of network traffic anomalies. An attacker must always find vulnerabilities in a host or in a network to be able to mount an attack. This book presents a systematic approach discussing how to design a tool for network traffic analysis. It also includes evaluation metrics which are necessary for measuring performance of a detection technique or a system. Finally, this book enumerates current issues and challenges to attract readers who want to engage in further research in network traffic anomaly detection and prevention. We believe that this book will help individuals who want to pursue research in this general area.

Jorhat, India — Monowar H. Bhuyan
Napaam, India — Dhruba K. Bhattacharyya
Colorado Springs, USA — Jugal K. Kalita
May 2017

Acknowledgments

This book would not have been possible to complete without the people who have constantly supported, encouraged, inspired, and provided suggestions and constructive criticisms from conception. Discussions and arguments on a point or on a topic among faculty colleagues, students, and friends force us to rethink what we know. It is really difficult to acknowledge all the people who have been involved directly or indirectly on the time frame needed to complete this book. Our special thanks and sincere appreciation go to our dedicated faculty colleagues and students including Abhishek Kalwar, Saurabh Choudhury, and Ram Prajapat, to name only a few.

We would like to acknowledge the Network Security Lab of Tezpur University for providing us the facilities and support to conduct experiments in a real-time testbed environment.

We are grateful to the panel of reviewers for their constructive suggestions and critical evaluations, leading to the publication of this book. The constant support and cooperation received from our colleagues, students, and others during the period of writing this book are sincerely acknowledged.

Jorhat, India — Monowar H. Bhuyan
Napaam, India — Dhruba K. Bhattacharyya
Colorado Springs, USA — Jugal K. Kalita

Contents

Acronyms

ADAM	Automated Data Analysis and Mining
AIS	Artificial Immune Systems
ANIDS	Anomaly-Based Network Intrusion Detection System
ANN	Artificial Neural Network
AOCD	Adaptive Outlier-Based Coordinated Scan Detection
ARP	Address Resolution Protocol
ARPANET	Advanced Research Projects Agency Network
ART	Adaptive Resonance Theory
AUC	Area Under ROC Curve
BDR	Bayesian Detection Rate
BPF	Berkeley Packet Filter
BSD	Berkeley Software Distribution
CAIDA	Cooperative Association for Internet Data Analysis
CBR	Case-Based Reasoning
CLUSSLab	Cluster Labeling
CTF	Capture the Flag
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DFS	Dominating Feature Subset
DGSOT	Dynamically Growing Self-Organizing Tree
DHCP	Dynamic Host Configuration Protocol
DMitry	Deepmagic Information Gathering Tool
DNIDS	Dependable Network Intrusion Detection System
DNS	Domain Name System
DoS	Denial of Service
DMZ	Demilitarized Zone
FDDI	Fiber Distributed Data Interface
FIRE	Fuzzy Intrusion Recognition Engine
FNR	False Negative Rate
FSAS	Flow-Based Statistical Aggregation

FSD	Flow Size Distribution
FPR	False Positive Rate
FTP	File Transfer Protocol
GA	Genetic Algorithm
GNP	Genetic Network Programming
Gulp	Lossless Gigabit Remote Packet Capture with Linux
HDLC	High-Level Data Link Control
HIDE	Hierarchical Network Intrusion Detection System
HIDS	Host-Based Intrusion Detection System
HIPS	Host-Based Intrusion Prevention Systems
HOIC	High Orbit Ion Cannon
HMM	Hidden Markov Model
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDA	Intrusion Detection Agent
IDMEF	Intrusion Detection Message Exchange Format
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPS	Intrusion Prevention Systems
IRC	Internet Relay Chat
ISO	International Organization for Standardization
ITU	International Telecommunication Union
KDD	Knowledge Discovery in Databases
LAN	Local Area Network
LBNL	Lawrence Berkeley National Laboratory
LOIC	Low Orbit Ion Cannon
MAN	Metropolitan Area Network
MAS	Multi-agent System
McPAD	Multiple Classifier Payload-Based Anomaly Detector
MCS	Multiple Classifier Systems
MINDS	Minnesota INtrusion Detection System
MMIFS	Modified Mutual Information-Based Feature Selection
MTU	Maximum Transmission Unit
NCP	Network Control Program
NFIDS	Neuro-fuzzy Anomaly-Based Network Intrusion Detection System
NFS	Network File System
N@G	Network at Guard
NIC	Network Interface Card
NIDS	Network Intrusion Detection System
NIPS	Network Intrusion Prevention Systems

NMAP	Network Mapper
NSOM	Network Self-Organizing Maps
NTP	Network Time Protocol
OS	Outlier Score
OSI	Open System Interconnection
OSPF	Open Shortest Path First
PAIDS	Proximity-Assisted IDS
PAYL	Payload-Based Anomaly Detector
PC	Personal Computer
PCA	Principal Component Analysis
PoD	Ping of Death
POP3	Post Office Protocol 3
POSEIDON	Payl Over Som for Intrusion DetectiON
R2L	Remote to Local
RARP	Reverse Address Resolution Protocol
RDP	Remote Desktop Protocol
RFC	Request for Comments
RMHC	Random Mutation Hill Climbing
RMLP	Recurrent Multilayered Perceptron
RNMAP	Remote Network Mapper
ROC	Receiver Operating Characteristics
RTT	Round Trip Time
RT-UNNID	Real-Time Unsupervised Neural Network-Based Intrusion Detector
RUC	Rapid Update Cycle
SCTP	Stream Control Transmission Protocol
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOM	Self-Organizing Map
SSH	Secure Shell
SSL	Secure Sockets Layer
STAT	State Transition Analysis Tool
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
TNR	True Negative Rate
TPR	True Positive Rate
TreeCLUSS	Tree-Based Clustering Scheme
TRW	Threshold Random Walk
TTL	Time to Live
TUIDS	Tezpur University Intrusion Detection System
U2R	User to Root
UDP	User Datagram Protocol

URL	Uniform Resource Locator
VLAN	Virtual Local Area Network
WAN	Wide Area Network
WNN	Wavelet Neural Network
XMPP	Extensible Messaging and Presence Protocol
XSS	Cross-Site Scripting

Chapter 1
Introduction

With advances in network technologies, the variety and volume, Internet services that are provided by commercial, nonprofit or governmental organizations undergo constant growth, causing commensurate and often exposure expansion in network traffic. This continued growth is accompanied by an increasing number of anomalies such as misconfigurations of network devices, port scans in preparation for future attacks, viruses and worms that consume resources and spread automatically, and denial of service (DoS) attacks that make network services unavailable. These anomalies drive a large fraction of Internet traffic that is unwanted and prevent legitimate users from accessing network resources in an optimal manner. Therefore, detecting and diagnosing these benign nuisances or actual threats to the well-being of the network services are crucial tasks for network operators to ensure that Internet resources remain available to those who need them and when they need them. Because legitimate traffic must be able to travel efficiently, quickly, and accurately, identifying anomalies in network traffic is important, requiring development of good detection techniques. Thus, anomalies are patterns of crucial interest to network defenders, who want to extract them quick from vast amount of network traffic data, with the goal of blocking them or preventing them in future. Data mining techniques have been popular in recent years, in detecting these harmful patterns in large volumes of data. Data mining is used in many other application areas as well, e.g., the business world, medical sciences, physical sciences, and engineering to make new discoveries. Extensive studies have been performed in applying data mining techniques to network traffic anomaly detection and prevention, but most methods [5, 15, 24] have limitations that make it difficult to use them in real environments. This book explores the application of data mining techniques in detection and prevention of network traffic anomalies. In addition, this book also explores different network anomaly detection and prevention tools from the perspective of both attackers and network defenders.

M.H. Bhuyan et al., *Network Traffic Anomaly Detection and Prevention*,
Computer Communications and Networks, DOI 10.1007/978-3-319-65188-0_1

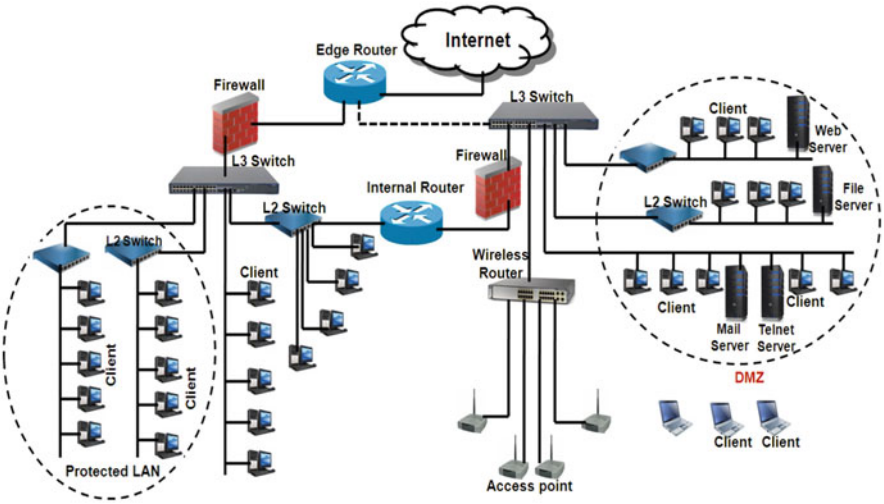


Fig. 1.1 A typical view of an enterprise network with DMZ

1.1 Modern Networks and the Internet

A network is a group of computer systems that are connected together to allow sharing of resources including files, printers and storage media, and sharing of services such as Web and DNS services. There are two main aspects to setting up a network: (a) the hardware used to connect the systems together and (b) the software installed on the hosts to allow them to communicate [20]. A typical view of a large network with a demilitarized zone (DMZ) is given in Fig. 1.1. A demilitarized zone is a network segment located between a secure local network and insecure external networks (i.e., Internet). A DMZ usually contains servers that provide services to users on the external network such as Web, mail, and DNS servers, which must be hardened systems. Two firewalls are typically installed to form the DMZ.

A typical network involves several hosts connected through network devices so that users can share data and resources with each other. A device or system that is connected to the network is known as a host. The workstation is a general purpose host with high-end configuration used for business, technical, and scientific applications. The server is a special host that contains more disk space and memory than are found on clients (viz., hosts, workstations). A server has special software installed that allows it to provide the intended function. It provides services such as file and print services, serving Web pages to clients, controlling remote access and security to clients. The Internet is the worldwide network of computers accessible to anyone through protocols such as HTTP, FTP, and SMTP. Day by day, computer users have become increasingly dependent on the Internet as users of many useful and entertainment services that are available. Some statistics on the world's Internet users are given in Fig. 1.2, as reported by the International Telecommunication Union (ITU).[1]

[1] http://www.itu.int/en/Pages/default.aspx.

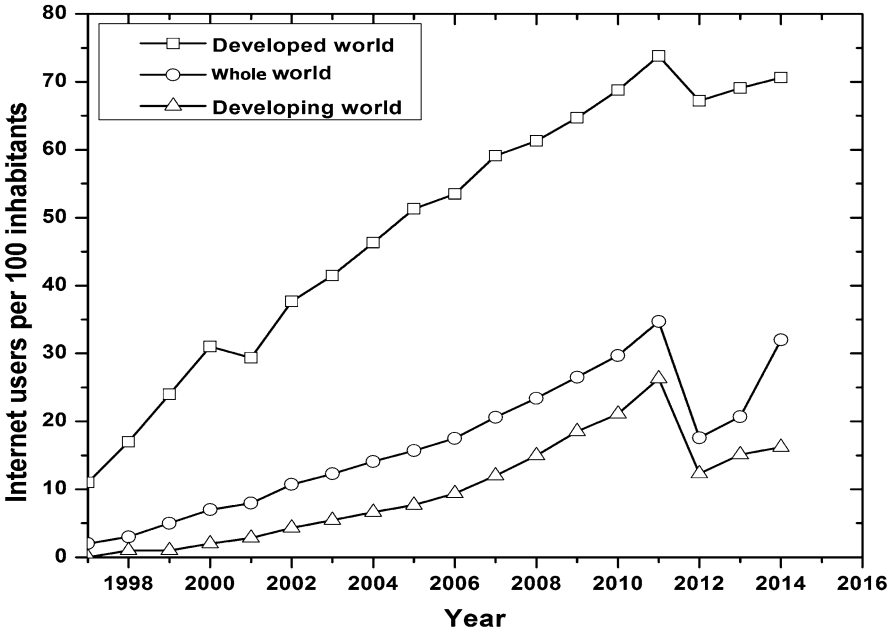


Fig. 1.2 The growth of world's Internet users

Network *vulnerabilities* are the inherent weaknesses in the design, configuration, or implementation of a computer network that render it susceptible to security threats. The growth of network vulnerabilities as reported in [23] is shown in Fig. 1.3. Threats may arise from exploitation of design flaws in the hardware and software components of computer network systems [21]. Systems may also be incorrectly configured and therefore vulnerable to attack. Vulnerabilities of this kind generally occur from inexperience, insufficient training, or careless half-done work by network administrators. Another source of vulnerability is poor management such as inadequate policies and procedures and insufficient checks of the network systems.

1.2 Network Traffic and Its Characteristics

Network traffic is defined as data present in a network when it is in active mode. In a computer network, at each and every moment, communication devices request access to resources for services and providers of services respond to them. A resource may not be available at the exact moment when a request is made. There is constant information exchange throughout the network in the form of requests, services, release and control data. Here, data actually means the millions or billions

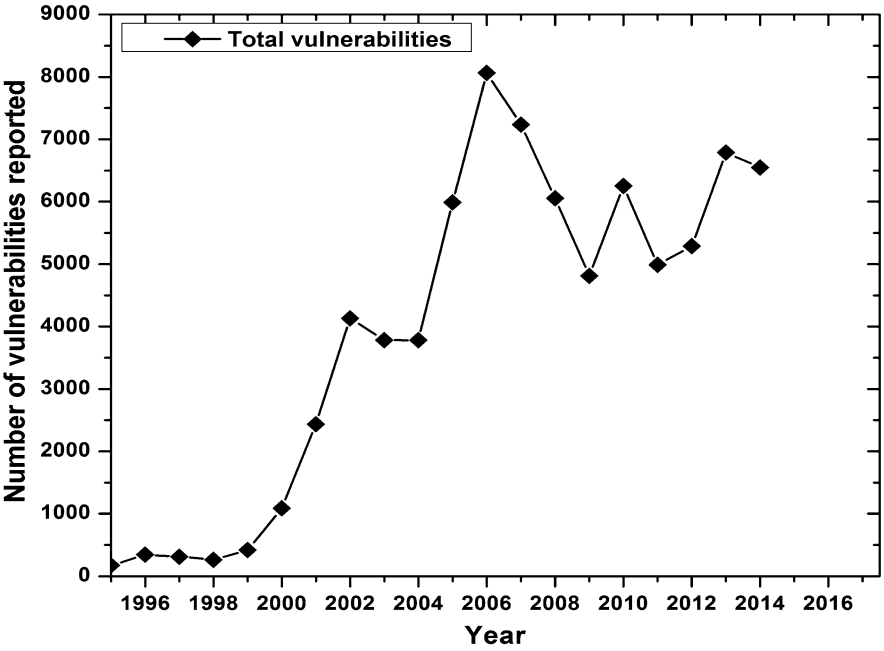


Fig. 1.3 The growth of reported network vulnerabilities

of packets or small units of data moving around the network. The volume of packets represents the load in the network, resulting in congestion in the network if it goes beyond the limits of communication devices [16]. An enterprise network must control the traffic behavior within its bounds to ensure that resources are used optimally. Controlling network traffic requires limiting bandwidth to certain applications, guaranteeing minimum bandwidth to others and assigning high or low priorities to various types of traffic. This is known as traffic management.

Traffic analysis is usually used to characterize healthy traffic to analyze if resources are being used efficiently. There are mainly two types of traffic based on the monitoring points. They are LAN (Local Area Network) traffic and WAN (Wide Area Network) traffic. LAN interconnects devices within limited geographical areas and provides fast data transfer rate with high accuracy. WAN interconnects different small area networks such as LANs. These LANs cover large physical areas with lower data transfer rate in comparison to LAN. In comparison to LAN traffic, WAN traffic varies from time to time as the use of network resources or services changes [16]. WAN traffic may be classified as Random traffic, Internet traffic and Bursty traffic. Random traffic seems to occur within certain random time intervals. Internet traffic follows the Poisson model. Finally, Bursty traffic has sudden bursts over either long or short periods of time.

Network traffic characteristics play an important role in detection or identification of maliciousness of traffic. Network traffic analysis may be possible at

both packet or NetFlow level. NetFlow is an IP (Internet Protocol) traffic flow technology introduced by Cisco[2] for collecting and monitoring traffic data generated by NetFlow-enabled routers or switches. NetFlow traffic analysis is faster than packet level analysis of traffic. It is due to fewer parameters associated with NetFlow traffic. Some important characteristics that are used to describe network traffic are (a) time duration or session duration; (b) dimensionality in terms of number of packet or flow level parameters, which is usually high; (c) large in volume of traffic; (d) noisy; and (e) traffic rate.

1.3 Network Traffic Anomalies

Traffic anomalies are instances in data that do not conform to the behavior exhibited by normal traffic. Traffic anomalies in a network can be defined as any network events or operations that deviate from normal network behavior. They happen due to the growing number of network-based attacks or intrusions. The recent growth of Internet threats[3] is given in Fig. 1.4. Network threats may occur due to many reasons including (i) malicious activities that take advantage of normal network services, (ii) network overload, (iii) device malfunction, and (iv) compromises in different network parameters such as protocol, port.

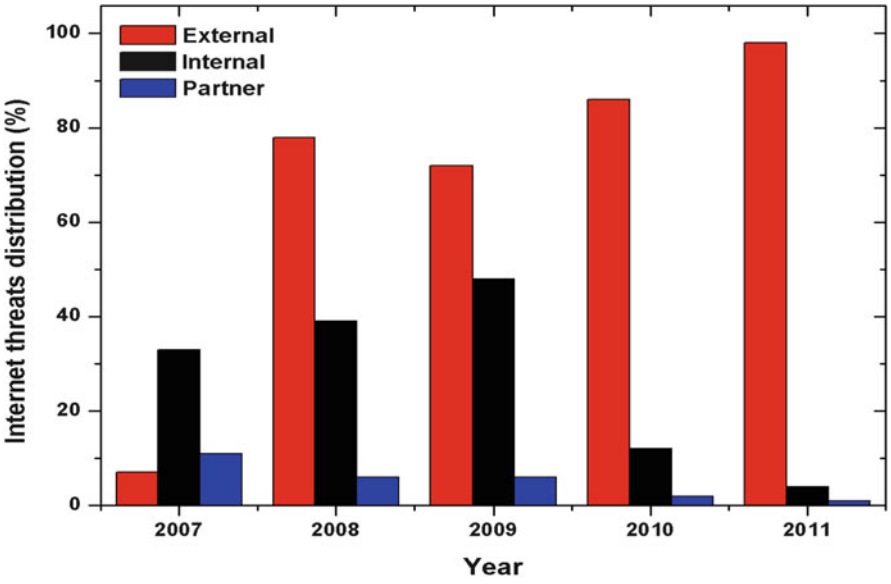


Fig. 1.4 The growth of Internet security threats

[2] http://www.cisco.com.

[3] http://www.verizon.com/enterprise/databreach.

1.4 Sophistication in Network Anomalies and Their Detection

Anomaly detection attempts to find in network traffic data patterns, which do not conform to expected normal behavior. The importance of anomaly detection is due to the fact that anomalies in data translate to significant (and often critical) actionable information in a wide variety of application domains [11]. For example, an anomalous traffic pattern in a computer network could mean that a hacked computer is sending out sensitive data to an unauthorized host. Anomalies in a network may be caused by different reasons. As stated in [21], there are two broad categories of network anomalies: (a) performance-related anomalies and (b) security-related anomalies. Various examples of performance-related anomalies are broadcast storms caused when network nodes send large amount of data to all other nodes, transient congestions that happen suddenly and fast for a short time, babbling nodes that sent out data increasingly, paging across the network, and file server failure. Security-related network anomalies may be due to malicious activities of the intruder(s) by intentional flooding of the network with unnecessary traffic to hijack the bandwidth so that legitimate users are unable to receive service(s). Our book is concerned with security-related network anomalies only. Currently, anomaly-based network intrusion detection is the most successful intrusion detection technique. It is currently a principal focus of research and development in the field of intrusion detection. Various systems with ANIDS (anomaly-based network intrusion detection system) capabilities are becoming available, and many new schemes are being explored. However, the subject is far from mature, and key issues remain to be solved before wide-scale deployment of ANIDS platforms becomes practicable.

Advances in networking technology have enabled us to connect distant corners of the globe through the Internet for sharing vast amount of information. However, as mentioned earlier along with this advancement, threats from spammers, attackers, and criminal enterprises are also growing in warp speed [15]. Normally, an intrusion attempts to compromise the confidentiality, integrity, or availability of a system, by bypassing the security mechanisms of a host or a network. As a result, security experts use intrusion detection technology to keep secure the infrastructure of large enterprises. An intrusion detection system (IDS) may be defined [2] as follows.

Definition 1.1 An intrusion detection system (IDS) is a software and/or hardware system that monitors events occurring in a computer system or a network and analyzes them for signs of intrusion by unwanted traffic, i.e., malicious.

Intrusion detection systems are divided into two broad categories: misuse detection [18] and anomaly detection [22] systems. *Misuse* detection can detect only known attacks based on signatures they have already created and stored. Thus, dynamic signature updation is important, and this is why new attack definitions are frequently released by IDS vendors. However, misuse-based systems cannot cope with the rapidly growing number of vulnerabilities and exploits. On the other hand, *anomaly*-based detection systems are designed to capture any deviation from

the profiles of normal behavior. They are more suitable than misuse detection for detecting unknown or novel attacks without any prior knowledge. But normally such systems generate a large number of false alarms.

There are four commonly used machine learning approaches for detecting intrusions or anomalies in network traffic [5]: (i) supervised, (ii) semi-supervised, (iii) unsupervised, and (iv) hybrid. In the *supervised* approach [10, 12, 25], a predictive model is developed based on a training dataset that contains normal and attack data instances. Any unseen data instance is compared against the model to determine which class it belongs to. In the *semi-supervised* approach [3, 7, 26], the training data instances contain only the normal class. Data instances are not labeled for the attack class. Many approaches are used to build the model for the class corresponding to normal behavior. This model is used to identify anomalies in the test data. In the *unsupervised* approach [4, 9, 13, 17], the model does not require any training data and thus is potentially most widely applicable. However, due to lack of labeled data, the results may be less useful. Finally, the *hybrid* approach [1, 14, 27] normally exploits the features of all of the above to get effective and efficient performance in detecting network anomalies on a large scale. All techniques usually make the implicit assumption that normal instances are far more frequent than anomalies in the test data. If this assumption is not true, such techniques suffer from high false alarm rates. The first two cases require training on labeled instances for finding anomalies. But getting a large amount of labeled normal and attack training instances may not be practical in many or most situations. In addition, generating a set of true normal instances with all the variations is an extremely difficult task. This book discusses the development of some effective and excellent network traffic anomaly detection methods to detect known as well as unknown attacks with high detection rate and low false-positive rate compared to competing methods.

1.5 Network Traffic Anomaly Prevention

Prevention of network anomalies in high-speed networks is a challenging task. Network intrusion prevention systems (NIPS) are proactive mechanisms for preventing malicious network traffic before entering the network. These systems drop malicious traffic automatically before it causes any harm to the network rather than raising an alarm afterward. Similar to firewalls, NIPS systems are effective and scalable because they perform deep packet inspection to ensure the legitimacy of the network traffic when it enters the network. Similar to IDSs, IPSs are also classified into two types: signature-based NIPS and anomaly-based NIPS [6]. A signature-based NIPS uses predefined signatures of known attacks. The prevention mechanism matches the signatures by using a high-speed pattern matching algorithm. On the other hand, an anomaly-based NIPS is designed to protect against zero-day attacks, which are new and unknown malicious traffic.

1.6 Data Mining Fundamentals

Data mining has gained a great deal of attention in the information industry and in the society as a whole in recent years due to the wide availability of huge amounts of data and the imminent need for turning such data into useful information and knowledge. Based on Tan et al. [8, 19], data mining is defined as follows.

Definition 1.2 Data mining is the process of automatically discovering useful information in large repositories. Data mining techniques are deployed to scour large databases in order to find novel and useful patterns that might otherwise remain unknown.

The term *Knowledge Discovery in Databases* (KDD) refers to the process of converting raw data into useful information or knowledge. Data mining is a step in the KDD process and applies a variety algorithms for extracting patterns from data. In addition, the KDD process has additional steps including data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, and proper interpretation of the results of mining to ensure useful knowledge is derived from the data.

1.7 Data Mining in Network Traffic Anomaly Detection and Prevention

In general, data mining tasks are partitioned into two major categories: predictive and descriptive. Predictive mining performs inference on the current data in order to make future predictions. Descriptive mining characterizes the general properties of the data and underlying relationships among them. Some of the most important data mining tasks [8, 19] are discussed below.

1. *Classification and Regression*: Classification is the process of classifying a data instance into one of several predefined categorical classes based on the training set containing known observations. A regression task begins with data instances for which the numerical target values are known. The relationships between predictors and the target are summarized in a regression model that can be applied to new data instances in which the target values are unknown.
2. *Cluster Analysis*: Cluster analysis seeks to find groups of closely related data objects based on relationships among them. The greater the similarity within a group and the greater the difference between groups, the better is the clustering.
3. *Association Analysis*: Association analysis is the task of efficiently discovering the most important and the most strongly associated feature patterns in data. The discovered patterns are represented in the form of implication rules.
4. *Evolution Analysis*: Data evolution analysis describes and models regularities or trends in objects whose behavior changes over time.

5. *Outlier Detection*: Outlier detection refers to finding those observations whose characteristics are significantly different from the rest of the data. These observations are known as outliers.

Three useful data mining tasks for identification and prevention of network traffic anomalies are clustering, classification, and outlier mining. Recently, there has been a realization that data mining can have significant impact on network security, especially network traffic analysis. Because most security systems are developed based on the interestingness of traffic patterns, it is important to discover interesting patterns correctly from large datasets as well as efficiently analyze them using effective data mining techniques. Data mining, if used properly, can detect and prevent known as well as unknown attacks.

1.8 Contributions of This Book

To detect and prevent network traffic anomalies, we must have a good understanding of the basics of networking. Thus, this book discusses the basic concepts of networks and causes that lead to network traffic anomalies. This book presents data mining techniques that have been used to successfully detect and prevent network traffic anomalies. It includes a taxonomy of network attacks followed by a discussion of the motivations that may prompt an attack. It introduces generic architectures of various anomaly-based network intrusion detection systems (ANIDSs) with pros and cons.

This book also discusses our hands-on experience in generating large real-life network intrusion datasets. It is necessary for the network security research community to evaluate recently developed detection and prevention techniques and systems using large intrusion datasets, and learning how to create their own intrusion datasets will be useful to such researchers. The book presents various network traffic anomaly detection techniques and systems under several categories such as statistical approaches, classification, clustering and outlier detection, soft computing, knowledge-based approaches, and combination learners. To better understand these techniques and systems, we also include detailed analysis of each category with pros and cons. The reader will be able to acquire knowledge about how to design an intrusion detection technique and incorporate it in a system. The book introduces at least one generic design for network traffic anomaly detection techniques and systems. Heterogeneous ANIDSs generate alerts, and proper analysis of each alert to detect and prevent actual attacks is really important. So, this book covers alert management and alert correlation techniques for automatic analysis of raw alerts without any specific prior knowledge of the actual alert. The reader will be able to learn how to analyze and correlate alerts without prior information. The reader will learn about the specific tools that may help to protect networks from attacks and sometimes even prevent attacks from happening. The book also discusses how an attacker initiates an attack using different tools. Finally, the reader will be briefed

on current research issues and challenges that still need to be addressed to protect networks from attacks. Thus, this book provides a comprehensive discussion on network traffic anomaly detection and prevention from basic concepts to in-depth analysis in a single volume. The main contributions of this book are as follows.

- *Networks and Network Traffic Anomalies*: This book covers basic concepts of networks, properties of network traffic, and traffic management in high-speed networks. It presents the types of anomalies that occur due to misconfiguration of networks or misuse of network access rights. It presents the motivations of attackers, followed by a taxonomy of network attacks, along with enumeration of different network vulnerabilities. A generic architecture of each anomaly-based network intrusion detection system (ANIDS) is provided with detailed discussion of each component. In addition, the book discusses different aspects of network traffic anomaly detection.
- *Hands-on Approach to Generate Real-life Intrusion Datasets*: To evaluate network traffic anomaly detection techniques and systems, we should have real-life benchmark intrusion datasets and also datasets that reflect the traffic our specific networks encounter. New and updated datasets with instances of recent attacks are necessary to the network security research community to keep their defense mechanisms up-to-date. So, this book discusses a systematic hands-on approach that are used to generate large network intrusion datasets. This book also discusses existing datasets in three different categories: (a) synthetic datasets, (b) benchmark datasets, and (c) real-life datasets. Then explains how to launch real-life attacks, capture traffic, perform feature extraction, feature correlation, and label traffic instances as normal or attack to generate the final datasets. We have prepared three different datasets using our environment, viz., (a) TUIDS intrusion dataset (b) coordinated scan dataset, and (c) DDoS flooding attack dataset. These datasets are available for performance evaluation of techniques for detection and prevention of network attacks.
- *Network Traffic Anomaly Detection Techniques and Systems*: This book presents a detailed study of network anomaly detection techniques and systems under six different categories: statistical approaches, classification, clustering and outlier detection, soft computing, knowledge-based, and combination learners. This book provides at least one generic design for each detection technique and system. We identify common pitfalls in network traffic anomaly detection techniques and systems. In addition, we compare various network traffic anomaly detection techniques and systems by considering several performance parameters. Such comparison is helpful in choosing a specific method to use for a specific purpose, as well as to extend and adapt them as needed.
- *Alert Management and Network Anomaly Prevention Techniques*: IDSs generate intrusion alerts. We should analyze each raw alert to extract actual attack information. This book presents preliminary concepts on alert management and anomaly prevention techniques. This book also discusses alert preprocessing, alert correlation, and alert post-processing to achieve network defender's goals quickly. An intrusion prevention system is a highly refined firewall, able to deny

access to hostile traffic while allowing legitimate traffic to pass through. We present some network anomaly prevention techniques to help network defenders prevent attacks before they start to damage the entire network.

- *Practical Tools*: Practical tools are necessary to capture, monitor, and analyze network traffic for detection and prevention of attacks. Similarly, attackers also use tools to find vulnerabilities in networks or hosts. We discuss our hands-on approach to develop a network traffic monitoring and analysis tool for both attackers and network defenders. In addition, we enumerate existing tools for attackers and network defenders and compare them based on relevant characteristics.
- *Evaluation Criteria*: Evaluation measures are important components of a detection technique or a system. Evaluation measures are used to determine the quality of a detection technique or a system. We discuss various evaluation criteria such as accuracy, performance, completeness, timeliness, reliability, quality, and Rapid Update Cycle (RUC) area for network anomaly detection and prevention. Accuracy can be measured using different metrics such as sensitivity and specificity, Receiver Operating Characteristics (ROC) curve (also area under ROC curve, i.e., AUC), misclassification rate, confusion matrix, precision, recall, and F-measure. This book covers all these evaluation metrics and explains the use of each with examples.
- *Issues and Challenges*: This book enumerates open issues and challenges in network traffic anomaly detection and prevention. This will help researchers who want to pursue future work in this area. It also covers how attackers' sophistication is increasing as the modern Internet grows in size and utility.

1.9 Organization of the Book

The structure of the book mimics the list of contributions above in that related contributions are organized into coherent chapters and parts. This book has four major parts with a total of eight chapters, organized as follows.

- In Part I, there are two chapters that discuss the basic concepts and fundamental terms related to networks and network traffic anomalies. Chapter 1 is the current chapter, where we discuss key terms in network traffic anomaly detection and prevention. It also discusses how data mining techniques can be applied to detect and prevent network anomalies. Chapter 2 introduces the basic concepts of networks, layered architectures, and sources of vulnerabilities because an understanding of these topics is necessary to detect network traffic anomalies. This chapter also provides the background needed to describe network traffic anomaly detection and prevention. Topics include anomalies in networks, anomaly detection and prevention, classification of ANIDSs, aspects of network anomalies, and network attacks.

- In Part II, there is only one chapter. Chapter 3 introduces our hands-on experience using a testbed to generate real-life network intrusion datasets. It discusses the importance of datasets, requirements that datasets should satisfy, and existing datasets. We describe how to generate legitimate and attack traffic considering different network environments or scenarios. The chapter explains how to go about capturing traffic, identifying raw parameters, extracting and correlating features, and labeling each record as normal or attack at both packet and flow levels. We have created three datasets for the network security research community, as mentioned earlier in the contributions of this book in Sect. 1.8.
- Part III of this book contains three chapters. This part covers topics such as design, architecture, monitoring and analysis for detection, and prevention of network traffic anomalies. It also covers tools for network defenders as well as for attackers and how these tools work to achieve a particular goal. Chapter 4 presents a variety of network traffic anomaly detection techniques and systems. This chapter also provides detailed design of at least one generic technique and a well-regarded system in each category. It discusses highly cited techniques and systems with pros and cons and gives a detailed comparison in terms of several parameters. Our discussion is organized as per the developed taxonomy. Chapter 5 discusses alert management and network anomaly prevention techniques. To find actual details of a network attack, alert analysis is important. We cover alert correlation techniques, network anomaly prevention techniques, and systems, with pros and cons and detailed comparison in terms of relevant parameters. Chapter 6 discusses tools for network defenders as well as attackers. The description is practical based on our hands-on experience for each. This chapter also covers how to develop a tool to detect network anomalies in real-life high-speed networks.
- Part IV contains two chapters. The detection and prevention techniques and systems discussed in the previous chapters need to be evaluated in terms of appropriate metrics. Chapter 7 discusses evaluation measures for network anomaly detection and prevention. Each metric is illustrated using examples. Finally, Chapter 8 summarizes the work described in detail in the previous chapters followed by a presentation of open issues and challenges in network traffic anomaly detection and prevention. The chapter also presents future research directions in the network anomaly detection and prevention domain, outlying work that needs to be done in the years to come.

References

1. Aydin, M.A., Zaim, A.H., Ceylan, K.G.: A hybrid intrusion detection system design for computer network security. Comput. Electr. Eng. **35**(3), 517–526 (2009)
2. Bace, R., Mell, P.: Intrusion detection systems. Tech. Rep. SP800-31, NIST Special Publications, US Department of Defence, USA (2001)
3. Burbeck, K., Nadjm-Tehrani, S.: Adaptive real-time anomaly detection with incremental clustering. Information Security Technical Report **12**(1), 56–67 (2007)

4. Casas, P., Mazel, J., Owezarski, P.: Unsupervised network intrusion detection systems: detecting the unknown without knowledge. Comput. Commun. **35**(7), 772–783 (2012)
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**(3), 1–58 (2009)
6. David, S.T.: Network intrusion prevention systems: signature-based and anomaly detection. Ph.D. thesis, The Hebrew University of Jerusalem, April (2011)
7. Erman, J., Mahanti, A., Arlitt, M., Cohen, I., Williamson, C.: Semi-supervised network traffic classification. SIGMETRICS Perform. Eval. Rev. **35**(1), 369–370 (2007)
8. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: The KDD process for extracting useful knowledge from volumes of data. Commun. ACM **39**(11), 27–34 (1996). doi:10.1145/240455.240464
9. Jiang, S., Song, X., Wang, H., Han, J.J., Li, Q.H.: A clustering-based method for unsupervised intrusion detections. Pattern Recogn. Lett. **27**(7), 802–810 (2006)
10. Khreich, W., Granger, E., Miri, A., Sabourin, R.: Adaptive ROC-based ensembles of HMMs applied to anomaly detection. Pattern Recogn. **45**(1), 208–230 (2012)
11. Kumar, V.: Parallel and distributed computing for cybersecurity. IEEE Distrib. Syst. Online **6**(10) (2005)
12. Laskov, P., Gehl, C., Krüger, S., Müller, K.R.: Incremental support vector learning: analysis, implementation and applications. J. Mach. Learn. Res. **7**, 1909–1936 (2006)
13. Noto, K., Brodley, C., Slonim, D.: FRaC: a feature-modeling approach for semi-supervised and unsupervised anomaly detection. Data Min. Knowl. Discov. **25**(1), 109–133 (2012). doi:10.1007/s10618-011-0234-x
14. Panda, M., Abraham, A., Patra, M.R.: Hybrid intelligent systems for detecting network intrusions. Secur. Commun. Netw. **8**(16), 2741–2749 (2015). http://dx.doi.org/10.1002/sec.592
15. Patcha, A., Park, J.M.: An overview of anomaly detection techniques: existing solutions and latest technological trends. Comput. Netw. **51**(12), 3448–3470 (2007)
16. Shimonski, R.: The Wireshark field guide: analyzing and troubleshooting network traffic. Syngress Media, U.S. (2013)
17. Song, J., Takakura, H., Okabe, Y., Nakao, K.: Toward a more practical unsupervised anomaly detection system. Inf. Sci. **231** (2011). http://dx.doi.org/10.1016/j.ins.2011.08.011
18. Su, M.Y.: Using clustering to improve the KNN-based classifiers for online anomaly network traffic identification. J. Netw. Comput. Appl. **34**(2), 722–730 (2011)
19. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining, 4th edn. Addison-Wesley, Pearson Education, India (2009)
20. Tanenbaum, A.: Computer Networks. Prentice Hall Professional Technical Reference, 4th edn. Prentice Hall, Pearson Education, India (2002)
21. Thottan, M., Ji, C.: Anomaly detection in IP networks. IEEE Trans. Signal Process. **51**(8), 2191–2204 (2003)
22. Toosi, A.N., Kahani, M.: A new approach to intrusion detection based on an evolutionary soft computing model using Neuro-fuzzy classifiers. Comput. Commun. **30**(10), 2201–2212 (2007)
23. Wood, P., Egan, G., Haley, K., Tran, T., Cox, O.: Internet security threat report. Tech. Rep. 17, Symantec, USA (2012)
24. Wu, S.X., Banzhaf, W.: The use of computational intelligence in intrusion detection systems: a review. Appl. Soft Comput. **10**(1), 1–35 (2010)
25. Yi, Y., Wu, J., Xu, W.: Incremental SVM based on reserved set for network intrusion detection. Expert Syst. Appl. **38**(6), 7698–7707 (2011)
26. Zhang, J., Chen, C., Xiang, Y., Zhou, W.: Semi-supervised and compound classification of network traffic. In: Proceedings of the 32nd International Conference on Distributed Computing Systems Workshops, pp. 617–621 (2012). doi:10.1109/ICDCSW.2012.12
27. Zhang, J., Zulkernine, M.: A hybrid network intrusion detection technique using random forests. In: Proceedings of the 1st International Conference on Availability, Reliability and Security, pp. 262–269. IEEE, CS, USA (2006). doi:10.1109/ARES.2006.7

Chapter 2
Networks and Network Traffic Anomalies

Before discussing the actual detection and prevention of network traffic anomalies, we must introduce fundamental concepts on networks, network traffic, and traffic measurement. Therefore, this chapter is comprised of two parts. The first part discusses components of networks, topologies, and layered architectures followed by protocols used, metrics to quantify network performance, and ideas in network traffic management. It also introduces how we represent normal and attack traffic. The second part of this chapter discusses network anomalies, causes of anomalies, and sources of anomalies followed by a taxonomy of network attacks, a note on precursors to network anomalies, and other aspects of network traffic anomalies.

2.1 Networking Fundamentals

A network is an interconnection of a set of devices, hosts, nodes, and end-computing devices that can communicate among themselves. For successful communication, it is necessary to follow certain rules to ensure efficient communication among the interconnected devices. Such a set of rules is known as a protocol. A modern data communication system needs many such protocols to communicate with each other over heterogeneous networks. This interconnection of one device to many other devices is known as networking. An example of network is given in Fig. 2.1.

2.1.1 Components of a Network

A network contains both hardware and software components to communicate between stations.

M.H. Bhuyan et al., *Network Traffic Anomaly Detection and Prevention*,
Computer Communications and Networks, DOI 10.1007/978-3-319-65188-0_2

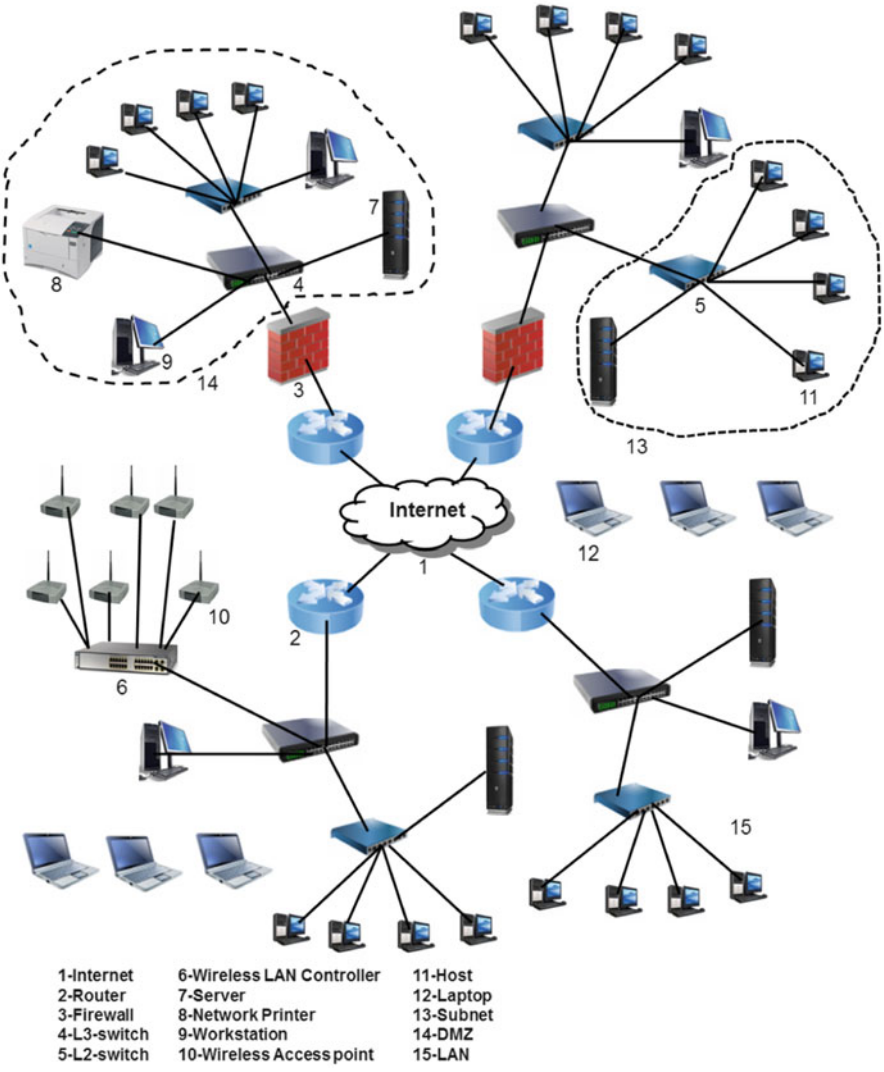


Fig. 2.1 An example of network

2.1.1.1 Software Components

Software components are used to share files, programs, printers, and operating systems. We introduce two examples:

- *Local operating system*: It is the master software system running on a personal computer (PC), and it controls access to the CPU shared files, printers, disk storage, and processes running on the computer. Examples include Windows XP, Windows 7, Windows 10, Linux, and MAC OS.

- *Network operating system*: This is an operating system normally installed on a router, switch, or firewall to support communication over the network.

2.1.1.2 Hardware Components

The primary focus of computer networks is on its hardware components which enable the PCs to receive services. The following are a few of the essential hardware components needed to run a network:

- *Network interface card (NIC)*: Each PC in a network has an expansion card called the network interface card. The NIC prepares and sends data, receives data, and controls data flow between the PCs within the network. On the sender side, the NIC passes frames of data to the physical layer that transmits the data to the physical link. In contrast, on the receiving side, the NIC passes bits received from the physical layer and processes them based on its contents.
- *Transmission media*: These are used to interconnect PCs in a network. Transmission media consist of twisted-pair cables, coaxial cables, and fiber-optics cables.
- *Clients*: Clients are PCs of normal specification that are used to access the network and the shared resources. Clients usually send requests and receive services from the server.
- *Servers*: Servers are high-end PCs that hold shared files, programs, and a network operating system. A server provides access to network resources to all users. There are many kinds of servers such as file servers, mail servers, print servers, database servers, web servers, and many more.
- *Network connecting devices*: Network connecting devices include NICs, hubs, switches, repeaters, bridges, and routers. These devices are used to connect two or more PCs through transmission media.

2.1.2 Network Criteria

A network must be able to meet certain criteria to provide services to the PCs connected in a network. The most important criteria among them are performance, reliability, and security:

(a) *Performance*: Performance can be measured in many ways. It can be measured in terms of transit time, response time, throughput, and delay. *Transit time* is the amount of time required for a message to travel from one device to another. *Response time* is the time elapsed between when a query is sent and when a response is received. Other factors that can impact on performance are as follows:

  (i) Number of users
  (ii) Types of transmission media

(iii) Ability of connecting hardware
(iv) Efficiency of software

Throughput is the amount of useful data transmitted per unit time. It is equal to the bandwidth if there is no protocol involved. However, in most practical cases, throughput is always less than the bandwidth due to two reasons: protocol overhead and protocol waiting time. *Delay* is the amount of time taken by a packet to reach destination from source. It is also known as the round-trip time (RTT) or latency. It has additional components such as processing delay, queuing delay, transmission delay, and propagation delay.

(b) *Reliability*: A network needs to be reliable, characterized by low frequency of failure. It can be measured by the number of failures that occur and the amount of time taken to recover from a failure.
(c) *Security*: Security refers to protection of data when it is transmitted over the network from access by unauthorized users. This is the main focus of this book.

2.1.3 Types of Connections

A network is composed of two or more devices connected through links. A link is a communication pathway to transfer data from one device to another. There are two possible ways to connect the devices:

- *Point-to-point connection*: A point-to-point connection provides a dedicated link between two devices, as shown in Fig. 2.2a. The entire capacity of the link is reserved for transmission between these two devices only.
- *Multipoint connection*: Multipoint connection, also known as multi-drop connection, shares a specific link with multiple devices, as shown in Fig. 2.2b. If multiple devices share the same link simultaneously, it is known as a *spatially shared* connection. But if users share turn by turn, then it is known as a *time-sharing* connection.

2.1.4 Network Topologies

The topology of a network is a geometric representation showing the relationship among all the links and linked devices (usually called nodes or PCs or hosts) to one another. In data communication, two or more devices are connected to form a link whereas two or more links form a topology. Topologies can be of two types: (a) physical and (b) logical. A *physical topology* represents the geometric shape of the configuration among the connected devices in the network. A *logical topology* represents the way of data flow from one system to another. The logical topology

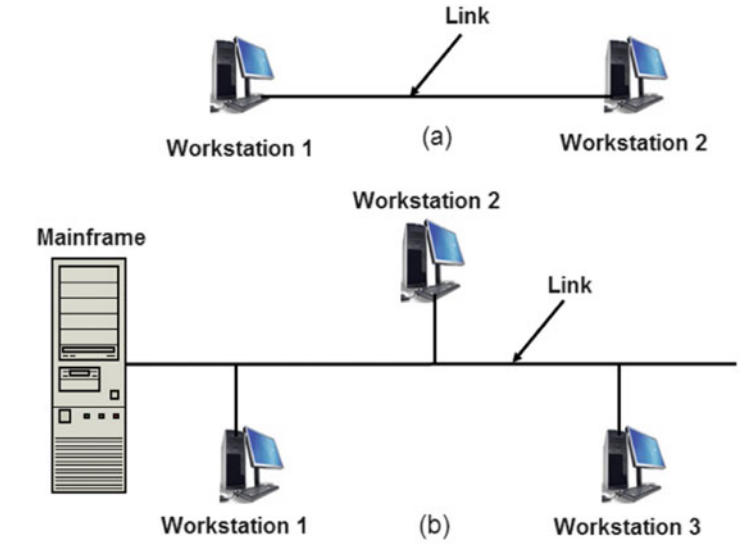


Fig. 2.2 Types of network connection (**a**) Point-to-Point and (**b**) Multi-point

Fig. 2.3 Illustration of bus topology

may not be the same as the physical topology of the network. It may be dynamically configured in the routers. There are five basic topologies, viz., bus, ring, star, mesh, and tree:

- *Bus topology*: A bus network has a multipoint connection where nodes are connected to a single cable called the *bus*. The bus topology has one medium shared by all nodes. The bus topology is usually used when a network installation is small, simple, or temporary as shown in Fig. 2.3. It is widely used in LAN. The speed of the bus topology is slow because only one PC can send a message at a time. A PC must wait until the bus is free before it can transmit. IEEE 802.3 is a broadcast bus network operating in the range 10 Mbps-10 Gbps. Token bus networks are defined by the IEEE 802.4 standard.
- *Ring topology*: A ring topology also has a multipoint connection, where all nodes are interconnected and work in tandem to form a closed loop. Each node in the ring contains a repeater. A signal is passed along the ring in one direction

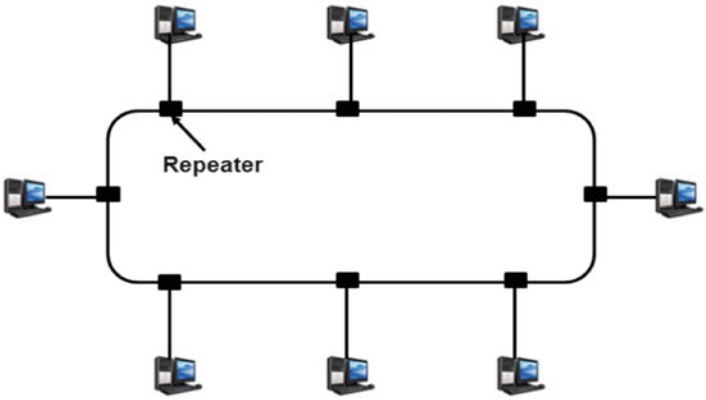
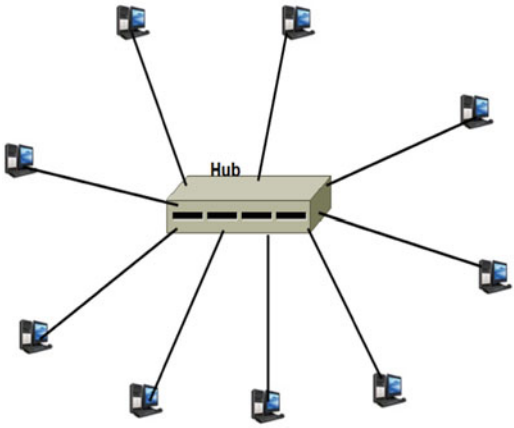


Fig. 2.4 Illustration of ring topology

Fig. 2.5 Illustration of star topology



until it reaches the destination. When a node receives a signal, it regenerates the bits and passes them along. For example, IEEE 803.5 token ring and FDDI (Fiber Distributed Data Interface) ring use this network topology. An example ring topology is shown in Fig. 2.4.

- *Star topology*: The star topology maintains a dedicated point-to-point link only to the central controller. The controller is either a hub or a switch. If it is a switch, it is known as a star switched network. A star topology does not allow direct traffic between nodes. The controller facilitates exchange of information. If one node wants to send data to another, it sends the data to the controller that transmits the data to the node in turn, as shown in Fig. 2.5.
- *Mesh topology*: A mesh topology maintains a set of dedicated point-to-point connections in such a way that each node can directly reach another node in the network. The presence of dedicated connections means that a link carries traffic only between the two devices it connects. Therefore, a fully connected

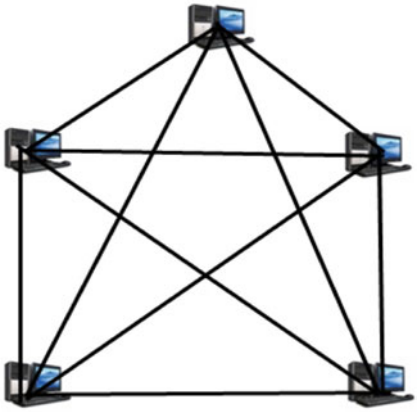


Fig. 2.6 Illustration of mesh topology

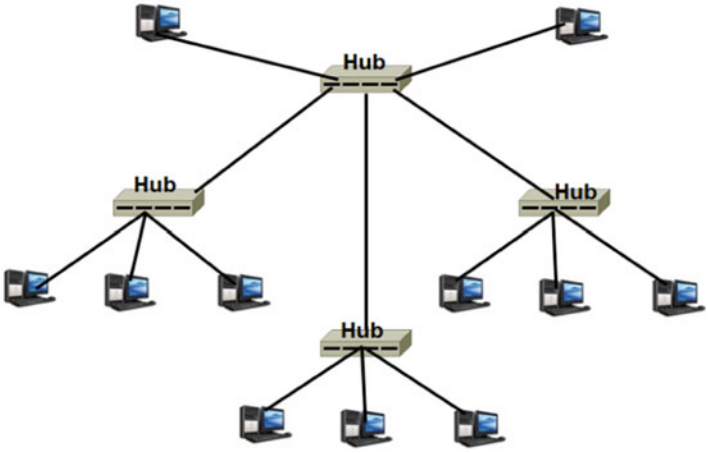


Fig. 2.7 Illustration of tree topology

mesh network has $\frac{1}{2}n(n+1)$ physical channels to link n nodes. To be able to handle these links, every node on the network must have $n-1$ input/output ports. An example of mesh topology is given in Fig. 2.6.

- *Tree topology*: The tree topology is a variation of the star topology. As in the star topology, nodes in a tree are linked to a central controller that controls the traffic to the network. However, not every node needs to be plugged into the centralized controller. The central hub of the network is an active hub that contains a repeater. It amplifies the signal to increase the distance the signal can travel. The secondary hub is either active or passive. A passive hub provides a simple physical connection between the attached nodes. An example of tree topology is given in Fig. 2.7.

2.1.5 *Types of Networks*

Based on size and geographical coverage, networks are classified as given in Fig. 2.8. The distances among the PCs that are connected as a network determine the type of network, viz., local area network (LAN), metropolitan area network (MAN), and wide area network (WAN). Each is described below:

- *Local area network (LAN)*: A LAN is the most popular and useful class of networks, designed to operate over a small physical area such as an office, a factory, an organization, or a group of buildings within a few kilometers of each other. A LAN is easy to design and troubleshoot. The exchange of information and sharing of resources between the PCs are straightforward within a LAN. Each PC in a LAN has an identifier, an address that uniquely identifies it as a host within the LAN. A packet sent by a host carries the address of the source as well as that of the destination. A LAN has several significant features such as (a) its components can use different technologies, (b) the size and coverage areas are usually small, (c) it can have different topologies such as star, ring, and bus within the LAN. With advances in technology, LAN speed is always increasing. It starts with 10 Mbps and can go up to 10 Gbps. An example of LAN is given in Fig. 2.9.
- *Metropolitan area network (MAN)*: A MAN is a bigger version of a LAN and normally uses similar technology. It is designed to extend over an entire city or a metropolitan area. It may connect a number of LANs into a larger network so that resources can be shared LAN to LAN as well as host to host as shown

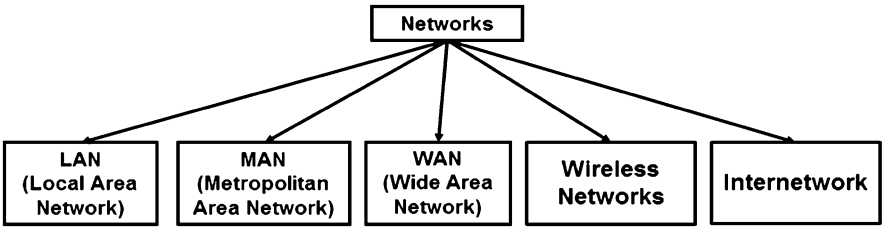


Fig. 2.8 Illustration of network categories

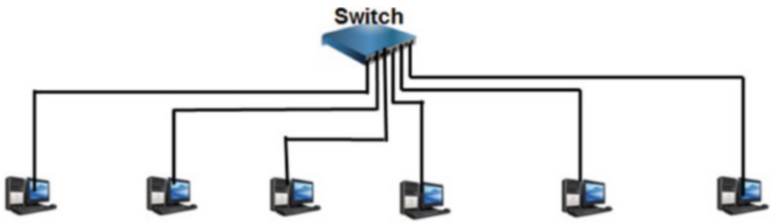


Fig. 2.9 LAN with switch

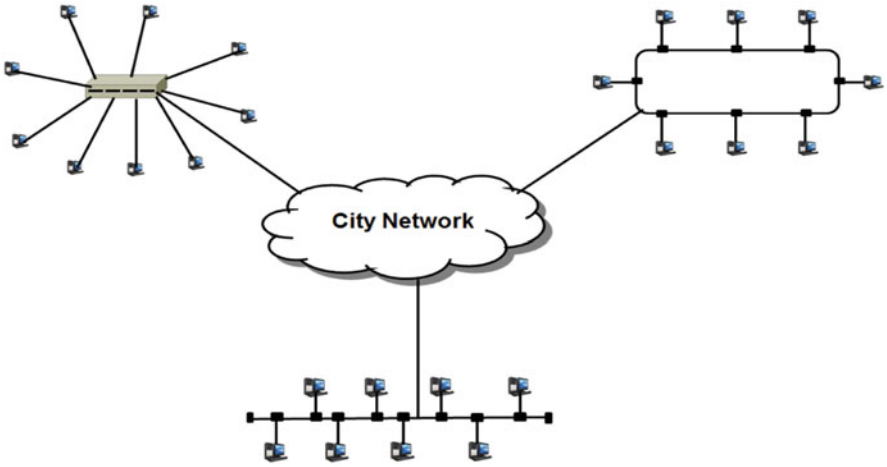


Fig. 2.10 Illustration of MAN

in Fig. 2.10. A MAN can cover approximately 30 to 100 kilometers, connecting multiple networks that are located at different locations in a city or town.

- *Wide area network (WAN)*: A WAN covers a wide geographical area that may span a town, a state, a country, or even the world. Usually a LAN interconnects different hosts but a WAN connects switches, routers, or modems. All hosts are connected by a communication subnet, where the subnet carries the messages from host to host. An example of wide area network is given in Fig. 2.11.
- *Wireless network*: A wireless network is convenient where wired connectivity fails or is difficult to construct practically. There is a growing segment of computer industry that requires wireless connectivity. Wireless networks have a wide range of applications in real-world products used every day such as in inside homes, cars, or planes, in mobile phone networks and military stations in remote areas, and many more. Wireless LAN is becoming very popular and an example of wireless network is shown in Fig. 2.11.
- *Internetwork*: The internetwork or the Internet is a collection of two or more networks located around the world, connected by gateways. Each gateway has a routing table containing information about the networks that are connected to the gateway. Several networks may be connected to one gateway. A common form of the Internet is a collection of LANs connected by a WAN. An Internet is a switched network where a switch connects at least two links together. The most common types of switched networks are (a) packet-switched and (b) circuit-switched. An example of an internetwork is shown in Fig. 2.11.

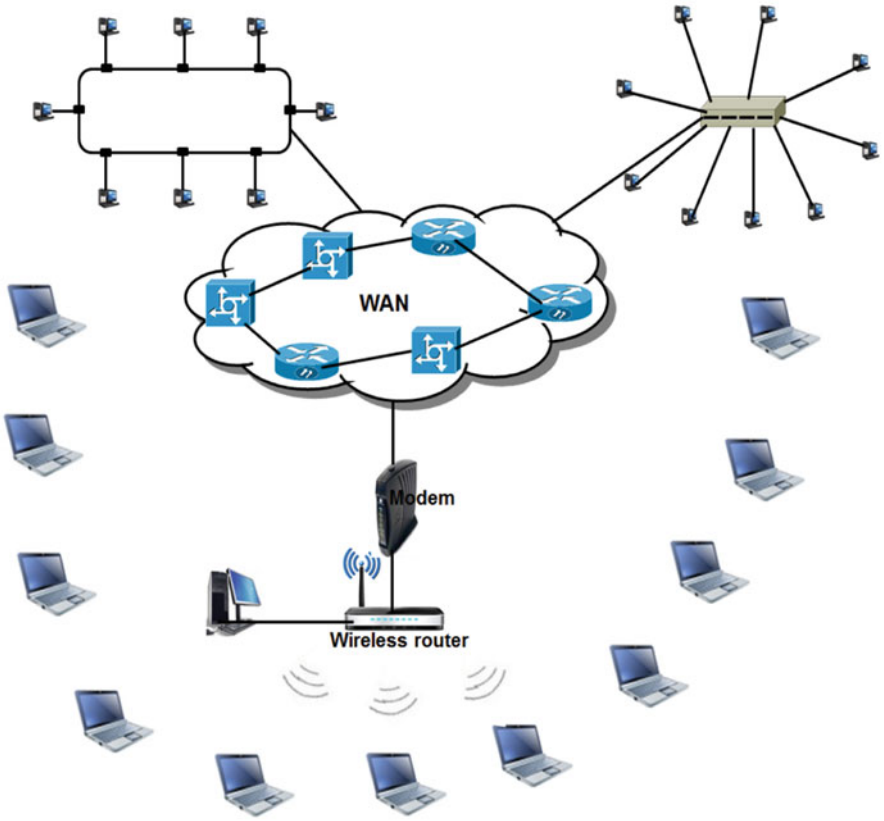


Fig. 2.11 Illustration of WAN, wireless network, and internetwork

2.1.6 Connection-Oriented and Connectionless Services

Two types of services are mainly offered to the network layers above them. They are (a) connection-oriented service and (b) connectionless service:

- *Connection-oriented service*: A connection-oriented service is similar to the one provided by the telephone system. Users of a connection-oriented service have to undertake a sequence of operations. They are: (i) establish a connection, (ii) use the connection, and (iii) release the connection. After establishing a connection, the sender and receiver can discuss and negotiate the parameters to be used such as maximum message size, quality of service, and other issues and then actually take part in the communication provided by the service.
- *Connectionless service*: A connectionless service is similar to the postal service. Each message carries full address of the destination. Each message is routed independently from source to destination through the system. It is possible that the order in which messages are sent and the order in which the messages are received between the same sender and same recipient are different.

Table 2.1 Service primitives

Name	Meaning
LISTEN	Wait for an incoming connection
CONNECT	Establish connection
RECEIVE	Wait for a message
SEND	Send the message
DISCONNECT	Terminate the connection

2.1.7 Service Primitives

The term primitive means an operation. A service is specified using a set of primitives. A user can execute these primitives to access the service. The primitives for connection-oriented services are different from those for connectionless services. The service primitives required to implement a client-server environment are given in Table 2.1.

2.1.8 Relationship Between Services and Protocols

Services and protocols are completely different concepts in computer networks:

- *Service*: It is defined as a set of operations that a network layer can provide to the layer above it. A service defines states and operations of a layer which is ready to provide the service but does not specify anything about the implementation of these operations.
- *Protocol*: This is a set of rules which governs the format and meaning of frames, packets, or messages that are exchanged between the peer entities. The entities use protocols to implement their services. A service is free to change the protocol used if the service states and operations can be ensured.

2.1.9 Reference Models

A network architecture or network reference model is a formal, logical structure that defines how network devices and software interact and function. It defines communication protocols, message formats, and standards required for interoperability. Network architectures are designed by standards organizations and manufacturers. The International Organization for Standardization (ISO) has designed the Open System Interconnection (OSI) architecture. In this chapter, we discuss the two popular reference models: (i) the OSI reference model and (ii) the TCP/IP reference model.

2.1.9.1 The OSI Reference Model

The users of a computer network may be spread over many physical locations across the globe. Therefore, to ensure nationwide and worldwide data communication, it is necessary to develop a system for communication that is compatible with each other. As a result, an international world-wide group has been formed to develop and maintain standards. Thus, standards fit into a framework developed by this organization called the International Organization for Standardization (ISO). This framework is a model for Open System Interconnection (OSI) and is usually referred to as the OSI reference model.

The most redeeming feature of the OSI reference model is that it is formally defined and it codifies the concept of layered network architectures. It uses well-defined operationally descriptive layers that describe what happens at each stage in data transmission. This layering concept is extremely important because networks are nontrivial systems. An outline of the OSI reference model is given in Fig. 2.12.

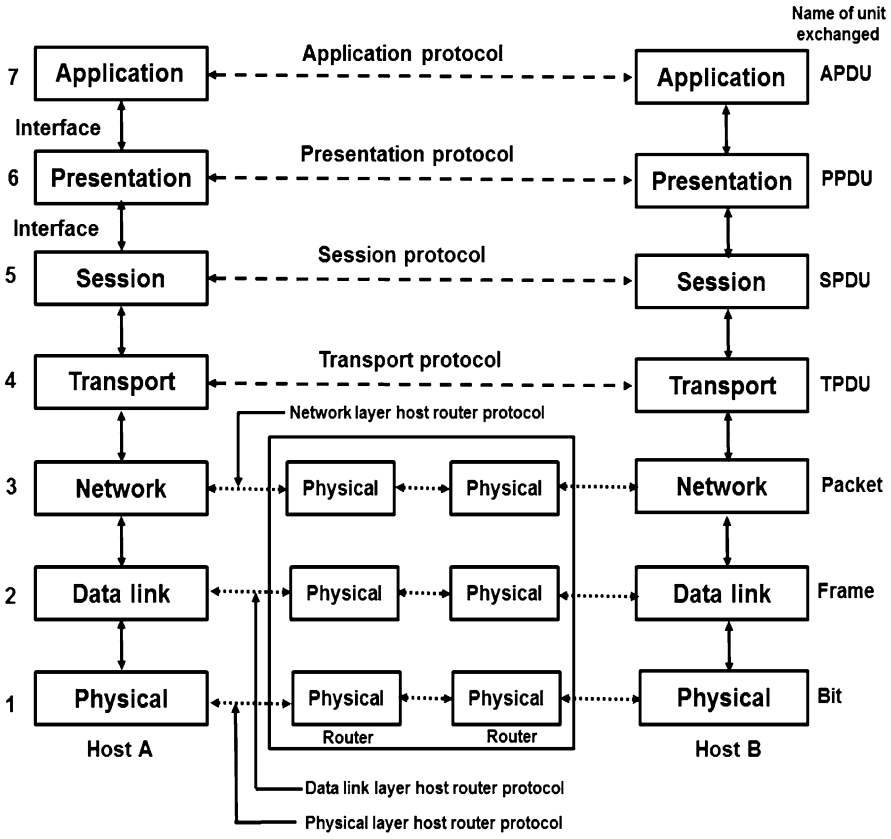


Fig. 2.12 The OSI reference model

Table 2.2 Functions of layers in OSI reference model

Level	Name of the layer	Functions
1	Physical	• Establish and detach connections, define voltage and data rates, convert data bits into electrical signal • Decide whether transmission is simplex, half-duplex, or full duplex
2	Data link	• Synchronize, detect error, and correct • Wait for acknowledgment for each transmitted frame
3	Network	• Route essential signals • Divide outgoing message into packets. • Act as network controller for routing data
4	Transport	• Decide whether transmission should be parallel or single path, and then multiplex, split or segment the data is required • Break data into smaller units for efficient handling
5	Session	• Manage synchronized conversation between two systems • Control logging on and off, user authentication, billing, and session management
6	Presentation	• Concerned with the syntax and semantics of the information transmitted • Known as translating layer
7	Application	• Retransfer files of information • Assist in LOGIN, check password, and so on

The OSI reference model is also known as the seven-layer model for data communication. The layers are physical, data link, network, transport, session, presentation, and application. Functions of the layers are described in Table 2.2.

2.1.9.2 The TCP/IP Reference Model

This model was developed for the ARPANET and it is used on the Internet. ARPANET was a research project on network interconnection sponsored by the US Department of Defense. It was a collaborative effort among many universities and government organizations that used leased telephone lines. Later, satellite and radio networks were added to it. Such inclusion could not be handled by the existing network protocols at that time. Therefore, a new architecture was needed and developed. It is called the TCP/IP reference model and it uses the Transmission Control Protocol (TCP) and the Internet Protocol (IP). When designing this model, certain goals were set to be achieved. These are given below:

(a) The protocol should have the ability to connect multiple networks together in a seamless way.
(b) The network should be able to survive even with the loss of subnet hardware with existing communication continuing to completion.

(c) It should be a flexible architecture that can deal successfully with the divergent requirements of various applications.

Since TCP/IP was initially developed for military use, it was robust to failures and flexible enough to be used by diverse networks. TCP/IP is the most popular and widely used protocol for interconnecting computers and it is the protocol of the Internet. This reference model has only four layers as given in Fig. 2.13. They are the internet layer, transport layer, application layer, and host-to-network layer. Each layer is described below:

(a) *Internet layer*: The main task of the *internet layer* is to allow a host to insert packets into any network and then ensure that they travel independently to the destination. This layer defines the packet format and protocol called the *Internet Protocol* (IP). The internet layer is entrusted with the delivery of IP packets to their destinations. So, routing of packets and congestion control are important issues in this layer. Hence, it is similar to the network layer in OSI reference model.
(b) *Transport layer*: The layer above the internet layer is known as the transport layer. It allows the entities on the source and destination machines to communicate with each other. Two end-to-end protocols have been defined in this layer, viz., Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). These protocols allow a byte stream transmitted from one machine to be delivered to the other machine without introducing errors. They also control flow of packets between the source and destination.
(c) *Application layer*: The TCP/IP model does not have session and presentation layers, though they are of little importance in most applications. The layer

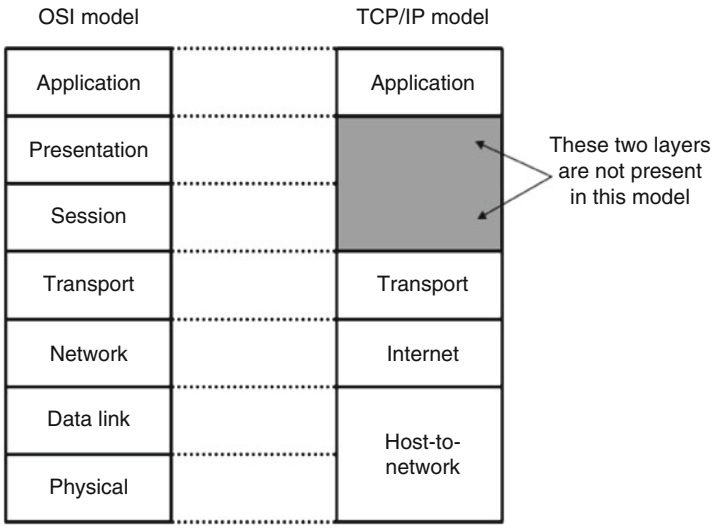


Fig. 2.13 The TCP/IP reference model

Table 2.3 Layer-specific protocols in both OSI and TCP/IP reference models

OSI model	TCP/IP model	Protocols
Application layer	Application layer	BitTorrent, BootP, DNS, DHCP, FTP, HTTP, HTTPS, NTP, SSH, SMTP, SNMP, Telnet, TFTP, XMPP, SIP, RDP, POP3, IMAP
Presentation layer		TLS
Session layer		NetBEUI, NetBIOS, NFS, NCP
Transport layer	Transport layer	SCTP, TCP, UDP, SSL
Network layer	Internet layer	ICMP, IGMP, IP4, IP6, IPX, OSPF
Data link layer	Host-to-network layer	Ethernet, Token Ring, FDDI, HDLC, IEEE 802.11, IEEE 802.16, VLAN, X.25, ARP, RARP
Physical layer		IEEE 1394, Etherloop, Bluetooth, 1000BASE-SX, ISDN, DSL, SONET/SDH, RS-449

above the transport layer is called the application layer. It contains all high-level protocols.

(d) *Host-to-network layer*: This is the lowest layer in the TCP/IP reference model. The host connects to the network using a protocol, so that it can send the IP packets over it. This protocol varies from host-to-host and network-to-network.

Based on the specification given above, several protocols have been developed to support communication between two hosts. Out of them, a few protocols in both OSI and TCP/IP reference models are given in Table 2.3. In the table, empty cells indicate that the two layers are absent in the TCP/IP reference model.

2.1.10 Protocols

To establish a connection between two hosts or networks for communication, it is necessary to follow a certain method or a set of rules, known as a protocol. It is likely to be effective if and only if a protocol can communicate with multiple devices or hosts or networks, a feat which is difficult in real time. Several protocols are used to communicate between devices, and each protocol has its own features, advantages, as well as disadvantages. The choice of protocols significantly affects network functioning and performance. This section explores some popular protocols that are commonly used during network traffic analysis.

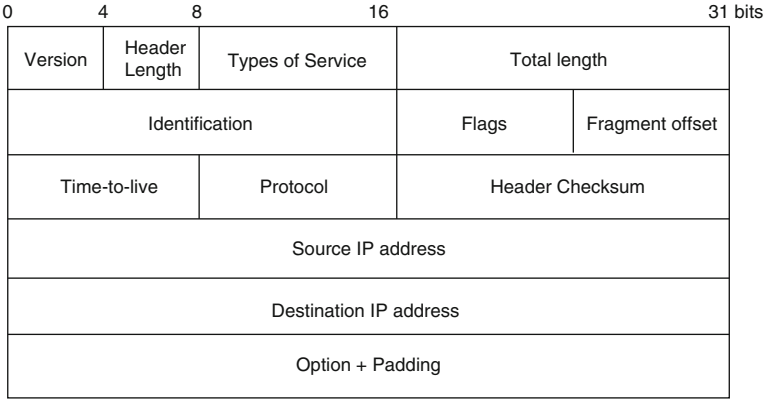


Fig. 2.14 IP header format

2.1.10.1 Internet Protocol (IP)

The Internet Protocol, defined in RFC 791, is used to transmit data from one device to another. It is a connectionless protocol, signifying that it cannot give guarantee for delivery. To ensure delivery, it performs fragmentation and reassembling of IP packets at the source and destination ends, respectively. Fragmentation is required because of the existence of a maximum transmission unit (MTU) size for transmission. Another major task of the IP is IP addressing. A detailed discussion on IP addressing can be found at [1]. The IP packet header format is given in Fig. 2.14.

2.1.10.2 Transmission Control Protocol (TCP)

The Transmission Control Protocol, defined in RFC 793, is a connection-oriented protocol that uses IP as the transport protocol. Before transmitting any data, this protocol establishes an acknowledged connection session. TCP provides several redeeming features such as reliability, flow control, sequencing, and error detection and correction. Once it finishes the data transmission, the connection is closed. The source host retransmits the data if it does not receive any acknowledgment of receipt within a time interval. This is known as the TCP connection *timeout*. The TCP header format is given in Fig. 2.15.

2.1.10.3 User Datagram Protocol (UDP)

User Datagram Protocol, defined in RFC 768, is a connectionless protocol that also uses IP as a transport protocol. UDP is a *fire and forget* protocol. UDP never checks whether the data is delivered to the destination or not, a task which is left to the upper-layer protocols. UDP has several features including low overhead and is economical in terms of bandwidth and processing effort. The UDP header format is given in Fig. 2.16.

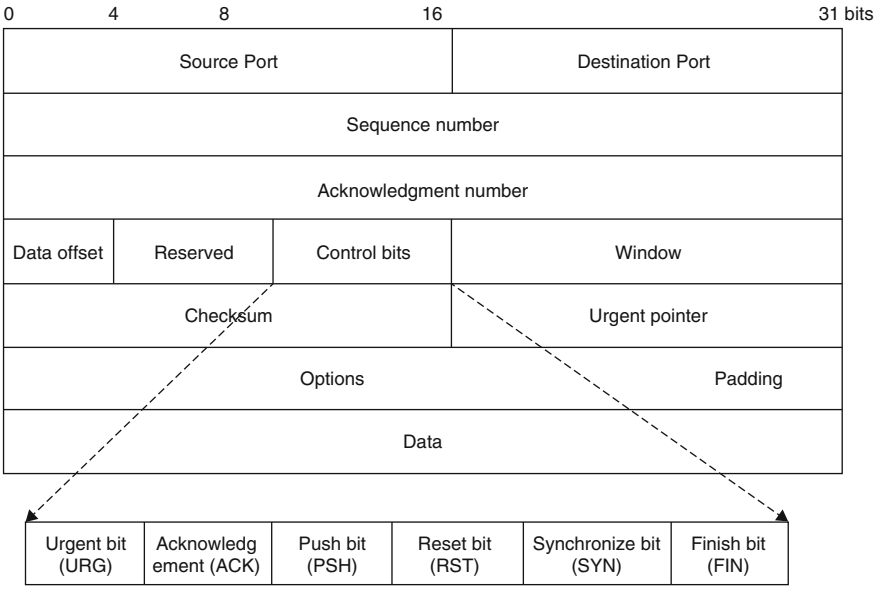


Fig. 2.15 TCP header format

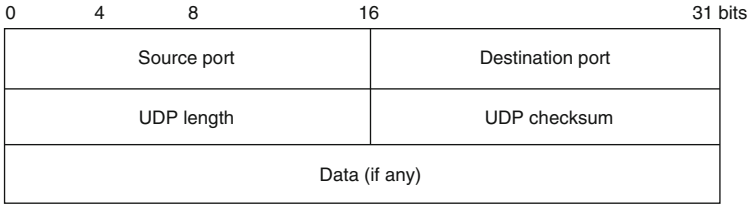


Fig. 2.16 UDP header format

2.1.10.4 Internet Control Message Protocol (ICMP)

ICMP, defined in RFC 792, is used to check error and perform as a reporting IP layer. ICMP has several functions. The most widespread use of the ICMP is found in the *ping* utility. Ping simply sends ICMP echo requests to a remote host and gets back reply through the ICMP echo reply message if the host is alive, i.e., it is working as expected. By using this simple process, ICMP can verify the protocol suite configuration at both source and destination ends.

ICMP can also facilitate the sending of return error messages such as *destination unreachable* when the destination cannot be contacted and *time exceeded* when time-to-live (TTL) value is exceeded. However, the ICMP is not limited to only these two utilities. It provides a *source quench* to control data transfer rate by requesting

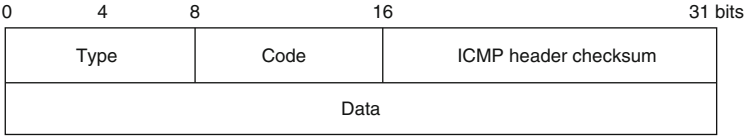


Fig. 2.17 ICMP header format

a message to be sent to the source end. So, it reduces packet drop significantly at the receiving end. The ICMP header format is given in Fig. 2.17.

2.1.10.5 Transport Layer Security (TLS)

To ensure privacy when client-server communication takes place for exchange of data, the TLS protocol is used to avoid any eavesdropping or intercepting. The TLS protocol is a successor of SSH. It has two parts, viz., (a) TLS record protocol and (b) TLS handshake protocol. The TLS record protocol uses a reliable transport protocol such as TCP and ensures that the connection made is private using data encryption. The TLS handshake protocol is used for authentication between client and server.

2.1.10.6 Address Resolution Protocol (ARP)

ARP, defined in RFC 826, is used to resolve an IP address to a MAC (Medium Access Control) address. Each host can be identified using its IP address. If any host wants to connect to another host, it immediately checks the ARP cache table to check if there is any entry for it. The ARP cache table contains three parameters, viz., IP address, MAC address, and entry type (static or dynamic). If there is an entry in the ARP cache table, the host broadcasts the message to the network. Only the target host that matches the IP sends reply directly to the source. A dynamic entry into the ARP cache table updates the entry automatically, but a static entry can be updated using an available command. A static entry is a permanent entry into the ARP cache table, but it can be removed using a command.

2.1.10.7 Reverse Address Resolution Protocol (RARP)

RARP, defined in RFC 903, performs similar operations like ARP but in the reverse order, i.e., resolves MAC addresses to IP addresses. The RARP is able to learn the IP address from a router or Domain Name System (DNS) server. It is necessary for reverse lookup of IP addresses in the DNS server.

2.1.10.8 Simple Mail Transfer Protocol (SMTP)

SMTP is specified in RFC 821. It describes how mail is sent to hosts. SMTP ensures and delivers error-free messages to the destination because it uses the TCP. It is not a sophisticated protocol. SMTP requires that the destination host is always available. A mail system spools the incoming mail messages and the user can read them later. The SMTP is used to send and receive mail, but POP3 and IMAP are used only for receiving mail.

2.1.10.9 Hypertext Transfer Protocol

HTTP is defined in RFC 2068. It is used to provide service in response to a client's request. HTTP clients (i.e., Web browsers) make requests using the HTTP format to the HTTP server (i.e., Web server). The clients make requests in the form of a special language, viz., the hypertext markup language (HTML). HTTP uses uniform resource locators (URLs) such as http://www.microsoft.com, to identify which page needs to be downloaded from the server. An extended and widely used version of the HTTP is HTTPS (Secure HTTP). HTTPS uses an approach called Secure Socket Layer (SSL) to encrypt the message when it sends it to the server and vice versa.

2.1.10.10 Telnet

Telnet is defined in RFC 854. It is a virtual terminal protocol that allows a session to be opened on a remote host. Once the connection is established, it allows the source agent (human or program) to execute commands on the remote host. Telnet can access multiuser systems such as mainframes and minicomputers on a terminal session. For several decades, telnet has been used in UNIX operating systems to provide services to users. Several managed network devices such as switches and routers also use telnet to communicate through the terminal. The main disadvantage of telnet is that it is not secure.

2.1.10.11 Simple Network Management Protocol (SNMP)

SNMP is a protocol that facilitates network management functions. However, it is not a network management system (NMS). It has a centrally managed component known as the SNMP *manager* that acts as a common communication point for all the SNMP-enabled devices on the network. For each device on the network that needs to be managed and monitored using SNMP, it is necessary to configure an SNMP *agent* with the SNMP manager's IP address. The SNMP manager communicates with the agents to retrieve information using a message known as *trap*. The function of the SNMP is illustrated in Fig. 2.18.

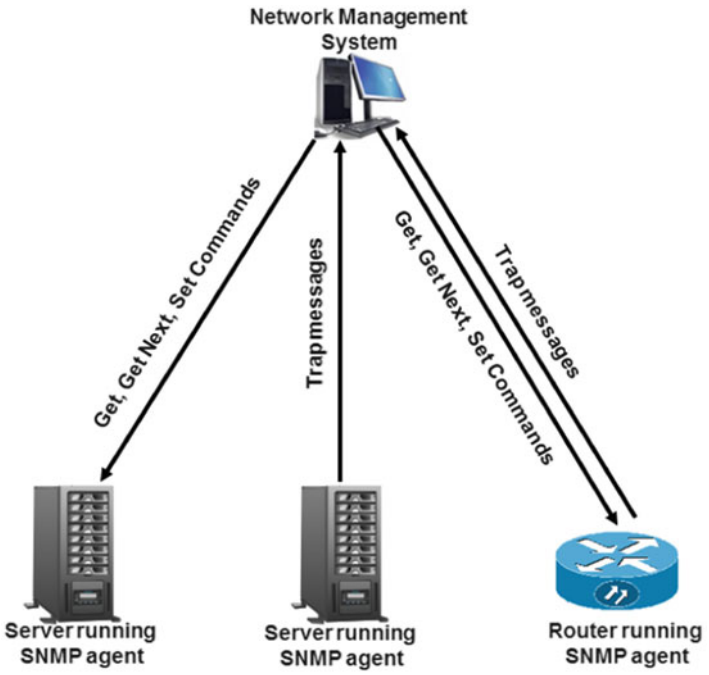


Fig. 2.18 How SNMP works

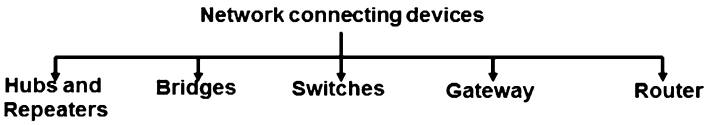


Fig. 2.19 Categories of network connecting devices

2.1.11 Network Connecting Devices

Devices that connect network components making communication possible can be classified as given in Fig. 2.19:

- *Hub*: It is a device that connects multiple PCs through transmission media. When a hub receives a request from a host, it transmits the data to the entire network. Each PC decides whether the broadcast data is for it or not. A hub works in the physical layer of the OSI reference model.
- *Repeater*: A repeater is a device similar to a hub, but it has additional features. It also works in the physical layer. Repeaters are used when amplification of the input signal is necessary. This amplification is required due to transmission of signals over long distances, when attenuation, delay, distortion, and noise lead to loss of data.

- *Bridge*: Bridges are used to connect two LANs. A bridge operates in the physical layer as well as in the data link layer. It regenerates a signal after it receives it. As a data link layer device, it can check the MAC addresses of the source and destination contained in a frame. There are mainly two kinds of bridges, viz., transparent bridges and routing bridges. A *transparent bridge* is a bridge in which the stations are not all aware of the existence of the bridge. A transparent bridge keeps a table of addresses in memory to determine where to send the data. The duties of a transparent bridge include filtering, forwarding, and blocking frames.

  In case of *routing bridges*, a sending station defines the bridges that must be visited by the frames. The addresses of these bridges are included in the frame. Hence, a bridge contains not only the source and destination addresses but also the bridge addresses. Source routing bridges are used to avoid a problem called *looping*. These bridges were designed for token ring LANs.
- *Switches*: A switch is a device that provides bridging functionality with greater efficiency. A switch acts as a multi-port bridge to connect devices in a LAN. The switch has a buffer to store packets for each link to which it is connected. When it receives a packet, it stores the packet in the buffer of the receiving link and finds the outgoing link. If the outgoing link is free, the switch sends the frame to that particular link. There are two types of switches, viz., store-and-forward and cut-through. A *store-and-forward* switch stores the frame in the input buffer until the whole packet arrives. In contrast, a *cut-through* switch forwards the packet to the outgoing buffer as soon as the destination address is found.

  Based on the working principles of switches, they are further classified into two categories: layer-2 switches and layer-3 switches. A *layer-2* switch works in physical and data link layers. A layer-2 switch is basically a bridge. It has a large number of ports and it is designed for better performance. A switch can connect many LANs due to the large number of ports it contains. One port can be allocated to one host in the LAN. Hence, each station has its separate identity. So, there is no competing traffic and as a result, there is no collision. A *layer-3* switch works in the network layer and is especially used for routing. To address two important limitations of layer-2 switches, viz., (i) broadcast overhead and (ii) lack of support for multiple links, layer-3 switches have come to the market with appropriate solutions. A layer-3 switch can logically break a large LAN into several sub-networks that are connected by routers. It improves the performance further. If it takes k inputs, then it generates same number of outputs. A three-bit number is used to decide the internal path over which the input is passed to output.
- *Routers*: A router is a device that connects two or more networks in which data flow from source to destination is based on information contained in its routing table. A routing table has information for all possible paths that an Internet datagram should follow during travel from source to destination. The software in a router includes an operating system and a routing protocol. Routers use physical and logical addressing to connect two or more logically separate networks. They build this connection by organizing a large network into logical network segments called *subnets*. Each subnet is given a logical address. Data

is grouped into packets. Each packet has a physical as well as a logical address. The network address allows routers to calculate the optimal path to a workstation or a computer. Routing of each packet is either static or dynamic. A *static routing* table maintains alternate routes to use if a router finds a particular router unavailable. A dynamic routing table provides more flexibility to respond to both error and congestion conditions. Intelligent routers are bundled with intrusion prevention solutions.

- *Gateways*: A gateway is a device that can interpret and translate among the different protocols that are used on different networks. It is a powerful and intelligent device for heterogeneous communication. A gateway is comprised of software, hardware, or a combination of both. A gateway operates across all seven layers of OSI model. A gateway can actually convert data from one network to another network even though their protocols are different. Some advantages of using gateways include the facts that (i) a gateway works as a proxy server for authentication and (ii) a gateway facilitates heterogeneous communication. Gateways are slow because they need intensive data conversion during communication.

2.1.12 Network Performance

Measuring network performance is important when working in a real environment. Many hosts and complex networks work together to exchange information between them within an internetwork. Frequently, this complexity may lead to poor performance. So, it is absolutely necessary to tune live network performance by applying different mechanisms. We discuss five different aspects for network performance. They are (a) measurement of network performance, (b) limitations of network performance, (c) performance-oriented system design, (d) fast processing of Transport Protocol Data Unit (TPDU), and (e) protocols for high-performance networks:

(a) *Measurement of network performance*: To maintain and improve network performance, the performance of the network must be measured periodically by the network administrator or network engineer. Round-Trip Time (RTT), packet loss, throughput, and availability are the basic measurements one can perform in a network. The basic steps are: (i) measure the relevant network parameters and performance, (ii) understand the bottlenecks, and (iii) change one parameter at a time and observe the performance.

(b) *Limitations of network performance*: Problems like temporary resource overloads can cause congestion in the network. If more traffic suddenly arrives at a router than the router can handle, congestion builds up and performance suffers. When there is a structural imbalance, like a high-speed line connected to a low-end PC, performance also suffers. Overloads can also be synchronously triggered. These packets are eventually retransmitted, causing delay, wasting

bandwidth, and resulting in reduced overall performance. Another tuning issue is setting time-outs correctly. For better quality, we have to keep in mind the bandwidth-delay product. It is the capacity in bits of the pipe from the sender to the receiver and back. The transmission time is another performance issue for time-crucial applications like audio and video.

(c) *Performance-oriented system design*: Measuring and tuning can improve network performance considerably, but they cannot be substitute for good design. So, we need to consider the system design, not just the network design, since software and operating system are often more important than the routers and interface boards. These are some observations that should impact on a good design such as: (i) keep in mind that CPU speed is more important than network speed, (ii) reduce packet count to reduce software overhead, (iii) minimize context switches because of their translation overhead, (iv) minimize copying of packets or datagrams, (v) increase bandwidth but not lower delay, (vi) avoid congestion because it is better than recovering from it, and (vii) avoid time-outs.

(d) *Fast processing of TPDU*: The key task to fast Transport Protocol Data Unit (TPDU) processing is to separate out the normal cases and handle them separately. On the receiver side, the connection record for an incoming TPDU has to be looked up. Once a sequence of special TPDUs is obtained to get into the established state, TPDU processing becomes straightforward until one side closes the connection.

(e) *Protocols for high-performance networks*: The major performance bottleneck in networks is that growth of communication speed is much higher than that of the processing speed. Therefore, when a packet travels over the network, the travel time may exceed the packet's lifetime, i.e., it may fail to reach its destination. The protocol designer for high-speed networks needs to take care of not only bandwidth optimization but also keep this in mind.

2.1.13 Network Traffic Management

Network traffic management aims to control network traffic by limiting bandwidth to certain applications, guaranteeing minimum bandwidth to others, and marking the traffic instances as high or low priority.

2.1.13.1 Steps in Traffic Management

Network traffic management requires a sequence of activities to accomplish appropriate control over the traffic. The usual steps for traffic management are given in Fig. 2.20.

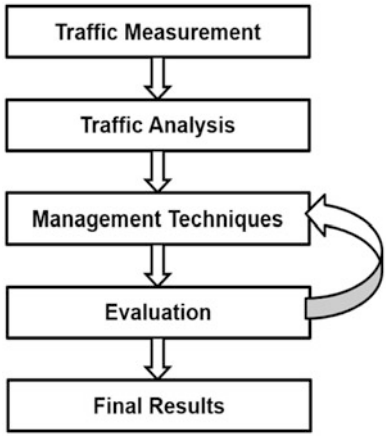


Fig. 2.20 Steps in network traffic management

2.1.13.2 Techniques for Traffic Measurement

Network traffic measurement is important to understand current network status. The main reasons to measure network traffic are given below:

- *Service monitoring*: Monitoring of services is necessary from time to time to ensure that the services to be provided by hosts in the network are working correctly.
- *Network planning*: Depending on demand for services and other communication, it needs to be decided when more capacity may be required to ensure smooth functioning.
- *Cost recovery*: Since cost is always an issue, we must maintain quantitative information on session and volume of traffic to provide the total cost to planners and administrators.
- *Research usage*: The important issues that must be considered for performance improvement need to be observed and measured to allow for changes or tune-up of the existing network.

There are two approaches to measure network traffic. They are (a) active measurement and (b) passive measurement:

- *Active measurement*: From the name itself, one can surmise that this involves direct measurement of the activities at runtime. For example, one can measure packet loss, delay, and throughput based on the active network activities.
- *Passive measurement*: This approach involves measuring network activities indirectly using software or hardware tools. It may use historical data to predict current traffic measurements. The main techniques to achieve this are (a) packet monitoring, (b) using router or switch statistics, and (c) analysis using router or server logs.

2.1.13.3 Types of Network Traffic

Based on the nature of the network itself, network traffic can be classified into two categories: (i) LAN traffic and (ii) WAN traffic:

- *LAN traffic*: LAN traffic is self-similar in nature. In other words, if we plot an hour's worth of traffic at several distinct time points, we likely to find almost similar patterns [36]. The traffic variation repeats after a regular time interval. It follows normal probability distribution.
- *WAN traffic*: WAN traffic pattern varies from time to time. For example, it may be (i) random traffic, (ii) Poisson traffic, or (iii) bursty traffic. *Random traffic* does not follow any fixed pattern. *Poisson traffic* shows trend similar to what is found on the Internet. The Poisson model estimates the probability of packets that should be present in the network after a given time if the arrival rate of the packets is specified. Attack traffic normally follows Poisson traffic's properties. The *bursty traffic* rate stays constant except for sudden bursts. The length of a burst may be short or long in nature. Bursty traffic may be attack traffic.

2.2 Network Traffic Anomalies

Anomalies are instances of data that do not conform to normal behavior. Instances may be called objects, points, events, vectors, or samples by various authors. Anomalies in networks can be defined as network events or operations that deviate from normal network behavior. They may occur due to reasons such as (i) malicious activities that misuse normal network services, (ii) overloading of networking infrastructure, (iii) malfunctioning of network devices, and (iv) compromises made when setting network parameters. Network anomalies are broadly categorized into two types [66]: (a) performance-related anomalies and (b) security-related anomalies. We discuss each of these with sources and causes.

2.3 Types of Anomalies

Anomalies in networks can be understood in the context of security and performance. In either case, some flaws or security holes remain knowingly or unknowingly and attackers find and exploit them. Anomalies can be classified into two major classes. They are (a) performance-related anomalies and (b) security-related anomalies. We discuss each below.

2.3.1 Performance-Related Anomalies

Anomalies may cause network or system failures or performance degradation. Typical examples of performance-related anomalies are file server failures, paging across the network, broadcast storms, babbling nodes, and transient congestion. A *broadcast storm* is caused when one or more computers broadcast a very high amount of traffic, consuming a large amount of network resources. In bus technology, two hosts may communicate unnecessarily and make the bus always busy. This is also an example of the *babbling nodes* problem. *Transient congestion* is caused by dropping of excess packets created by protocols like TCP packets, which overload the link bandwidth reducing transmission rates in the network. As a result, the network cannot transmit normal traffic. Performance-related anomalies may occur due to *vulnerabilities* in a network or a system. In a networked system, vulnerabilities are inherent weaknesses in the design, implementation and management of the system, rendering the systems susceptible to threats [39]. It is not possible to list all the sources of network-based system vulnerabilities. Many common sources of network vulnerabilities are pointed out below:

1. *Poor design*: Design flaws in a hardware or software system may lead to threats to the system. For example, a sendmail flaw in earlier versions of UNIX enabled hackers to gain privileged access to the system.
2. *Incorrect implementation*: Incorrect or erroneous configuration of the system, leading to vulnerabilities, may occur due to lack of experience, insufficient training, or sloppy work. For example, configuring a system that does not have restricted access privileges on system files may allow these files to be altered by unauthorized users.
3. *Poor security management*: Use of inadequate management procedures is another source of network vulnerabilities. For example, a lack of guarantee that security procedures are being followed and that no single person has total control of a system may lead to vulnerabilities.
4. *Vulnerability in Internet technology*: Internet technology has been and continues to be vulnerable. Every day, there are reports of all sorts of loopholes, weaknesses, and gaping holes in both software and hardware technologies. Such vulnerabilities have led to attacks such as the CodeRed worm and the Slammer worm.
5. *The nature of intruder activity*: Hacker technologies are developing faster than the rest of the technology and are flourishing. For example, $W32/Mydoom.n@MM!1812A23B5D92$ is a new malware threat.[1]
6. *The difficulty in fixing vulnerable systems*: It is often difficult to fix a vulnerable system within a stipulated time period. For example, there is concern about the ability of system administrators to cope with the number of patches issued

[1] http://www.mcafee.com/us/mcafee-labs.aspx

for system vulnerabilities. As the number of vulnerabilities rises, system and network administrators face a more difficult situation.

7. *Social engineering*: Social engineering involves a hacker's use of psychological tricks on legitimate users of a computer system. For example, the hacker may meet potential victims in a social networking environment and try to gain information such as username and password that one needs to enter the system. So, social engineering can be used to collect information that may be used to cause a potential threat.

Thus, there are many causes for traffic anomalies that provide attackers with opportunities to exploit network-based vulnerabilities. Many are difficult to fix quickly. As noted earlier, some examples of vulnerabilities are network configuration, malfunctioning hardware, flawed parameter setting, detailed logging and monitoring without protection, and communication and wireless vulnerabilities. In the next section, we discuss security-related network anomalies and possible sources and causes of network vulnerabilities. Security-related anomalies are our prime focus in this book.

2.3.2 Security-Related Anomalies

Security-related anomalies occur when the network traffic does not conform to the normal and expected behavior. Network attacks include denial of service (DoS), distributed DoS, probe, user to root, remote to local, and coordinated scan attacks. The main causes of network attacks or intrusions are malicious entities or agents, who hijack network bandwidth by flooding the network with unnecessary traffic, thus starving legitimate users. Malicious activities in a network are of various types such as point anomaly, contextual anomaly, and collective anomaly [15]. We describe these briefly below:

1. *Point anomaly*: When an instance of data is found to be anomalous with respect to the rest of data, it is known as point anomaly. For example, one instance of an individual typing in a wrong password is a point anomaly, which may or may not mean much in the overall context. A single instance of credit card fraud is also a point anomaly.
2. *Contextual anomaly*: A data instance which has been found anomalous in a specific context is known as a contextual anomaly. Context is induced by the structure of the dataset. Context in data and behavioral attributes of users can be used to establish contextual anomalies. For example, providing a malformed email address or uniform resource locator (URL) in some situations may simply be a minor nuisance but in other cases may lead to more serious problems.
3. *Collective anomalies*: A collection of related data instances found to be anomalous with respect to the entire dataset constitutes collective anomalies. In such a case, a collection of events is an anomaly, but the individual events may not be anomalies when they occur alone. For example, if there is someone who tries to

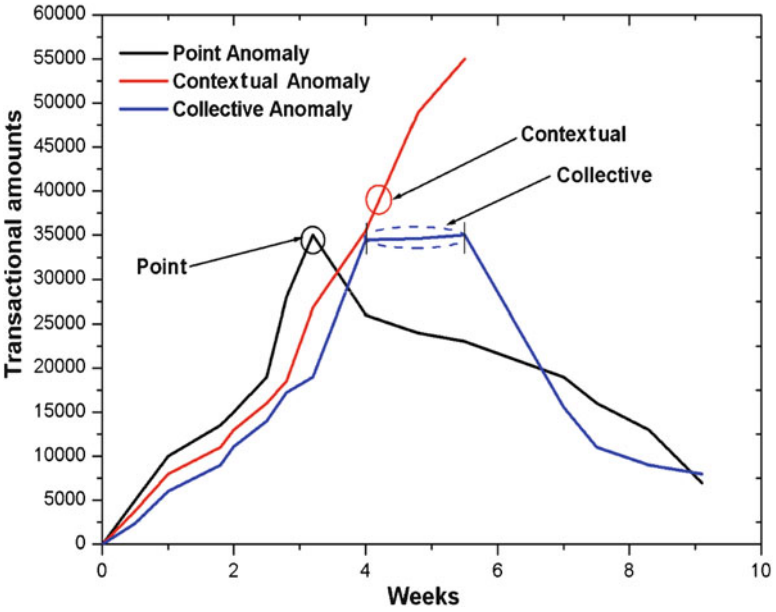


Fig. 2.21 Illustration of point, contextual, and collective anomalies

log in with the wrong password for a dozen accounts, it is likely to be a potential intruder, but typing the password wrong 1, 2, or 3 times for a single account may be simply a case forgotten password.

We illustrate each type of anomaly in terms of fraudulent credit card transactions given in Fig. 2.21. The anomalies occur while it deviates the usual pattern for credit card transactions, i.e., the usual transactional amount *INR* 30,000.

2.4 Network Attacks

A network attack exploits one or more vulnerabilities in a computer or network and attempts to break into or compromise the security of the system. One who performs or attempts an attack or intrusion into a system is an attacker or intruder. Anderson [3] classifies attackers or intruders into two types: external and internal. *External intruders* are unauthorized users of the systems or machines they attack, whereas *internal intruders* have permission to access the system but do not have privileges to access the system as root or superuser. Internal intruders are further divided into masquerade intruders and clandestine intruders. A *masquerade intruder* logs in as another user with legitimate access to sensitive data, whereas a *clandestine intruder*, the most dangerous, has the power to turn off audit control for themselves.

An attack or intrusion is perpetrated by an inside or outside intruder of a system to gain unauthorized entry and control of the security mechanism of the system.

2.5 Attack Taxonomy

An attack or intrusion is usually a sequence of operations that starts with the goal of placing a backdoor for exploiting in a network or a computer system and to eventually cause damage by stealing private information or by disrupting service [41]. Several network attack taxonomies are available in the literature [28, 32, 51, 69]. The taxonomy of *intrusions* or *attacks* [26, 37] in computer systems that we adopt is summarized in Table 2.4.

Network attacks are also classified as *active* or *passive*. Passive attacks are designed to monitor and record traffic on the network. They are usually employed to gather information that can be used later in active attacks. They are very difficult to detect, because there is usually no overt activity that can be monitored or detected. Examples of passive attacks are packet sniffing or traffic analysis. In contrast, active attacks employ more overt actions on the network or system. They can be much more devastating to a network.

2.6 Motivations of Attackers

The precursors to an attack are usually a series of events used to trigger an attack. A network attacker executes a series of steps to achieve the desired goal. The order and duration of these steps are dependent on factors such as the attacker's skill level, the type of vulnerability to exploit, prior knowledge, and the location of the attacker. An attacker generally follows the steps shown in Fig. 2.22 to launch an attack.

Performing reconnaissance means that the attacker uses certain techniques to gather information about the strengths and positioning of enemy forces using *scanning* and enumeration of *services*. Once the vulnerabilities are identified, the attacker attempts to exploit them when *launching the attack*. The attacker can gain access as a regular user and then as root user to a system. Finally, to *place backdoors* on the system for further exploits, the attacker *maintains* access and cleans up any evidence left in the system.

2.7 Traffic Monitoring and Analysis

Network traffic monitoring is usually performed at the intra-domain level in every large-scale autonomous system (AS) because the network topology is completely known and the AS is under the control of a single network operator, who can

Table 2.4 Taxonomy of computer attacks: characteristics and examples

Attack name	Characteristics	Examples
Virus	(i) A self-replicating program that infects the system without any knowledge or permission from the user. (ii) Increases the infection rate of a network file system if the system is accessed by another computer	Trivial.88.D, Polyboot.B, Tuareg
Worm	(i) A self-replicating program that propagates through network services on computer systems without user intervention. (ii) Can highly harm network by consuming network bandwidth	SQL Slammer, Mydoom, CodeRed Nimda
Trojan	(i) A malicious program that cannot replicate itself but can cause serious security problems in the computer system. (ii) Appears as a useful program but in reality it has secret code that can create a backdoor to the system, allowing it to do anything on the system easily, and can be called as the hacker gets control on the system without user permission	Mail Bomb, phishing attack
Denial of service (DoS)	(i) Attempts to block access to system or network resources. (ii) The loss of service is the inability of a particular network or a host service such as email to function. (iii) It is implemented by either forcing the targeted computer(s) to reset or consuming resources. (iv) Legitimate users can no longer communicate adequately due to nonavailability of service or because of obstructed communication media	Buffer overflow, ping of death (PoD), TCP SYN, smurf, teardrop
General network attack	(i) Any process used to maliciously attempt to compromise the security of the network ranging from the data link layer to the application layer by various means such as manipulation of network protocols. (ii) Illegally using user accounts and privileges, performing actions that consume large amount of network resources and bandwidth, and performing actions that prevent legitimate authorized users from accessing network services and resources	Packet injection, SYN flood
Physical attack	Attempts to damage the physical components of networks or computers	Cold boot, evil maid
Password attack	Aims to gain a password within a short period of time and is usually indicated by a series of log-in failures	Dictionary attack, SQL injection attack
Information gathering attack	Gathers information or finds known vulnerabilities by scanning or probing computers or networks	SYS scan, FIN scan, XMAS scan
User to root (U2R) attack	(i) It is able to exploit vulnerabilities to gain privileges of superuser of the system while starting as a normal user on the system. (ii) Vulnerabilities include sniffing passwords, dictionary attack, or social engineering	Rootkit, loadmodule, perl

(continued)

Table 2.4 (continued)

Attack name	Characteristics	Examples
Remote to local (R2L) attack	(i) Ability to send packets to a remote system over a network without having any account on that system, gain access either as a user or as a root to the system and do harmful operations. (ii) Ability to perform attack against public services (such as HTTP and FTP) or during the connection for protected services (such as POP and IMAP)	Warezclient, warezmaster, imap, ftp_write, multihop, phf, spy
Probe	(i) Scans the networks to identify valid IP addresses and to collect information about host (e.g., what services they offer, operating system used). (ii) Provides information to an attacker with the list of potential vulnerabilities that can later be used to launch an attack against selected systems and services	IPsweep, portsweep

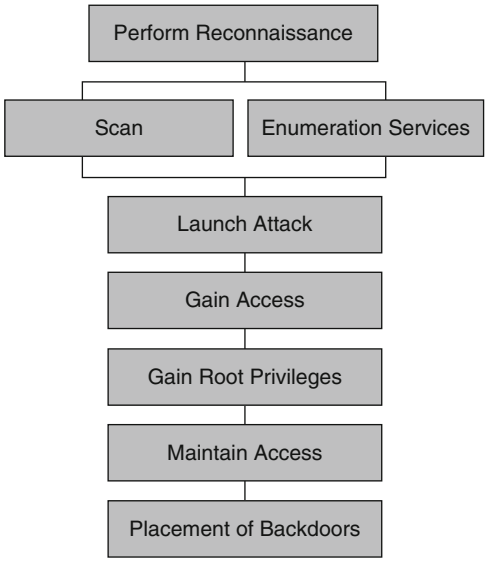


Fig. 2.22 Steps in performing an attack

therefore manipulate his network and traffic without restrictions [67]. Network administrators usually deploy the monitoring systems in the intra-domain Internet to capture, analyze, and decide whether an instance is normal or anomalous. A view of a deployed monitoring system with DMZ[2] is shown in Fig. 2.23.

[2]A demilitarized zone is a network segment located between a secure local network and insecure external networks (Internet). A DMZ usually contains servers that provide services to users on the external network, such as Web, mail, and DNS servers. These servers must be hardened systems. Two firewalls are typically installed to form the DMZ.

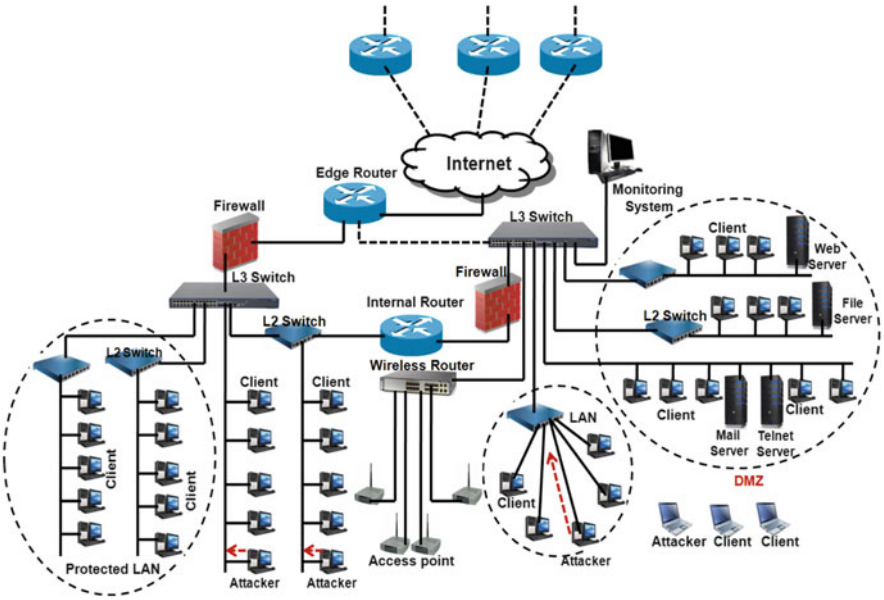


Fig. 2.23 A typical view of monitoring system deployment with DMZ

2.8 Anomaly Detection and ANIDSs

Network anomaly detection is crucial for securing a network or a host. It allows for timely mitigation when anomalous traffic instances appear. Anomalies may be caused due to many reasons as discussed above. Specifically, security-related anomalies occur due to malicious activity (e.g., scanning, denial of service, and probe) initiated by the attackers or intruders at different times. They are usually due to misconfiguration of a network or a host and may have serious consequences. A network anomaly detection system captures and analyzes traffic and reports alarms when an anomaly is detected and updates profiles of normal as well as attack instances.

An anomaly-based network intrusion detection system (ANIDS) is a system for detecting network anomalies by monitoring network traffic and classifying them as either normal or anomalous. The classification may be based on heuristics or may look for patterns or signatures, attempting to detect anomalies that are not found in normal system operation. An ANIDS should ideally be able to detect known as well as unknown attacks without any prior knowledge.

2.9 Classification of ANIDSs

Intrusion detection has been studied for at least 20 years. Intrusions can be detected because an intruder's behavior is noticeably different from that of a legitimate user. In addition, many unauthorized actions are inherently detectable [49]. ANIDSs are deployed as a second line of defense along with other preventive security mechanisms such as user authentication and access control. ANIDSs are classified into two types based on location of deployment in real time.

A *host-based IDS* (HIDS) monitors and analyzes the internals of a computing system rather than its external interfaces. It monitors all or parts of the dynamic behavior and the states of a computer system [68]. A HIDS may detect internal activity such as which program accesses what resources and makes attempts at illegitimate access. An example is a word processor that suddenly and inexplicably starts modifying the system password database. Similarly, a HIDS might look at the state of a system and other stored information whether it is in RAM or in the file system or in log files or elsewhere. One can think of a HIDS as an agent that monitors whether anything or anyone internal or external has circumvented the security policy that the operating system tries to enforce.

A *network-based IDS* (NIDS) detects intrusions in network traffic. Intrusions typically occur as anomalous patterns. These anomalies are primarily caused by attacks launched by outside attackers who want to gain unauthorized access to computers on the network to steal information or to disrupt the network. Some techniques model the network traffic in a sequential fashion and detect anomalous sub-sequences [68]. In a typical setting, a network is connected to the rest of the world through the Internet. The NIDS reads all incoming packets or flows, trying to find suspicious patterns. For example, if a large number of TCP connection requests to a large number of different ports are observed within a short time, one could assume that there is someone committing a 'port scan' at some of the computer(s) in the network. Various kinds of port scans and tools used to launch them are discussed in detail in [9]. Port scans mostly try to detect incoming shell codes in the same manner that an ordinary intrusion detection system does. In addition to inspecting the incoming network traffic, a NIDS also provides valuable information about intrusion from outgoing as well as local traffic. Some attacks might even be staged from inside of a monitored network or network segment and, therefore, not regarded as incoming traffic at all. The data available for intrusion detection systems can be at different levels of granularity, e.g., packet level traces or IPFIX records. The data is high dimensional, typically, with a mix of categorical and continuous attributes.

Misuse-based intrusion detection normally searches for known intrusive patterns but anomaly-based intrusion detection tries to identify unusual patterns. Today, researchers mostly concentrate on anomaly-based network intrusion detection because it can detect known as well as unknown attacks. In practice, this usually assumes that there are programs that can detect intrusion type that are known a priori and unknown ones are detected using anomaly detection.

There are several reasons that make intrusion detection a necessary part of the entire defense system. First, many traditional systems and applications were

developed without security in mind. Such systems and applications were designed many years ago or in a laboratory to work in an environment, where security was never a major issue. However, the same systems and applications when deployed in the current network scenario become major security headaches. For example, a system may be perfectly secure when it is isolated but becomes vulnerable when it is connected to the Internet. Intrusion detection provides a way to identify and thus allow response to attacks against these systems. Second, due to limitations of information security and software engineering practices, computer systems and applications may have design flaws or bugs that could be used by an intruder to attack systems or applications. As a result, certain preventive mechanisms (e.g., firewalls) may not be as effective as expected. Intrusion detection techniques are classified into three types based on the detection mechanism [15, 50, 62]. This classification scheme is described below:

1. *Misuse-based*: This type of detection is based on a set of rules or signatures for known attacks and can detect all known attacks based on reference data. How to write a signature that encompasses all possible variations of a pertinent attack is a challenging task.
2. *Anomaly-based*: The principal assumption is that all intrusive activities are necessarily anomalous. Such a method builds a *normal activity profile* and checks whether the system state varies from the established profile by a statistically significant amount to report intrusion attempts. Anomalous activities that are not intrusive may also be flagged as intrusive. These are false positives. One should select threshold levels so that the above two situations do not cause the system to malfunction considerably. Anomaly-based intrusion detection is computationally expensive because of overhead and the need to update several system profile matrices.
3. *Hybrid*: This detection mechanism exploits the benefits of both misuse and anomaly-based detection techniques. It also attempts to detect known as well as unknown attacks.

In addition to the above, an anomaly detection system works in any of four modes, viz., (i) supervised, (ii) semi-supervised, (iii) unsupervised, and (iv) hybrid based on the availability of labeled data.

2.9.1 Supervised ANIDS

A supervised ANIDS detects network anomalies using prior knowledge. It builds a predictive model for both normal and anomalous classes and compares any new instance with the predictive model to determine which class it belongs to. Thus, to provide an appropriate solution in network anomaly detection, we need the concept of normal behavior of the network traffic. An event or an object is detected as anomalous if its degree of deviation with respect to the profile or behavior of the system, specified by the normality model, is high enough. We define a supervised system as follows.

Definition 2.1 Let us consider an anomaly detection system I that uses a supervised approach. It can be thought of as a pair $I = (M, D)$, where M is the model of normal behavior of the system and D is a proximity measure that allows one to compute, given an activity record, the degree of deviation that such activities have from the model M. Thus, each system has two main modules: (i) a modeling module and (ii) a detection module. One trains the system for both normal and attack classes to obtain the model M. The obtained model is subsequently used by the detection module to evaluate new events or objects or traffic as normal or anomalous or outliers. In particular, the modeling module needs to be adaptive to cope with the dynamic scenarios.

A generic architecture for a supervised ANIDS is given in Fig. 2.24. A brief description of each component of the above system is given below:

(a) ***Traffic capturing***: Traffic capturing is an important module in any NIDS. In this module, live network traffic is captured using the libpcap [34] library, an open source C library offering an interface for capturing link-layer frames over a wide range of system architectures. It provides a high-level common application programming interface (API) to the different packet capture frameworks of various operating systems. The resulting abstraction layer allows programmers to rapidly develop highly portable applications.

Libpcap defines a standard format for files in which captured frames are stored. This format also known as the *tcpdump* format is currently a de facto standard widely used in public network traffic archives. Modern kernel-level capture

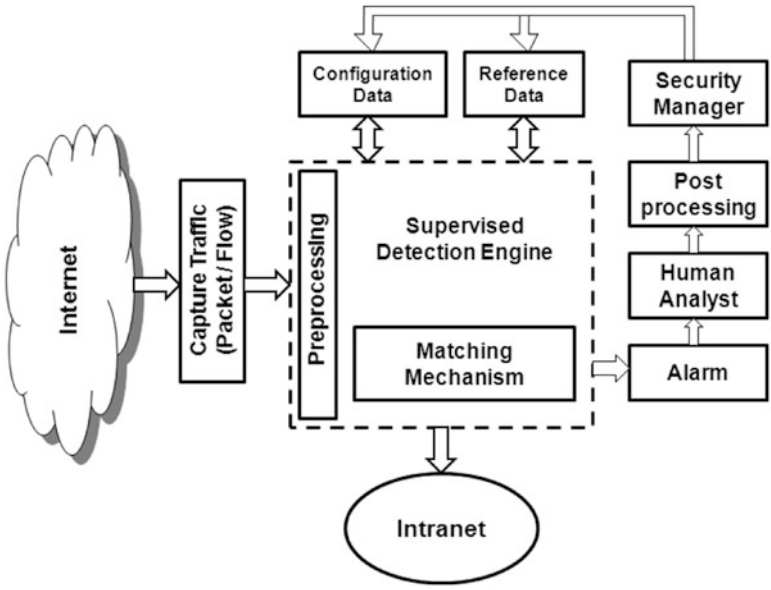


Fig. 2.24 A generic architecture of supervised ANIDS

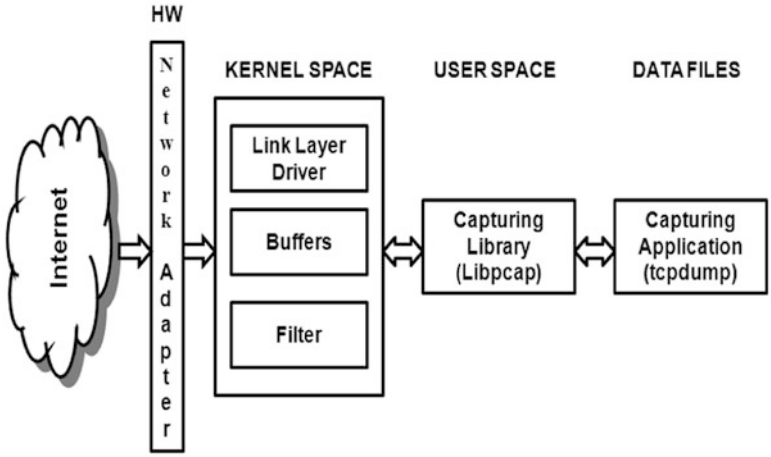


Fig. 2.25 Hierarchy of network traffic capturing components

frameworks on UNIX operating systems are mostly based on BSD or Berkeley Packet Filter (BPF) [19, 47]. BPF is a software device that taps network interfaces, copying packets into kernel buffers and filtering out unwanted packets directly in interrupt context. Definition of packets to be filtered can be written in a simple human readable format using Boolean operators and can be compiled into pseudo-code and passed to the BPF device driver by a system call. The pseudo-code is interpreted by the BPF pseudo-Machine, a lightweight high-performance state machine specifically designed for packet filtering. Libpcap also allows a programmer to write applications that transparently support a rich set of constructs to build detailed filtering expressions for most network protocols. Libpcap calls can use these Boolean expressions, which can read directly from the user's command line, compile into pseudo-code, and pass to Berkeley Packet Filter. Libpcap and BPF interact to allow network packet data to traverse several layers to finally be processed and transformed into capture files (i.e., tcpdump format) or to samples for statistical analysis.

The raw network traffic is captured at both packet and flow levels. Packet level traffic can be captured using popular tools such as Gulp (Lossless Gigabit Remote Packet Capture With Linux)[3] and Wireshark[4] and then preprocessed before sending to the detection engine. In addition, flow-level traffic can be captured using some other tools, e.g., NFDUMP,[5] NFSEN,[6] and ntop. The hierarchy of network traffic capturing components is given in Fig. 2.25.

[3] http://staff.washington.edu/corey/gulp/

[4] http://www.wireshark.org/

[5] http://nfdump.sourceforge.net/

[6] http://nfsen.sourceforge.net/

(b) ***Preprocessor***: In order to evaluate an IDS, an unbiased intrusion dataset in a standard format is required. Generally, a captured live packet contains a lot of raw data, some of which may not be relevant in the context of an IDS. Therefore, filtration of irrelevant parameters during capture and extraction of features from the filtered data are important preprocessing functions of an IDS. In addition to these, data type conversion, normalization, and discretization are also useful functions of this module depending on the anomaly detection mechanism used in the IDS:

1. *Feature extraction*: Feature extraction from raw data is an important step for anomaly-based network intrusion detection. The evaluation of any intrusion detection algorithm on real-time network data is difficult, mainly due to the high cost of obtaining proper labeling of network connections. The extracted features are of four types [37, 38] as follows:
   - *Basic features*: These can be derived from packet headers without inspecting the payload. The protocol type, services used, flag, source bytes, and destination bytes are examples of basic features.
   - *Content-based features*: Domain knowledge is used to assess the payload of the original Transmission Control Protocol (TCP) packets. An example of this type of features is the number of failed log-in attempts.
   - *Time-based features*: These features are estimated by capturing properties that hold over a t-second time window. One example of such a feature is the number of connections to the same host during the t-second time interval.
   - *Connection-based features*: These features are computed over a historical window estimated over the last n packets. An example of such a feature is the number of packets flowing from source to destination within a specified time period.
2. *Data type conversion*: Both features and raw data may include numeric as well as categorical data. For example, the protocol feature takes values such as *tcp*, *icmp*, *telnet*, and *udp*. Therefore, to apply a clustering technique based on a proximity measure that works either with numeric or categorical data and not both to detect network anomalies, it may be necessary to convert the data.
3. *Normalization*: In an intrusion dataset, all parameters or field values may not be of equal importance, or their value ranges may be very different. In such cases, normalization is considered useful before applying an anomaly detection mechanism.
4. *Discretization*: The network intrusion data contains continuous valued attributes such as the number of packets, the number of bytes, and the duration of each connection. These attributes may need to be transformed into binary features or range-based features before any standard association mining algorithms can be applied. The transformation can be performed using a variety of supervised and unsupervised discretization techniques. For example, using the output scores of the anomaly detector as its ground truth, MINDS (Minnesota INtrusion Detection System) [23] employs a supervised binning strategy to discretize attributes. Initially, all distinct values of continuous attributes are put into one bin. The

worst bin in terms of purity is selected for partitioning until the desired number of bins is reached. The discretization of numeric attributes contributes to the comprehension of the final results.

These traffic features are designed to assess attacks, which span intervals longer than 2 s. It is well-known that features constructed from the data content of the connections are more important when detecting R2L (remote to local) and U2R (user to root) attack types in the KDD99 intrusion dataset [38]. The time-based and connection-based features are more important for detection of denial of service (DoS) and probing attack types [42].

(c) ***Anomaly detection engine***: This is the heart of any network anomaly detection system. It attempts to detect the occurrence of any intrusion either online or offline. In general, any network traffic data needs preprocessing before it is sent to the detection engine. If the attack belongs to a known type, it can be detected using a misuse detection approach. Unknown attacks can be detected with the anomaly-based approach using an appropriate matching mechanism or a classifier. The following are some important requirements that a matching mechanism must satisfy:

- Matching determines whether the new instance belongs to a known class defined by a high-dimensional profile or not. Matching may be inexact. The membership of a test instance to a given predefined class represented by its profile depends on (i) the proximity computed between the profile and the new test instance using a relevant subspace of features and (ii) a user-defined proximity threshold. Thus, the selection of an appropriate proximity measure and an appropriate threshold is crucial here.
- Matching must be fast.
- Effective organization of the profiles may facilitate faster search during matching.

(d) ***Alarm***: This module is responsible for generation of alarm based on the indication received from the anomaly detection engine. In addition to indicating the occurrence of an attack, alarms are useful for post-diagnosis or alarm correlation analysis [59] of the performance of the detection system. Alarms should indicate (i) the causes that raised the alarm, (ii) the source IP/Port address and target IP/Port address associated with the attack, and (iii) any background information to justify why it is a putative alarm.

(e) ***Human analyst***: A human analyst is responsible for analysis and interpretation and for taking necessary action based on the alarm information provided by the detection engine. The analyst also takes necessary steps to diagnose the alarm information as a post-processing activity to support reference or profile updation with the help of security manager.

(f) ***Post-processing***: This is an important module in a NIDS. This module processes the generated alarms for diagnosis of actual attacks. Appropriate post-processing activities can help reduce the false positive rate significantly.

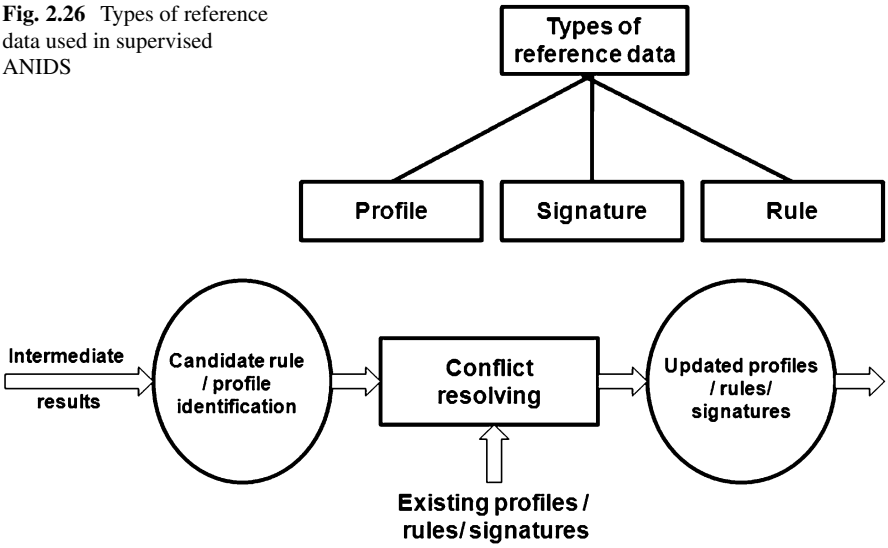


Fig. 2.26 Types of reference data used in supervised ANIDS

Fig. 2.27 Steps in updation of configuration data in ANIDS

(g) ***Security manager***: Stored intrusion signatures are updated by the security manager (SM) as and when new intrusions become known. The analysis of novel intrusions is a highly complex task. The security manager has multifaceted roles to play such as (i) analysis of alarm data, (ii) recognition novel intrusion(s), and (iii) updation of the signature or profile base.

(h) ***Reference data***: The reference data stores information about signatures or profiles of known intrusions or normal behavior. Reference data must be stored in an efficient manner. Possible types of reference data used in the generic architecture of a NIDS are shown in Fig. 2.26. In the case of ANIDS, it is mostly profiles. The processing elements update the profiles as new knowledge about the observed behavior becomes available. These updates are performed in a batch-oriented fashion by resolving conflicts, if they arise.

Intermediate results such as partially created intrusion signatures are stored as *configuration data*. The space needed to store such information is usually quite large. The main steps for updation of configuration data are given in Fig. 2.27. Intermediate results need to be integrated with existing knowledge to produce consistent, up-to-date results.

There are two major issues that arise in supervised anomaly detection. First, the anomalous instances are far fewer in number compared to normal instances in the training data. Issues that arise due to unbalanced class distributions have been addressed in the data mining literature [35]. Second, obtaining accurate and representative labels, especially for the anomaly class, is usually challenging. A number of proposed techniques inject artificial anomalies in a normal dataset to

obtain a larger labeled training dataset [65]. Other than these two issues, the supervised anomaly detection problem is similar to building predictive models.

2.9.2 Semi-supervised ANIDS

A semi-supervised ANIDS trains using labeled instances only for the normal class [75]. Since they do not require labels for the anomaly class, it is more readily used compared to supervised approaches. We define a semi-supervised system as follows.

Definition 2.2 Let I be a semi-supervised anomaly-based detection system. It can be thought of as a pair $I = (M, D)$, where M is the model of normal behavior of the system and D is a proximity measure that allows one to compute, given an activity record, the degree of deviation that such activities have from the model M. As discussed in the context of supervised ANIDS, each system has primarily two modules. The modeling module trains to get the normality model M and detect new traffic as normal or anomalous.

For example, in spacecraft fault detection [24], an anomaly scenario may signify an accident, which is not easy to model. The typical approach used in such techniques is to build a model for the class corresponding to normal behavior and use the model to identify anomalies in the test data. However, semi-supervised learning uses normal data during training and the rest of the approach is the same as supervised approach.

2.9.3 Unsupervised ANIDS

An unsupervised ANIDS can be used for novel intrusion detection without prior knowledge using purely normal data. Unsupervised network anomaly detection works well due to two major reasons: (i) nonavailability of labeled or purely normal data and (ii) the expense of manual classification of a large volume of network traffic. When collecting normal traffic data, it is extremely difficult to guarantee that there is no anomalous instance. Clustering is a widely used method for unsupervised anomaly-based intrusion detection [12, 13, 29, 56, 74]. From classical data mining, we know that clustering is a method of grouping of objects based on similarity among the objects. The similarity within a cluster is high, whereas dissimilarity among clusters is high. Clustering is a method of unsupervised exploration of the data that is performed on unlabeled data [72]. Unsupervised anomaly detection clusters test data into groups of similar instances which may be either normal or anomalous. We define an unsupervised system as follows.

Definition 2.3 Let I be an unsupervised anomaly-based detection system. It can be thought of as a pair $I = (M, D)$, where $M = \{G, A\}$, G represents groups of traffic based on proximity measure D, and A is the estimated score computed from each

group. The system I labels each traffic instance as normal or anomalous w.r.t. the estimated score, A.

A generic architecture of an unsupervised ANIDS is given in Fig. 2.28. This includes almost all the modules found in a supervised ANIDS except the anomaly detection engine and the labeling technique. We discuss the modules below:

(a) ***Unsupervised engine***: This module is the heart of an anomaly detection system. It consists of two modules, viz., *detection* and *labeling*. Based on the approach used, the *detection* module either groups similar instances or identifies exceptional instances in input data. The *labeling* module works after completion of the *detection* module to label each instance either as normal or anomalous based on the characteristics of each individual group such as size, compactness, the dominating subset of features, and outlier score of each instance.

(b) ***Labeling strategy***: A clustering method merely groups the data without any interpretation of the nature of the groups. To support appropriate interpretation of the groups, labeling techniques are used. Labeling of clusters is a difficult issue. A labeling strategy typically makes the following assumptions [54]:

- The number of normal instances vastly outnumbers the number of anomalous instances.
- Anomalies themselves are qualitatively different from normal instances.
- Similarity among the instances of an anomalous group is higher than the same among instances in a normal group.

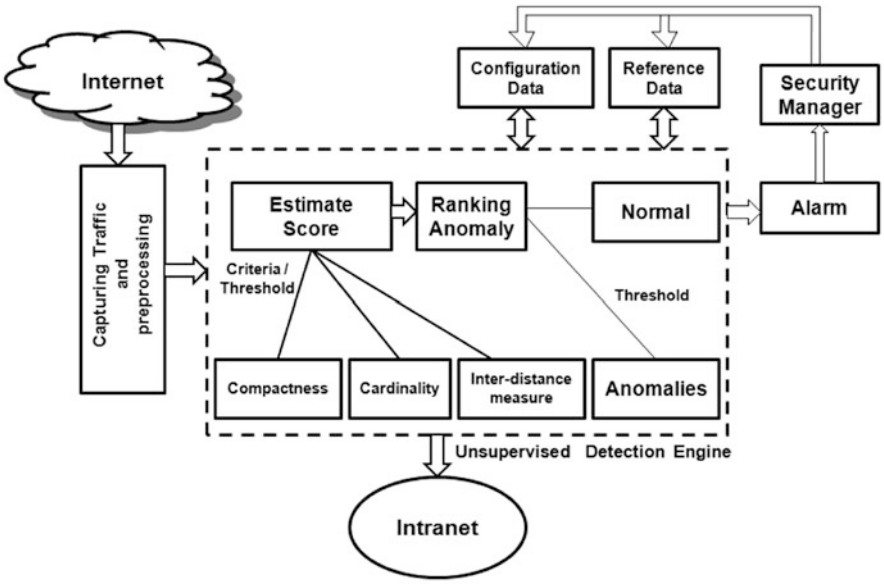


Fig. 2.28 A generic architecture of unsupervised ANIDS

Unsupervised anomaly detection approaches work without any training data. In other words, these models are trained on unlabeled or unclassified data, and they attempt to find intrusions lurking inside the data. The biggest advantage of the anomaly detection approach is the detection of unknown intrusions without any previous knowledge. In order to label clusters, an unsupervised ANIDS models normal behavior by using certain assumptions [54] presented above. If these assumptions hold, intrusive instances can be identified based on characteristics of the group the instances belong to. However, these assumptions are not always true, especially in the case of DDoS attacks [10]. Therefore, accurate labeling of an instance is a significant and crucial issue in an unsupervised ANIDS.

2.9.4 Hybrid ANIDS

A hybrid ANIDS combines both supervised and unsupervised approaches of network anomaly detection. Such approaches can detect known as well as unknown attacks. One type of hybrid approach attempts to identify known attacks based on a supervised model with reference to a set of sample training data using an appropriate matching mechanism. The test instances that neither belong to normal nor any of the known attack instances are handled by the unsupervised model for the identification of new normal or novel intrusions. Several successful efforts have been proposed by researchers to develop hybrid ANIDSs [4, 60, 73]. A hybrid system is defined as follows.

Definition 2.4 Let I be a hybrid anomaly-based detection system. It can be thought of as a pair $I = (M, D)$, where $M = \{B, U\}$, B represents the supervised module that uses proximity measure D to detect known attacks, and U is the unsupervised module which uses estimated score computed from each group to detect unknown attacks.

A generic architecture of a hybrid ANIDS is given in Fig. 2.29. The modules in this architecture are the same as in supervised and unsupervised ANIDSs discussed above, except the detection engine. This detection engine is a combination of a supervised module and an unsupervised module. As shown in the figure, the unsupervised module is used for only undetected test instances forwarded by the supervised module. Once a novel intrusion is identified and confirmed, its reference (i.e., rule or signature) is built and inserted into the rule-base for future reference of the supervised module.

The performance of an individual approach, either supervised or unsupervised, is not equally good for detection of all categories of attack as well as normal instances. There is the possibility of obtaining good detection accuracy for all categories in a dataset by using an appropriate combination of multiple well-performing detection approaches. The objective of such a combination is to provide the best performance from each participating approach for all attack classes. The selection of a supervised or unsupervised method at a particular level for a given dataset is a critical issue for the hybrid ANIDS.

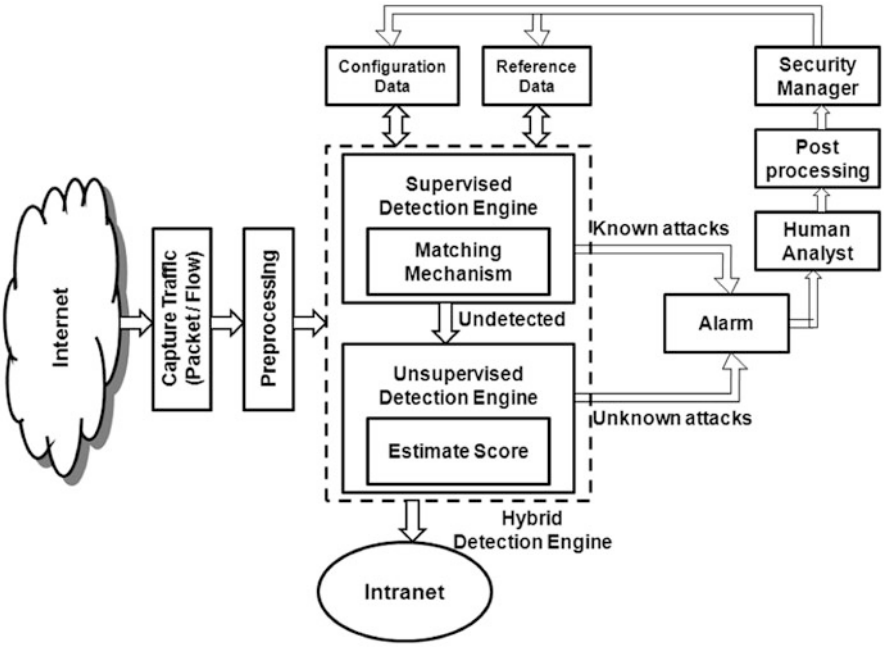


Fig. 2.29 A generic architecture of a hybrid ANIDS

2.10 Aspects of Network Traffic Anomaly Detection

In this section, we present some important aspects of anomaly-based network intrusion detection that we should be aware of. The network intrusion detection problem is a classification or clustering problem formulated with the following components [15]: (i) types of input data, (ii) appropriate proximity measures, (iii) labeling of data, (iv) identification of relevant features, and (v) reporting of anomalies. We discuss each of these topics in brief.

2.10.1 Types of Input Data

A key aspect of any anomaly-based network intrusion detection technique is the nature of the input data used for analysis. Input is generally a collection of data instances, also referred to as objects, records, points, vectors, patterns, events, cases, samples, observations, entities [64]. Each data instance can be described using a set of attributes of binary, categorical or numeric type. Each data instance may consist of only one attribute (univariate) or multiple attributes (multivariate). In the case of multivariate data instances, all attributes may be of the same type or may be a mixture of data types. The nature of attributes determines the applicability of anomaly detection techniques.

2.10.2 Appropriateness of Proximity Measures

Proximity (similarity or dissimilarity) measures are necessary to solve many pattern recognition problems in classification and clustering. Distance is a quantitative degree of how far apart two objects are. Distance measures that satisfy metric properties [64] are simply called *metric*, while other nonmetric distance measures are occasionally called *divergence*. The choice of a proximity measure depends on the measurement type or representation of objects.

Generally, a proximity measure is a function that takes an object pair as argument and returns a numerical value that becomes higher as the objects become more alike. A proximity measure is usually defined as follows.

Definition 2.5 A proximity measure D is a function $X \times X \rightarrow \mathbb{R}$ that has the following properties [43].

- Positivity: $\forall_{x,y} \in X, D(x, y) \geq 0$
- Symmetry: $\forall_{x,y} \in X, D(x, y) = D(y, x)$
- Maximality: $\forall_{x,y} \in X, D(x, x) \geq D(x, y)$

where X is the data space (also called the *universe*) and x, y are a pair of k-dimensional objects.

The most common proximity measures for numeric [14, 17, 44], categorical [11], and mixed type [25] data are listed in Table 2.5. For numeric data, it is assumed that the data is represented as real vectors. The attributes take their values from a continuous domain. In Table 2.5, we assume that there are two objects, $x = x_1, x_2, x_3 \cdots x_d$, $y = y_1, y_2, y_3 \cdots y_d$, and $\sum^{-1}$ represents the data covariance with d number of attributes, i.e., dimensions.

For categorical data, computing similarity or proximity measures is not straightforward owing to the fact that there is no explicit notion of ordering among categorical values. The simplest way to find the similarity between two categorical attributes is to assign a similarity of 1 if the values are identical and a similarity of 0 if the values are not identical. In Table 2.5, $D_k(x_k, y_k)$ represents per-attribute similarity. The attribute weight w_k for attribute k is computed as shown in the table. Consider a categorical dataset X containing n objects, defined over a set of d categorical attributes where A_k denotes the kth attribute. $D_k(x_k, y_k)$ is the per-attribute proximity between two values for the categorical attribute A_k. Note that $x_k, y_k \in A_k$. In Table 2.5, *IOF* denotes *inverse occurrence frequency* and *OF* denotes *occurrence frequency* [11].

Finally, mixed type data includes both categorical and numeric values. A common practice in clustering a mixed dataset is to transform categorical values into numeric values and then use a numeric clustering algorithm. Another approach is to compare the categorical values directly, in which two distinct values result in a distance of 1 while identical values result in a distance of 0. Of course, other transformations for categorical data can be used as well. Two well-known proximity measures, general similarity coefficient and general distance coefficient [25], for mixed type data are shown in Table 2.5. Such methods may not take into account

Table 2.5 Proximity measures for numeric, categorical, and mixed type data

Numeric [14]

Name	Measure, $D_i(x_i, y_i)$	Name	Measure, $D_i(x_i, y_i)$
Euclidean	$\sqrt{\sum_{i=1}^{d} \lvert x_i - y_i \rvert^2}$	Weighted Euclidean	$\sqrt{\sum_{i=1}^{d} \alpha_i \lvert x_i - y_i \rvert^2}$
Squared Euclidean	$\sum_{i=1}^{d} \lvert x_i - y_i \rvert^2$	Squared-chord	$\sum_{i=1}^{d} (\sqrt{x_i} - \sqrt{y_i})^2$
Squared X^2	$\sum_{i=1}^{d} \frac{(x_i - y_i)^2}{x_i + y_i}$	City block	$\sum_{i=1}^{d} \lvert x_i - y_i \rvert$
Minkowski	$\sqrt[p]{\sum_{i=1}^{d} \lvert x_i - y_i \rvert^p}$	Chebyshev	$\max_i \lvert x_i - y_i \rvert$
Canberra	$\frac{\sum_{i=1}^{d} \lvert x_i - y_i \rvert}{x_i + y_i}$	Cosine	$\frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2} \sqrt{\sum_{i=1}^{d} y_i^2}}$
Jaccard	$\frac{\sum_{i=1}^{d} x_i y_i}{\sum_{i=1}^{d} x_i^2 + \sum_{i=1}^{d} y_i^2 - \sum_{i=1}^{d} x_i y_i}$	Bhattacharyya	$-\ln \sum_{i=1}^{d} \sqrt{(x_i y_i)}$
Pearson	$\sum_{i=1}^{d} (x_i - y_i)^2$	Divergence	$2 \sum_{i=1}^{d} \frac{(x_i - y_i)^2}{(x_i + y_i)^2}$
Mahalanobis	$\sqrt{(x-y)^t \sum^{-1} (x-y)}$	–	–

(continued)

Table 2.5 (continued)

Numeric [14]			
$w_k, k{=}1 \ldots d$	Measure, $D_k(x_k, y_k)$	$w_k, k{=}1 \ldots d$	Measure $D_k(x_k, y_k)$
Categorical [11]			
$\frac{1}{2}$	$Overlap = \begin{cases} 1 & \text{if } x_k = y_k \\ 0 & \text{otherwise} \end{cases}$	$\frac{1}{d}$	$Eskin = \begin{cases} 1 & \text{if } x_k = y_k \\ \frac{n_k^2}{n_k^2 + 2} & \text{otherwise} \end{cases}$
$\frac{1}{d}$	$IOF = \begin{cases} 1 & \text{if } x_k = y_k \\ \frac{1}{1 + \log f_k(x_k) x \log f_k(y_k)} & \text{otherwise} \end{cases}$	$\frac{1}{d}$	$OF = \begin{cases} 1 & \text{if } x_k = y_k \\ \frac{1}{1 + \log \frac{N}{f_k(x_k)} x \log \frac{N}{f_k(y_k)}} & \text{otherwise} \end{cases}$
Mixed [25]			
General similarity coefficient	$D_{gsc}(x, y) = \frac{1}{\sum_{k=1}^{d} w(x_k, y_k)} \sum_{k=1}^{d} w(x_k, y_k) D(x_k, y_k)$, • For *numeric* attributes, $D(x_k, y_k) = 1 - \frac{\lvert x_k - y_k \rvert}{R_k}$, where R_k is the range of the kth attribute; $w(x_k, y_k) = 0$ if x or y has missing value for the kth attribute; otherwise $w(x_k, y_k) = 1$. • For *categorical* attributes, $D(x_k, y_k) = 1$ if $x_k = y_k$; otherwise $D(x_k, y_k) = 0$; $w(x_k, y_k) = 0$ if data point x or y has missing value at kth attribute; otherwise $w(x_k, y_k) = 1$	General distance coefficient	$D_{gdc}(x, y) = \left(\frac{1}{\sum_{k=1}^{d} w(x_k, y_k)} \sum_{k=1}^{d} w(s_k, y_k) D^2(x_k, y_k) \right)^{\frac{1}{2}}$, where $D^2(x_k, y_k)$ is the squared distance for the kth attribute; $w(x_k, y_k)$ is the same as in General Similarity Coefficient. • For *numeric* attributes, $D(x_k, y_k) = \frac{\lvert x_k - y_k \rvert}{R_k}$, where R_k is the range of kth attribute. • For *categorical* attributes, $D(x_k, y_k) = 0$ if $x_k = y_k$; otherwise $D(x_k, y_k) = 1$

the similarity information embedded in categorical values. Consequently, clustering may not faithfully reveal the similarity structure in the dataset [25, 30].

Information theory is a self-contained formal mathematical theory, which is again branch of probability theory [18]. Information theoretic measures can be used for comparing clusters, when the measures are computed after clusters have been formed. Suppose two clusters C_1 and C_2 are obtained from a cluster-based network anomaly detection method, and then entropy, joint entropy, conditional entropy, and mutual information (MI) are defined via the marginal and joint distributions of data items in C_1 and C_2, respectively.

$$Entropy, H(C_1) = -\sum_{i=1}^{R} \frac{a_i}{N} \log \frac{a_i}{N}$$

$$Joint\ Entropy, H(C_1, C_2) = -\sum_{i=1}^{R} \sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}}{N}$$

$$Conditional\ Entropy, H(C_1|C_2) = -\sum_{i=1}^{R} \sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}/N}{b_j/N}$$

$$Mutual\ Information, I(C_1, C_2) = \sum_{i=1}^{R} \sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}/N}{a_i b_j/N^2}$$

where N is the total number of data items in a cluster, n_{ij} is the number of data items common to clusters C_1 and C_2, R is the total number of rows or data instances, C is the total number of columns or features, a_i is the row sum, and b_j is the column sum. Mutual information can be employed as the most basic similarity measure in information theory. Several normalized versions of the mutual information (NMI) are given in Table 2.6. All these normalized variants are bounded in the range [0, 1]. 1 means that two clusters C_1 and C_2 are identical and 0 means they are independent, sharing no information with each other.

2.10.3 Labeling of Data

The label associated with a data instance denotes if the instance is normal or anomalous. It should be noted that obtaining accurate labeled data of both normal or anomalous types is often very difficult and expensive. Labeling is often done manually by human experts and hence substantial effort is required to obtain the labeled training dataset [15]. Moreover, anomalous behavior is often dynamic in nature, e.g., new types of anomalies may arise, for which there is no labeled training data.

Table 2.6 Information theoretic similarity measures

Reference	Measure	Formula	Range
Banerjee et al. [6]	Mutual Information (MI)	$I(C_1, C_2)$	$[0, \min\{H(C_1), H(C_2)\}]$
Yao [71]	Normalized Mutual Information (NMI) – joint	$\frac{I(C_1,C_2)}{H(C_1,C_2)}$	[0, 1]
Kvalseth [40]	Normalized Mutual Information (NMI) – max	$\frac{I(C_1,C_2)}{\max\{H(C_1),H(C_2)\}}$	[0, 1]
Kvalseth [40]	Normalized Mutual Information (NMI) – sum	$\frac{2I(C_1,C_2)}{H(C_1)+H(C_2)}$	[0, 1]
Strehl and Ghosh [61]	Normalized Mutual Information (NMI) – sqrt	$\frac{I(C_1,C_2)}{\sqrt{H(C_1)H(C_2)}}$	[0, 1]
Liu et al. [46], Kvalseth [40]	Normalized Mutual Information (NMI) – min	$\frac{I(C_1,C_2)}{\min\{H(C_1),H(C_2)\}}$	[0, 1]

2.10.4 Relevant Feature Selection

Feature selection plays an important role in detecting network anomalies. Feature selection methods are used in the intrusion detection domain for eliminating unimportant or irrelevant features. Feature selection reduces computational complexity, removes information redundancy, increases the accuracy of the detection algorithm, facilitates data understanding, and improves generalization. The feature selection process includes three major steps: (a) subset generation, (b) subset evaluation, and (c) validation. Three different approaches for subset generation are: *complete*, *heuristic*, and *random*. Evaluation functions are categorized into five [20] distinct categories: score-based, entropy or mutual information-based, correlation-based, consistency-based, and detection accuracy-based. Simulation and real-world implementation are two ways to validate the evaluated subset. A conceptual framework of the feature selection process is shown in Fig. 2.30.

Feature selection algorithms have been classified into three types: *wrapper*, *filter*, and *hybrid* [16]. While wrapper methods try to optimize some predefined criteria with respect to the feature set as part of the selection process, filter methods rely on the general characteristics of the training data to select features that are independent of each other and are highly dependent on the output. A hybrid feature selection method attempts to exploit the salient features of both wrapper and filter methods [16].

An example of a wrapper-based feature selection method is represented in [45], where the authors propose an algorithm to build a lightweight IDS using modified Random Mutation Hill Climbing (RMHC) as a search strategy to specify a candidate subset for evaluation and using a modified linear support vector machine (SVM)-based iterative procedure as a wrapper approach to obtain an optimum feature subset. The authors establish the effectiveness of their method in terms of efficiency

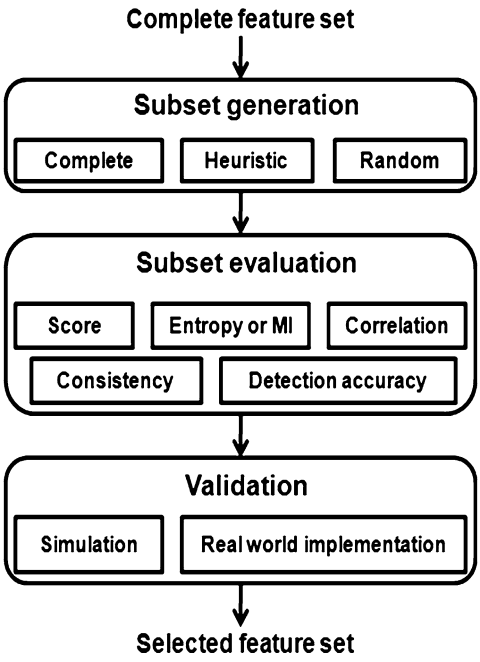


Fig. 2.30 Framework of feature selection process

in intrusion detection without compromising the detection rate. An example filter model for feature selection is given in [48], where the authors use correlation based and minimal redundancy-maximal-relevance measures. They evaluate their method on benchmark intrusion datasets for classification accuracy. Several other methods for feature selection are found in [2, 37, 52, 63].

2.10.5 Reporting Anomalies

An important aspect of any anomaly detection technique is the manner in which anomalies are reported [15]. Typically, the outputs produced by anomaly detection techniques are of two types: (a) a *score*, which is a value that combines (i) distance or deviation with reference to a set of profiles or signatures, (ii) influence of the majority in its neighborhood, and (iii) distinct dominance of the relevant subspace (as discussed in Sect. 2.10.4), and (b) a *label*, which is a value (normal or anomalous) given to each test instance. Usually the labeling of an instance depends on (i) the size of groups generated by an unsupervised technique, (ii) the compactness of the group(s), (iii) majority voting based on the outputs given by multiple indices (several example indices are given in Table 2.7), or (iv) distinct dominance of a subset of features.

Table 2.7 Cluster validity measures

Reference	Name of index	Formula	Remark(s)
Dunn [22]	Dunn index	$DI = \frac{d_{\min}}{d_{\max}}$, where $d_{\min}$ denotes the smallest distance between two objects from different clusters, $d_{\max}$ the largest distance within the same cluster	(i) Can identify dense and well-separated clusters. (ii) High Dunn index is more desired for a clustering algorithm. (iii) May not perform well with noisy data
Davies et al. [21]	Davies Bouldin's index	$DB = \frac{1}{n}\sum_{i=1, i\neq j}^{n} \max(\frac{\sigma_j+\sigma_j}{d(c_i,c_j)})$, where n is the number of clusters; σ_i is the average distance of all patterns in cluster i to their cluster center, c_i; σ_j is the average distance of all patterns in cluster j to their cluster center, c_j; and $d(c_i, c_j)$ represents the proximity between the cluster centers c_i and c_j	(i) Validation is performed using cluster quantities and features inherent to the dataset. (ii) For compact clustering, DB values should be as minimum as possible. (iii) It is not designed to accommodate overlapping clusters
Hubert and Schultz [31]	C-index	$C = \frac{S-S_{\min}}{S_{\max}-S_{\min}}$, where S is the sum of distances over all pairs of objects from the same cluster, n is the number of those pairs, $S_{\min}$ and $S_{\max}$ are the sum of n smallest distances and n largest distances, respectively	It needs to be minimized for better clustering
Baker and Hubert [5]	Gamma index	$G = \frac{S+ - S-}{S+ + S-}$, where (S+) represents the number of times that a pair of samples not clustered together have a larger separation than a pair that were in the same clusters; $(S-)$ represents reverse outcome	This measure is widely used for hierarchical clustering
Rohlf [57]	G+ index	$G+ = \frac{2(S-)}{n*(n-1)}$, where $(S-)$ is defined as for gamma index and n is the number of within cluster distances	It uses the minimum value to determine the number of clusters in the data
Rousseeuw [58]	Silhouette index	$SI = \frac{b_i-a_i}{\max\{a_i,b_i\}}$, where a_i is the average dissimilarity of the ith-object to all other objects in the same cluster; b_i is the minimum of average dissimilarity of the ith-object to all objects in other cluster;	This index cannot be applied to datasets with sub-clusters
Goodman and Kruskal [27]	Goodman-Kruskal index	$GK = \frac{N_c-N_d}{N_c+N_d}$, where N_c and N_d are the numbers of concordant and disconcordant quadruples, respectively	(i) It is robust in outliers detection. (ii) It requires high computation complexity in comparison to C-index

(continued)

Table 2.7 (continued)

Reference	Name of index	Formula	Remark(s)
Jaccard [33]	Jaccard index	$JI = \frac{a}{a+b+c}$, where a denotes the number of pairs of points with the same label in C and assigned to the same cluster in k, b denotes the number of pairs with the same label, but in different clusters and c denotes the number of pairs in the same cluster, but with different class labels	It uses less information than the Rand index measure
Rand [55]	Rand index	$RI = \frac{a+d}{a+b+c+d}$, where d denotes the number of pairs with a different label in C that were assigned to a different cluster in k, rest are same with JI	It gives equal weights to false positives and false negatives during computation
Bezdek [8]	Partition coefficient	$PC = \frac{1}{n}\sum_{i=1}^{N}\sum_{j=1}^{n_c} u_{ij}^2$, where n_c is the number of clusters, N is the number of objects in the dataset, u_{ij} is the degree of membership	(i) It finds the number of overlaps between clusters, (ii) It lacks connection with dataset
Bezdek [7]	Classification entropy	$CE = \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{n} u_{ij}\log(u_{ij})$, same with partition coefficient	It measures the fuzziness of the cluster partitions
Xie and Beni [70]	Xie-Beni index	$XB = \frac{\pi}{N.d_{\min}}$, where $\pi = \frac{\sigma_i}{n_i}$, is called compactness of cluster i. Since n_i is the number of points in cluster i, σ, is the average variation in cluster i; $d_{\min} = \min\|\|k_i - k_j\|\|$	(i) It combines the properties of membership degree and the geometric structure of dataset. (ii) Smaller XB means more compact and better separated clusters

2.10.6 Post-processing Anomalies

Post-processing of anomalies is another important module in anomaly detection for high-speed networks. As technologies advance rapidly, gigabyte Ethernet and optical switching are being deployed in border routers, and as a result, the rates of packet forwarding at these aggregation routers are outpacing the performance gains in personal computer hardware. Therefore, post-processing has become important [53]. For example, Plonka [53] post-processes NetFlow traffic to detect abuse on a large-scale network in real time. This tool is known as FlowScan. The post-processing ensures discovery of the actual network traffic anomalies.

2.11 Chapter Summary

In this chapter, we have introduced fundamental concepts in anomalies that commonly arise in networks. We note that there are two major categories of network anomalies, viz., performance-related anomalies and security-related anomalies. We describe network vulnerabilities that exist with their possible sources. An attacker exploits these vulnerabilities to cause network failure or degrade performance. In addition, we discuss sources of security-related anomalies, types of network attacks, steps to launch an attack, and a taxonomy of attacks. We also introduce the main categories of network anomaly detection methods with architectures, components, and pros and cons. These concepts are used in detection and prevention methods discussed in subsequent chapters.

References

1. 3com: Understanding IP addressing: everything you ever wanted to know. White Paper 501302-001, 3Com Corporation, CA (2001)
2. Amiri, F., Yousefi, M.M.R., Lucas, C., Shakery, A., Yazdani, N.: Mutual information-based feature selection for intrusion detection systems. J. Netw. Comput. Appl. **34**(4), 1184–1199 (2011)
3. Anderson, J.: Computer security threat monitoring and surveillance. Tech. Rep. 215 646-4706, James P Anderson Co., Fort Washington, Pennsylvania (1980)
4. Bahrololum, M., Salahi, E., Khaleghi, M.: Anomaly intrusion detection design using hybrid of unsupervised and supervised neural network. Int. J. Comput. Netw. Commun. **1**(2), 26–33 (2009)
5. Baker, F.B., Hubert, L.J.: Measuring the power of hierarchical cluster analysis. J. Am. Stat. Assoc. **70**(349), 31–38 (1975)
6. Banerjee, A., Dhillon, I.S., Ghosh, J., Sra, S.: Clustering on the unit hypersphere using Von Mises-Fisher distributions. J. Mach. Learn. Res. **6**, 1345–1382 (2005)
7. Bezdek, J.C.: Cluster validity with fuzzy sets. J. Cybern. **3**(3), 58–78 (1974)
8. Bezdek, J.C.: Numerical taxonomy with fuzzy sets. J. Math. Biol. **1**(1), 57–71 (1974)
9. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Surveying port scans and their detection methodologies. Comput. J. **54**(10), 1565–1581 (2011)
10. Bhuyan, M.H., Kalwar, A., Goswami, A., Bhattacharyya, D.K., Kalita, J.K.: Low-rate and high-rate distributed DoS attack detection using partial rank correlation. In: 2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT), pp. 706–710 (2015). doi:10.1109/CSNT.2015.24
11. Boriah, S., Chandola, V., Kumar, V.: Similarity measures for categorical data: a comparative evaluation. In: Proceedings of the 8th SIAM International Conference on Data Mining, pp. 243–254 (2008)
12. Burbeck, K., Nadjm-tehrani, S.: ADWICE – anomaly detection with real-time incremental clustering. In: Proceedings of the 7th International Conference on Information Security and Cryptology, Seoul, Korea. Springer (2004)
13. Burbeck, K., Nadjm-Tehrani, S.: Adaptive real-time anomaly detection with incremental clustering. Inf. Secur. Tech. Rep. **12**(1), 56–67 (2007)
14. Cha, S.H.: Comprehensive survey on distance/similarity measures between probability density functions. Int. J. Math. Models Methods Appl. Sci. **1**(4), 300–307 (2007)

15. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**(3), 15:1–15:58 (2009)
16. Chen, Y., Li, Y., Cheng, X.Q., Guo, L.: Survey and taxonomy of feature selection algorithms in intrusion detection system. In: Proceedings of the Second SKLOIS Conference on Information Security and Cryptology, pp. 153–167. Springer (2006). doi:10.1007/11937807_13
17. Choi, S., Cha, S., Tappert, C.C.: A survey of binary similarity and distance measures. J. Syst. Cybern. Inform. **8**(1), 43–48 (2010)
18. Cover, T.M., Thomas, J.A.: Elements of information theory. Wiley-Interscience, New York (1991)
19. Dainotti, A., Pescape, A.: PLAB: a packet capture and analysis architecture (2004). Http://www.grid.unina.it/software/ITG/D-ITGpublications/TR-DIS-122004.pdf
20. Dash, M., Liu, H.: Feature selection for classification. Intell. Data Anal. **1**, 131–156 (1997)
21. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **1**(2), 224–227 (1979)
22. Dunn, J.C.: Well separated clusters and optimal fuzzy partitions. J. Cybern. **4**(1), 95–104 (1974)
23. Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P., Kumar, V., Srivastava, J.: MINDS – Minnesota intrusion detection system. In: Next Generation Data Mining, chap. 3, pp. 1–21. CRC Press, Boca Raton (2004)
24. Fujimaki, R., Yairi, T., Machida, K.: An approach to spacecraft anomaly detection problem using kernel feature space. In: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 401–410. ACM, New York (2005)
25. Gan, G., Ma, C., Wu, J.: Data Clustering Theory, Algorithms and Applications. SIAM, Philadelphia (2007)
26. Ghorbani, A.A., Lu, W., Tavallaee, M.: Network Intrusion Detection and Prevention: Concepts and Techniques. Advances in Information Security. Springer, Boston (2009)
27. Goodman, L., Kruskal, W.: Measures of associations for cross-validations. J. Am. Stat. Assoc. **49**, 732–764 (1954)
28. Hansman, S., Hunt, R.: A taxonomy of network and computer attacks. Comput. Secur. **24**(1), 31–43 (2005)
29. Hsu, C.C., Huang, Y.P.: Incremental clustering of mixed data based on distance hierarchy. J. Expert Syst. Appl. **35**(3), 1177–1185 (2008)
30. Hsu, C.C., Wang, S.H.: An integrated framework for visualized and exploratory pattern discovery in mixed data. IEEE Trans. Knowl. Data Eng. **18**(2), 161–173 (2005)
31. Hubert, L., Schultz, J.: Quadratic assignment as a general data analysis strategy. Br. J. Math. Stat. Psychol. **29**(2), 190–241 (1976)
32. Hunt, R., Hansman, S.: A Taxonomy of Network and Computer Attack Methodologies. University of Canterbury, New Zealand (2003)
33. Jaccard, P.: The distribution of flora in the alpine zone. New Phytol. **11**(2), 37–50 (1912)
34. Jacobson, V., Leres, C., McCanne, S.: tcpdump. ftp://ftp.ee.lbl.gov/tcpdump.tar.gz
35. Joshi, M.V., Agarwal, R.C., Kumar, V.: Mining needle in a haystack: classifying rare classes via two-phase rule induction. SIGMOD Rec. **30**(2), 91–102 (2001)
36. Kalyankar, N.V.: Network traffic management. J. Comput. **1**(1), 191–194 (2009)
37. Kayacik, H.G., Zincir-Heywood, A.N., Heywood, M.I.: Selecting features for intrusion detection: a feature relevance analysis on KDD'99 intrusion detection datasets. In: Proceedings of the Third Annual Conference on Privacy, Security and Trust (PST-2005) (2005)
38. KDDcup99: Knowledge Discovery in Databases DARPA Archive. http://www.kdd.ics.uci.edu/databases/kddcup99/task.html (1999)
39. Kizza, J.M.: Computer Network Security, 1st edn. Springer, New York (2005)
40. Kvalseth, T.O.: Entropy and correlation: some comments. IEEE Trans. Syst. Man Cybern. **17**(3), 517–519 (1987). doi:10.1109/TSMC.1987.4309069
41. Lau, S.: The spinning cube of potential doom. Commun. ACM **47**(6), 25–26 (2004)
42. Lee, W., Stolfo, S.J.: Data mining approaches for intrusion detection. In: Proceedings of the 7th Conference on USENIX Security Symposium, vol. 7, pp. 1–7. USENIX Association, Berkeley (1998)

43. Lesot, M.J., Rifqi, M.: Anomaly-based network intrusion detection: techniques, systems and challenges. Int. J. Knowl. Eng. Soft Data Paradigms **1**(1), 63–84 (2009)
44. Lesot, M.J., Rifqi, M., Benhadda, H.: Similarity measures for binary and numerical data: a survey. Int. J. Knowl. Eng. Soft Data Paradigms **1**(1), 63–84 (2009). doi:10.1504/IJKESDP.2009.021985
45. Li, Y., Wang, J.L., Tian, Z., Lu, T., Young, C.: Building lightweight intrusion detection system using wrapper-based feature selection mechanisms. Comput. Secur. **28**(6), 466–475 (2009)
46. Liu, Z., Lin, D., Guo, F.: A method for locating digital evidences with outlier detection using support vector machine. Int. J. Netw. Secur. **6**(3), 301–308 (2008)
47. McCanne, S., Jacobson, V.: The BSD packet filter: a new architecture for user level packet capture. In: Proceedings of the Winter 1993 USENIX Conference, pp. 259–269. USENIX Association (1993)
48. Nguyen, H.T., Franke, K., Petrovic, S.: Towards a Generic Feature-Selection Measure for Intrusion Detection. In: Proceedings of the 20th International Conference on Pattern Recognition, pp. 1529–1532 (2010)
49. Ning, P., Jajodia, S.: Intrusion detection techniques. In: Bidgoli, H. (ed.) The Internet Encyclopedia, vol. 2, John-Wiley & Sons, US (2003)
50. Park, B.C., Won, Y.J., Kim, M.S., Hong, J.W.: Towards automated application signature generation for traffic identification. In: Proceedings of the IEEE/IFIP Network Operations and Management Symposium: Pervasive Management for Ubiquitous Networks and Services, pp. 160–167 (2008)
51. Paulauskas, N., Garsva, E.: Computer system attack classification. Electron. Electr. Eng. Technol. Kaunas **2**(66), 84–87 (2006)
52. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Mach. Intell. **27**(8), 1226–1238 (2005)
53. Plonka, D.: FlowScan: a network traffic flow reporting and visualization tool. In: Proceedings of the 14th USENIX Conference on System Administration, LISA'00, pp. 305–318. USENIX Association, Berkeley (2000)
54. Portnoy, L., Eskin, E., Stolfo, S.: Intrusion detection with unlabeled data using clustering. In: Proceedings of the ACM CSS Workshop on Data Mining Applied to Security, Philadelphia, pp. 5–8 (2001)
55. Rand, W.M.: Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc. **66**(336), 846–850 (1971)
56. Ren, F., Hu, L., Liang, H., Liu, X., Ren, W.: Using density-based incremental clustering for anomaly detection. In: Proceedings of the International Conference on Computer Science and Software Engineering, pp. 986–989. IEEE Computer Society, Washington, DC (2008)
57. Rohlf, F.J.: Methods of comparing classifications. Ann. Rev. Ecol. Syst. **5**(1), 101–113 (1974)
58. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**(1), 53–65 (1987)
59. Sadoddin, R., Ghorbani, A.: Alert correlation survey: framework and techniques. In: Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services, pp. 37:1–37:10. ACM, New York (2006). doi:10.1145/1501434.1501479
60. Shon, T., Moon, J.: A hybrid machine learning approach to network anomaly detection. Inf. Sci. **177**, 3799–3821 (2007)
61. Strehl, A., Ghosh, J.: Cluster ensembles – a knowledge reuse framework for combining multiple partitions. J. Mach. Learn. Res. **3**, 583–617 (2003). doi:10.1162/153244303321897735. http://dx.doi.org/10.1162/153244303321897735
62. Sundaram, A.: An introduction to intrusion detection. Crossroads **2**(4), 3–7 (1996)
63. Sung, A.H., Mukkamala, S.: Identifying important features for intrusion detection using support vector machines and neural networks. In: Proceedings of the Symposium on Applications and the Internet, pp. 209–217. IEEE Computer Society (2003)

64. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining, 4th edn. Addison-Wesley, Pearson Education, India (2009)
65. Theiler, J., Cai, D.M.: Resampling approach for anomaly detection in multispectral images. In: Proceedings of SPIE, vol. 5093, pp. 230–240. SPIE (2003)
66. Thottan, M., Ji, C.: Anomaly detection in IP networks. IEEE Trans. Signal Process. **51**(8), 2191–2204 (2003)
67. Uhlig, S.: Implications of traffic characteristics on interdomain traffic engineering. Ph.D. thesis, Université catholique de Louvain (2004)
68. Wikimedia, F.: Intrusion detection system. http://en.wikipedia.org/wiki/Intrusion-detection_system (2009)
69. Wu, Z., Ou, Y., Liu, Y.: A taxonomy of network and computer attacks based on responses. In: Proceedings of the International Conference on Information Technology, Computer Engineering and Management Sciences, pp. 26–29. IEEE Computer Society, Nanjing, Jiangsu (2011)
70. Xie, X.L., Beni, G.: A validity measure for fuzzy clustering. IEEE Trans. Pattern Anal. Mach. Intell. **13**(4), 841–847 (1991)
71. Yao, Y.Y.: Information-theoretic measures for knowledge discovery and data mining. In: Entropy Measures, Maximum Entropy Principle and Emerging Applications. Studies in Fuzziness and Soft Computing, vol. 119, pp. 115–136. Springer, New York (2003)
72. Zhang, C., Zhang, G., Sun, S.: A mixed unsupervised clustering-based intrusion detection model. In: Proceedings of the 3rd International Conference on Genetic and Evolutionary Computing, pp. 426–428. IEEE Computer Society, USA (2009)
73. Zhang, J., Zulkernine, M.: Anomaly based network intrusion detection with unsupervised outlier detection. In: Proceedings of the IEEE International Conference on Communications, vol. 5, pp. 2388–2393 (2006)
74. Zhong, C., Li, N.: Incremental clustering algorithm for intrusion detection using clonal selection. In: Proceedings of the IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, pp. 326–331. IEEE Computer Society, Washington, DC (2008)
75. Zhu, X.: Semi-supervised learning literature survey. Tech. Rep. 1530, Department of Computer Sciences, University of Wisconsin-Madison (2005)

Chapter 3
A Systematic Hands-On Approach to Generate Real-Life Intrusion Datasets

To evaluate a network anomaly detection or prevention, it is essential to test using benchmark network traffic datasets. This chapter aims to provide a systematic hands-on approach to generate real-life intrusion dataset. It is organized in three major sections. Section 3.1 provides the basic concepts. Section 3.2 introduces several benchmark and real-life datasets. Finally, Sect. 3.3 provides a systematic approach toward generation of an unbiased real-life intrusion datasets. We establish the importance of intrusion datasets in the development and validation of a detection mechanism or a system, identify a set of requirements for effective dataset generation, and discuss several attack scenarios.

3.1 Introduction

In network intrusion detection, particularly when using anomaly-based detection, it is difficult to accurately evaluate, compare, and deploy a system that is expected to detect novel attacks due to scarcity of adequate datasets. Before deploying in any real-world environment, an anomaly-based network intrusion detection system (ANIDS) needs to be tested and evaluated using real and labeled network traffic traces with a comprehensive set of intrusions or attacks. To generate such a dataset is a challenging task, since not many such datasets are available. Therefore, the detection methods and systems are evaluated only with a few publicly available datasets that lack comprehensiveness and completeness. For example, Cooperative Association for Internet Data Analysis (CAIDA) distributed denial of service (DDoS) 2007, Lawrence Berkeley National Laboratory (LBNL), and ICSI datasets are heavily anonymized without payload information and decreasing research utility. Researchers also frequently use a single NetFlow-based intrusion dataset found at [33] with limited number of attacks.

M.H. Bhuyan et al., *Network Traffic Anomaly Detection and Prevention*,
Computer Communications and Networks, DOI 10.1007/978-3-319-65188-0_3

3.1.1 Importance of Datasets

In network traffic anomaly detection, it is always important to test and evaluate detection methods and systems using datasets as network scenarios evolve. We enumerate the following reasons to justify the importance of a dataset:

- *Repeatability of experiments*: Researchers should be able to repeat experiments with the dataset and get similar results, when using the same approach. This is important because the proposed method should cope with the evolving nature of attacks and network scenarios.
- *Validation of new approaches*: New methods and algorithms are being continuously developed to detect network anomalies. It is necessary that every new approach be validated.
- *Comparison of different approaches*: State-of-the-art network anomaly detection methods must not only be validated but also show improvements over older methods in performance in a quantifiable manner. For example, the DARPA 1998 dataset [18] is commonly used for performance evaluation of anomaly detection systems [16].
- *Parameters tuning*: To properly obtain the model to classify the normal traffic from malicious traffic, it is necessary to tune model parameters. Network anomaly detection generally assumes the normality model to identify malicious traffic. For example, Cemerlic et al. [5] and Thomas et al. [37] use the attack-free part of the DARPA 1999 dataset for training to estimate parameter values.
- *Dimensionality or the number of features*: An optimal set of features or attributes should be considered to represent normal as well as all possible attack instances.

3.1.2 Requirements

Although good datasets are necessary for validating and evaluating IDSs, generating such datasets is a time-consuming task. A dataset generation approach should meet the following requirements:

- *Real world*: A dataset should be generated by monitoring the daily situation in a realistic way, such as the daily network traffic of an organization.
- *Completeness in labeling*: The labeling of traffic as benign or malicious must be backed by proper evidence for each instance. The aim these days should be to provide labeled datasets at both packet and flow levels for each piece of benign and malicious traffic.
- *Correctness in labeling*: Given a dataset, labeling of each traffic instance must be correct. This means that our knowledge of security events represented by the data has to be certain.
- *Sufficient trace size*: The generated dataset should be unbiased in terms of size in both benign and malicious traffic instances.

- *Concrete feature extraction*: Extraction of an optimal set of concrete features when generating a dataset is important because such features play an important role during validating a detection mechanism.
- *Diverse attack scenarios*: With the increasing frequency, size, variety, and complexity of attacks, intrusion threats have become more complex including the selection of targeted services and applications. When contemplating attack scenarios for dataset generation, it is important to tilt toward a diverse set of multistage attacks that are recent.
- *Ratio between normal and attack traffic*: Most existing datasets have been created based on the following assumptions.
  - Anomalous traffic is statistically different from normal traffic [10].
  - The majority of network traffic instances is normal [29].

  Unlike most traditional intrusion datasets, DDoS attacks do not follow these assumptions because they change attack traffic rate dynamically and employ considered multistage attacks.

3.2 Existing Datasets

As discussed earlier, datasets play an important role in the testing and validation of network anomaly detection methods or systems. A good quality dataset allows us to identify the ability of a method or a system to detect anomalous behavior, when deployed in real operating environments. Several datasets are publicly available for testing and evaluation of network anomaly detection methods and systems. A taxonomy of network intrusion datasets is shown in Fig. 3.1. We briefly discuss each of them below.

3.2.1 Synthetic Datasets

Synthetic datasets are generated to meet specific needs or certain conditions or tests that real data satisfy. Such datasets are useful when designing any prototype system for theoretical analysis so that the design can be refined. As stated previously, a synthetic dataset can be used to test and create many different types of test scenarios. This enables designers to build realistic behavior profiles for normal users and attackers based on the dataset to validate an IDS. This provides initial validation of a specific method or a system; if the results prove to be satisfactory, the developers then continue to evaluate a method or a system in a specific domain.

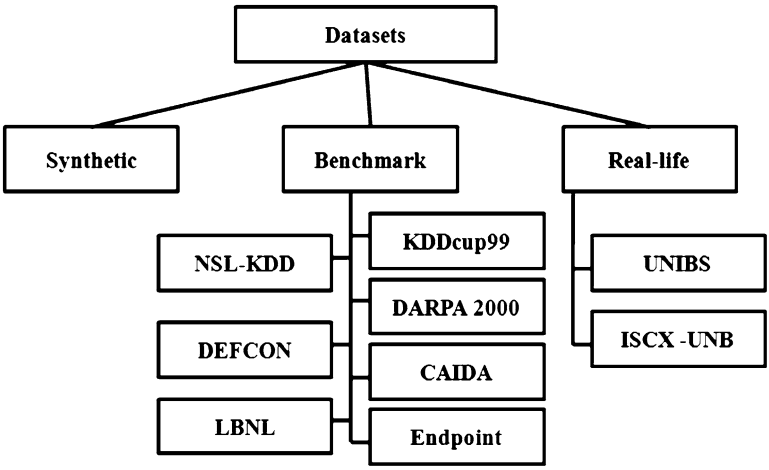


Fig. 3.1 A taxonomy of network intrusion datasets

3.2.2 Benchmark Datasets

We discuss seven publicly available benchmark datasets generated using simulated environments in large networks, executing different attack scenarios.

3.2.2.1 KDDcup99 Dataset

Since 1999, the KDDcup99 dataset [14] has been the most widely used dataset for evaluation of network-based anomaly detection methods and systems. This dataset was prepared by Stolfo et al. [34] and is built upon the data captured in the DARPA98 IDS evaluation program. The KDD training dataset consists of approximately 4,900,000 single connection vectors, each of which contains 41 features and is labeled as either normal or attack of a specific attack type. The test dataset contains about 300,000 samples with a total 24 training attack types, with an additional 14 attack types in the test dataset only [11]. The represented attacks are mainly four types: denial of service, remote to local, user to root, and surveillance or probing.

- *Denial of service (DoS)*: An attacker attempts to prevent valid users from using a service provided by a system. Examples include SYN flood, smurf, and teardrop attacks.
- *Remote to local (r2l)*: Attackers try to gain entrance to a victim machine without having an account on it. An example is the password guessing attack.
- *User to root (u2r)*: Attackers have access to a local victim machine and attempt to gain privilege of a superuser. Examples include buffer overflow attacks.

- *Probe*: Attackers attempt to acquire information about the target host. Some examples of probe attacks are port-scans and ping-sweep attacks.

In this dataset, background traffic was simulated and the attacks were all known. The training set, consisting of 7 weeks of labeled data, is available to the developers of intrusion detection systems. The testing set also consists of simulated background traffic and known attacks, including some attacks that are not present in the training set. The distribution of normal and attack traffic for this dataset is reported in Table 3.1. We also identify the services associated with each category of attacks [9, 15] and summarize them in Table 3.2.

3.2.2.2 NSL-KDD Dataset

Analysis of the KDD dataset showed that there were two important issues with the dataset which highly affect the performance of evaluated systems resulting in poor evaluation of anomaly detection methods [36]. To address these issues, a new dataset known as NSL-KDD [25], consisting of selected records of the complete KDD dataset, was introduced. This dataset is publicly available for researchers[1] and has the following advantages over the original KDD dataset:

- It does not include redundant records in the training set, so classifiers will not be biased toward more frequent records.
- There are no duplicate records in the test set. Therefore, the performance of learners is not biased by the methods which have better detection rates on frequent records.
- The number of selected records from each difficulty level is inversely proportional to the percentage of records in the original KDD dataset. As a result, the classification rates of various machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of various learning techniques.
- The number of records in the training and testing sets is reasonable, which makes it affordable to run experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research groups are consistent and comparable.

The NSL-KDD dataset consists of two parts: (i) KDDTrain$^+$ and (ii) KDDTest$^+$. The distribution of attack and normal instances in the NSL-KDD dataset is shown in Table 3.3.

[1] http://www.iscx.ca/NSL-KDD/

Table 3.1 Distribution of normal and attack traffic instances in KDDCup99 dataset

	DoS		Probe		u2r		r2l		
Dataset	Total instances	Attacks	Total instances	Attacks	Total instances	Attacks	Total instances	Attacks	Normal
10% KDD	391,458	smurf, neptune, back,	4,107	satan, ipsweep, portsweep, nmap	52	buffer_overflow, rootkit, loadmodule,	1,126	warezclient, guess_passwd, warezmaster, imap,	97,277
Corrected KDD	229,853	teardrop, pod, land	4,107		52	perl	1,126	ftp_write, multihop, phf, spy	97,277
Whole KDD	229,853		4,107		52		1,126		97,277

Table 3.2 List of attacks and corresponding services in KDDcup99 dataset

	DoS		Probe		u2r		r2l	
Dataset	Attack name	Service(s)	Attack name	Service(s)	Attack name	Service(s)	Attack name	Service(s)
KDD99	apache2	http	ipsweep	icmp	eject	Any user session	dictionary	telnet, rlogin, pop, imap, ftp
	back	http	mscan	many	ffbconfig	Any user session	ftp-write	ftp
	land	N/A	nmap	many	fdformat	Any user session	guest	telnet, rlogin
	mailbomb	smtp	saint	many	loadmodule	Any user session	imap	imap
	SYN flood	Any TCP	satan	many	perl	Any user session	named	dns
	ping of death	icmp	–	–	ps	Any user session	named	dns
	process table	Any TCP			Xterm	Any user session	sendmail	smtp
	smurf	icmp			–	–	xlock	X
	syslogd	syslog					xsnoop	X
	teardrop	N/A					–	–
	udpstorm	echo/chargen						

Table 3.3 Distribution of normal and attack traffic instances in NSL-KDD dataset

Dataset	DoS	u2r	r2l	Probe	Normal	Total
KDDTrain$^+$	45,927	52	995	11,656	67,343	125,973
KDDTest$^+$	7,458	67	2,887	2,422	9,710	22,544

3.2.2.3 DARPA 2000 Dataset

A DARPA[2] evaluation project [23] targeted the detection of complex attacks that contain multiple steps. Two attack scenarios were simulated in the DARPA 2000 evaluation contest, namely, Lincoln Laboratory scenario DDoS (LLDOS) 1.0 and LLDOS 2.0. To achieve variations, these two attack scenarios were carried out over several network and audit scenarios. These sessions were grouped into four attack phases: (a) probing, (b) breaking into the system by exploiting vulnerability, (c) installing DDoS software for the compromised system, and (d) launching DDoS attack against another target. LLDOS 2.0 is different from LLDOS 1.0 in that attacks are more stealthy and thus harder to detect. Since this dataset contains multistage attack scenarios, it is also commonly used for evaluation of alert correlation techniques.

3.2.2.4 DEFCON Dataset

The DEFCON[3] dataset is another commonly used dataset for evaluation of IDSs [8]. It contains network traffic captured during a hacker competition called Capture The Flag (CTF), in which competing teams are divided into two groups: *attackers* and *defenders*. The traffic produced during CTF is very different from real-world network traffic since it contains only intrusive traffic without any normal background traffic. Due to this limitation, DEFCON dataset has been found useful only in evaluating alert correlation techniques.

3.2.2.5 CAIDA Dataset

CAIDA[4] (Center for Applied Internet Data Analysis) collects many different types of data and makes them available to the research community. CAIDA datasets [4] are very specific to particular events or attacks. Most of its longer traces are anonymized backbone traces without their payload. The CAIDA DDoS 2007 attack dataset contains 1 h of anonymized traffic traces from DDoS attacks on August

[2] http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html

[3] http://cctf.shmoo.com/data/

[4] http://www.caida.org/home/

4, 2007, which attempted to consume a large amount of network resources when connecting to Internet servers. The traffic traces contain only attack traffic to the victim and responses from the victim with 5 min split form. All traffic traces are in pcap (tcpdump) format. The creators removed non-attack traffic as much as possible when creating the CAIDA DDoS 2007 dataset.

3.2.2.6 LBNL Dataset

LBNL's internal enterprise traffic traces are full header network traces [17] without payload. This dataset suffers from heavy anonymization to the extent that scanning traffic was extracted and separately anonymized to remove any information which could identify individual IPs. The background and attack traffic in the LBNL dataset are described below:

- *LBNL background traffic:* This dataset can be obtained from the Lawrence Berkeley National Laboratory (LBNL) in the USA. Traffic in this dataset is comprised of packet level incoming, outgoing, and internally routed traffic streams at the LBNL edge routers. Traffic was anonymized using the *tcpmkpub* tool [28]. The main applications observed in the internal and external traffic are Web, email, and name services. Other applications like Windows services, network file services, and backup were used by internal hosts. The details of each service and information on each packet and other relevant description are given in [27]. The background network traffic statistics of LBNL dataset are given in Table 3.4.
- *LBNL attack traffic:* This dataset identifies attack traffic by isolating scans in aggregate traffic traces. Scans are identified by flagging those hosts which unsuccessfully probe more than 20 hosts, out of which 16 hosts are probed in ascending or descending IP order [28]. Malicious traffic mostly consists of failed incoming TCP SYN requests, i.e., TCP port scans targeted toward LBNL hosts. However, there are also some outgoing TCP scans in the dataset. Most UDP traffic observed in the data (incoming and outgoing) is comprised of successful connections, i.e., host replies for the received UDP flows. Clearly, the attack rate is significantly lower than the background traffic rate. Details of the attack traffic in this dataset are shown in Table 3.4. Complexity and privacy were two main reservations of the participants of the endpoint data collection study. To address these reservations, the dataset creators developed a custom multi-

Table 3.4 Background and attack traffic information for the LBNL datasets

Date	Duration (mins)	LBNL hosts	Remote hosts	Background traffic rate (packet/sec)	Attack traffic rate (packet/sec)
10/04/2004	10 min	4,767	4,342	8.47	0.41
12/15/2004	60 min	5,761	10,478	3.5	0.061
12/16/2004	60 min	5,210	7,138	243.83	72

Table 3.5 Background traffic information for four endpoints with high and low rates

Endpoint ID	Endpoint type	Duration (months)	Total sessions	Mean session rate (/sec)
3	Home	3	3,73,009	1.92
4	Home	2	4,44,345	5.28
6	University	9	60,979	0.19
10	University	13	1,52,048	0.21

Table 3.6 Endpoint attack traffic for two high-rate and two low-rate worms

Malware	Release date	Avg. scan rate (/sec)	Port (s) used
Dloader-NY	Jul 2005	46.84 sps	TCP 1,35,139
Forbot-FU	Sept 2005	32.53 sps	TCP 445
Rbot-AQJ	Oct 2005	0.68 sps	TCP 1,39,769
MyDoom-A	Jan 2006	0.14 sps	TCP 3127–3198

threaded MS Windows tool using the Winpcap API [3] for data collection. To reduce packet logging complexity at the endpoints, they only logged very elementary session-level information (bidirectional communication between two IP addresses on different ports) for the TCP and UDP packets. To ensure user privacy, an anonymization policy was used to anonymize all traffic instances.

3.2.2.7 Endpoint Dataset

The background and attack traffic for the endpoint datasets are explained below.

- *Endpoint background traffic:* In the endpoint context, we see in Table 3.5 that home computers generate significantly higher traffic volumes than office and university computers because (*i*) they are generally shared between multiple users, and (*ii*) they run peer-to-peer and multimedia applications. The large traffic volumes of home computers are also evident from their high mean number of sessions per second. To generate attack traffic, the developers infected Virtual Machines (VMs) on the endpoints with different malware, viz., Zotob.G, Forbot-FU, Sdbot-AFR, Dloader-NY, So-Big.E@mm, MyDoom.A@mm, Blaster, Rbot-AQJ, and RBOT.CCC. Details of the malware can be found in [35]. Characteristics of the attack traffic in this dataset are given in Table 3.6. These malwares have diverse scanning rates and attack ports or applications.
- *Endpoint attack traffic:* The attack traffic logged at the endpoints is mostly comprised of outgoing port scans. Note that this is the opposite of the LBNL dataset, in which most attack traffic is inbound. Moreover, the attack traffic rates at the endpoints are generally much higher than the background traffic rates of the LBNL datasets. This diversity in attack direction and rates provides a sound basis for performance comparison among scan detectors. For each malware, attack traffic of 15 min duration was inserted in the background traffic for each

endpoint at a random time instance. This operation was repeated to insert 100 nonoverlapping attacks of each worm inside each endpoint's background traffic.

3.2.3 Real-Life Datasets

We discuss three real-life datasets created by collecting network traffic on several consecutive days during a week or a month. The details include both normal as well as attack traffic in appropriate proportions in the authors' respective campus networks (i.e., testbed).

3.2.3.1 UNIBS Dataset

The UNIBS packet traces [38] were collected on the edge router of the campus network of the University of Brescia in Italy, on three consecutive working days. The dataset includes traffic captured or collected and stored using 20 workstations, each running the GT (Ground Truth) client daemon. The dataset creators collected the traffic by running tcpdump on the faculty router, which was a dual Xeon Linux box that connected the local network to the Internet through a dedicated 100 Mb/s uplink. They captured and stored the traces on a dedicated disk of a workstation connected to the router through a dedicated ATA controller.

3.2.3.2 ISCX-UNB Dataset

Real packet traces [30] were analyzed to create profiles for agents that generate real traffic for HTTP, SMTP, SSH, IMAP, POP3, and FTP protocols. Various multistage attack scenarios were explored to generate malicious traffic.

3.2.3.3 KU Dataset

The Kyoto University dataset[5] is a collection of network traffic data obtained from honeypots. The raw dataset obtained from the honeypot system consisted of 24 statistical features, out of which 14 significant features were extracted [31]. The dataset developers extracted *ten* additional features that could be used to investigate network events inside the university more effectively. However, they used 14 conventional features only during training and testing.

[5] http://www.takakura.com/kyoto_data

3.2.4 Discussion

The datasets described above are valuable assets for the intrusion detection community. However, the benchmark datasets suffer from the fact that they are not good representatives of real-world traffic. For example, the DARPA dataset has been questioned about the realism of the background traffic [19, 21] because it is synthetically generated. In addition to the difficulty of simulating real-life network traffic, there are additional challenges in IDS evaluation [22]. These include difficulties in collecting attack scripts and victim software, differing requirements for testing signature-based vs. anomaly-based IDSs and host-based vs. network-based IDSs.

3.3 Hands-On for Real-Life Dataset Generation

As noted above, the generation of an unbiased real-life intrusion dataset incorporating a large number of real-world attacks is important to evaluate network anomaly detection methods and systems. In this chapter, we describe the generation of three real-life network intrusion datasets[6] including (a) a TUIDS (Tezpur University Intrusion Detection System) intrusion dataset, (b) a TUIDS coordinated port scan dataset, and (c) a TUIDS DDoS dataset at both packet and flow levels [12]. The resulting details and supporting infrastructure is discussed in the following subsections.

3.3.1 Testbed Network Architecture

The TUIDS testbed network consists of 250 hosts, 15 Layer 2(L2) switches, 8 Layer 3 (L3) switches, 3 wireless controllers, and 4 routers that compose 5 different networks inside the Tezpur University campus. The architectures of the TUIDS testbed and TUIDS testbed for DDoS dataset generation are given in Figs. 3.2 and 3.3, respectively. The hosts are divided into several VLANs, each VLAN belonging to an L3 switch or an L2 switch inside the network. All servers are installed inside a DMZ (demilitarized zone)[7] to provide an additional layer of protection in the security system of an organization.

[6] http://agnigarh.tezu.ernet.in/~dkb/resources.html

[7] A demilitarized zone is a network segment located between a secure local network and unsecure external networks (Internet). A DMZ usually contains servers that provide services to users on the external network, such as Web, mail, and DNS servers, that are hardened systems. Typically, two firewalls are installed to form the DMZ.

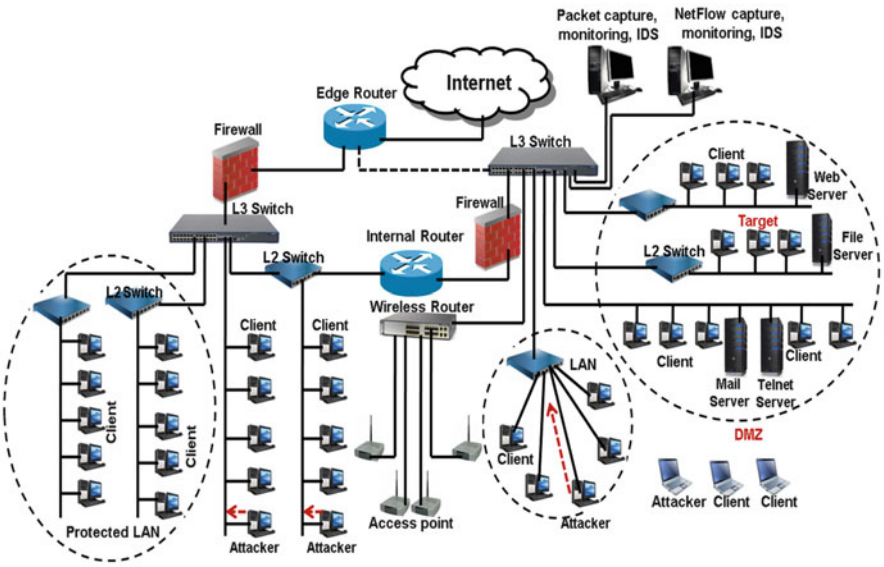


Fig. 3.2 Testbed network architecture used during TUIDS dataset generation

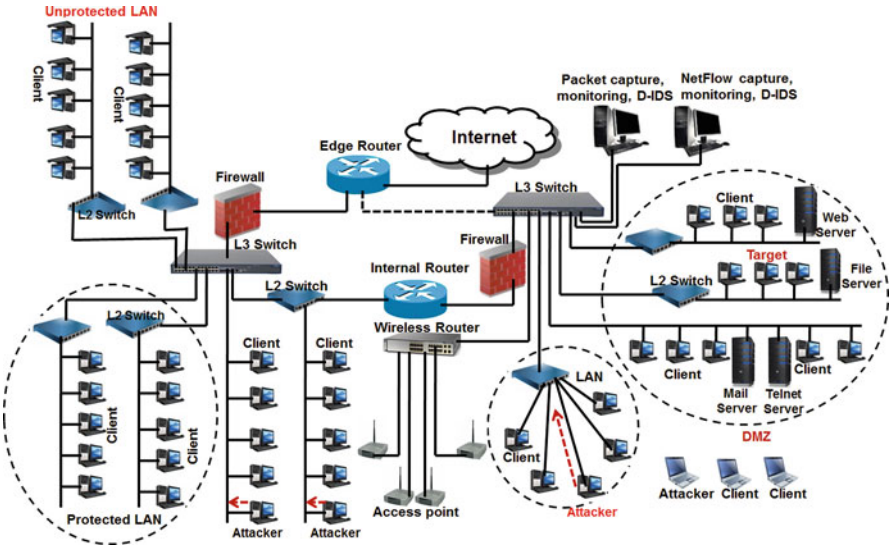


Fig. 3.3 Testbed network architecture used during DDoS dataset generation

Table 3.7 Servers and their services running on the testbed network

Server	Operating system	Services	Provider
Main server	Ubuntu 10.10	Web, eMail	Apache 2.4.3, Dovecot 2.1.14
Network file server	Ubuntu 10.10	Samba	Samba 4.0.2
Telnet server	Ubuntu 10.10	Telnet	telnet-0.17-36bulid1
FTP server	Ubuntu 10.10	ftp	vsFTPd 2.3.0
Windows server	Windows server 2003	Web	IIS v7.5
MySQL server	Ubuntu 10.10	Database	MySQL 5.5.30

3.3.2 Network Traffic Generation

To generate real-life normal and attack traffic, we configured several hosts, workstations, and servers in the TUIDS testbed network. The network consists of *six* interconnected Ubuntu 10.10 workstations. On each workstation, we have installed several servers including a network file server (Samba), a mail server (Dovecot), a telnet server, an FTP server, a Web server, and an SQL server with PHP compatibility. We also installed and configured 4 Windows Servers 2003 to exploit a diverse set of known vulnerabilities against the testbed environment. Servers and their services running in our testbed are summarized in Table 3.7.

The normal network traffic is generated based on the day-to-day activities of users and especially generated traffic from configured servers. It is important to generate different types of normal traffic. So, we capture traffic from students, faculty members, system administrators, and office staff on different days within the university. The attack traffic is generated by launching attacks within the testbed network in three different subsets, viz., a TUIDS intrusion dataset, a coordinated scan dataset, and a DDoS dataset. The attacks launched in the generation of these real-life datasets are summarized in Table 3.8.

As seen in the table above, 22 distinct attack types (1–22 in Table 3.8) were used to generate the attack traffic for the TUIDS intrusion dataset; six attacks (17–22 in Table 3.8) were used to generate the attack traffic for the coordinated scan dataset and finally six attacks (23–28 in Table 3.8) were used to generate the attack traffic for a DDoS dataset with combination of TCP, UDP, and ICMP protocols.

3.3.3 Attack Scenarios

The attack scenarios start with information gathering techniques collecting target network IP ranges, identities of name servers, mail servers, user email accounts, etc. This is achieved by querying the DNS for resource records using network administrative tools like nslookup and dig. nslookup and dig commands work as follows.

Table 3.8 List of real-life attacks and their generation tools

Attack name	Generation tool	Attack name	Generation tool
1.bonk	targa2.c	15.linux-icmp	linux-icmp.c
2.jolt	targa2.c	16.syn flood	synflood.c
3.land	targa2.c	17.window-scan	nmap/rnmap
4.saihyousen	targa2.c	18.syn-scan	nmap/rnmap
5.teardrop	targa2.c	19.xmasstree-scan	nmap/rnmap
6.newtear	targa2.c	20.fin-scan	nmap/rnmap
7.1234	targa2.c	21.null-scan	nmap/rnmap
8.winnuke	targa2.c	22.udp-scan	nmap/rnmap
9.oshare	targa2.c	23.syn-flood(DDoS)	LOIC
10.nestea	targa2.c	24.rst-flood(DDoS)	Trinity v3
11.syndrop	targa2.c	25.udp-flood(DDoS)	LOIC
12.smurf	smurf4.c	26.ping-flood(DDoS)	DDoS ping v2.0
13.opentear	opentear.c	27.fraggle udp-flood(DDoS)	Trinoo
14.fraggle	fraggle.c	28.smurf icmp-flood(DDoS)	TFN2K

Command: root@monowar-TravelMate-5744Z:/mhb♯ **nslookup google.co.in**
Meaning: Find out "A" details of a domain

Command: root@monowar-TravelMate-5744Z:/mhb♯ **dig springlink.com**
Meaning: Find out "A" details of a domain

We consider six attack scenarios when collecting real-life network traffic for dataset generation.

3.3.3.1 *Scenario 1*: Denial of Service Using Targa

This attack scenario is designed toward performing attacks on a target using the targa[8] tool until it is successful. Targa is a very powerful tool to quickly damage a particular network belonging to an organization within short time interval. Targa is program in C that can be used to launch eleven different denial of service (DoS) attacks. It was developed by Mixter and initial version is available at https://dl.packetstormsecurity.net/DoS/targa.c for download. But the revised version is available at ftp://ftp.ntua.gr/mirror/technotronic/denial/targa2.c for download. The attacker has both option to launch individual attacks and as a whole all attacks at a time. *targa.c* is the original version of the program that contains only eight varieties of DoS attacks. But *targa*2.*c* contains eleven different DoS attacks. To compile *targa*2.*c*, it needs arpa, netinet, time, unistd, netdb, string, and sys C libraries.

[8] http://packetstormsecurity.com/

To compile and execute the targa2.c, you can use the following commands:

```
root@monowar-TravelMate-5744Z:/targa♯make  targa2
cc targa2.c -o targa2
root@monowar-TravelMate-5744Z:/targa♯./targa2
targa 2.0 by Mixter
usage: ./targa <startIP> <endIP> [-t type] [-n repeats]
type ./targa - -h to get more help
```

As seen in the output of the above command, targa2 needs three parameters such as:

- IP range where attack traffic will send
- Type of attack that you want to launch
- Number of times you want to repeat a particular attack to the target

We ran *targa*2 by specifying different parameter values such as IP ranges, attacks to run, and number of times to repeat the attack. Targa2 attack tool works as follows for various attacks.

Command: root@monowar-TravelMate-5744Z:/targa♯**./targa2 172.16.153.15 172.16.25.231 -t0 1**
Meaning: It runs *all the attacks* once with IP range provided in the command.

Command: root@monowar-TravelMate-5744Z:/targa♯**./targa2 172.16.153.15 172.16.25.231 -t1 5**
Meaning: It runs *bonk* attack and repeats five times in the specified IP range provided in the command. *Bonk* basically manipulates fragment offset field in TCP/IP packets. By manipulating this number, it causes the attempted machine to reorganize a packet that is too large to be reassembled.

Command: root@monowar-TravelMate-5744Z:/targa♯**./targa2 172.16.153.15 172.16.25.231 -t2 5**
Meaning: It runs *jolt* attack and repeats five times in the specified IP range provided in the command. *Jolt* communicated the target machine with a very large, fragmented ICMP packets. It fragments the ICMP packets in such a way to that the target machine is incompatible to reorganize them for use.

Command: root@monowar-TravelMate-5744Z:/targa♯**./targa2 172.16.153.15 172.16.25.231 -t3 5**
Meaning: It runs *land* attack and repeats five times in the specified IP range provided in the command. This attack is launched by sending a TCP SYN spoofed packet with the IP address of the target host and an open port using as source and destination port both. *Land* makes the attempted machine to respond to itself in continuous.

Command: root@monowar-TravelMate-5744Z:/targa♯**./targa2 172.16.153.15 172.16.25.231 -t4 5**
Meaning: It runs *nestea* attack and repeats five times in the specified IP range provided in the command. It launches IP fragments to a machine which is connected to the Internet or a network. *Nestea* is specific to the Linux operating system and exploits a bug (commonly known as the "off by one IP header" bug) in the Linux refragmentation code.

Command: root@monowar-TravelMate-5744Z:/targa♯**./targa2 172.16.153.15 172.16.25.231 -t5 5**
Meaning: It runs *newtear* attack and repeats five times in the specified IP range provided in the command. *Newtear* attempts to exploit a problem with a smarter way. The Microsoft TCP/IP stack handles exceptions caused due to misformed UDP header information, which changes padding length and increases the UDP header length field to twice the size of the packet.

Command: root@monowar-TravelMate-5744Z:/targa♯**./targa2 172.16.153.15 172.16.25.231 -t6 5**
Meaning: It runs *syndrop* attack and repeats five times in the specified IP range provided in the command. When a system reconstructs a packet, it performs a loop to store it in a new buffer. Actually, they control the size of the packet only if it is too big. If the size of the packet is too small, it can cause a kernel problem, which may lead to crash of the system.

Command: root@monowar-TravelMate-5744Z:/targa♯**./targa2 172.16.153.15 172.16.25.231 -t7 5**
Meaning: It runs *teardrop* attack and repeats five times in the specified IP range provided in the command. *Teardrop* exploits an overlapping IP fragment bug that causes the TCP/IP fragmentation reassembly code to improperly handle overlapping IP fragments. The fragmentation offset of the second segment is smaller than the size of the first and the offset plus the size of the second. This means that the second fragment contains the first.

Command: root@monowar-TravelMate-5744Z:/targa♯**./targa2 172.16.153.15 172.16.25.231 -t8 5**
Meaning: It runs *winnuke* attack and repeats five times in the specified IP range provided in the command. *Winnuke* is a window-based attack by sending OOB (out-of-band) data to an IP address of a Windows-based machine connected to the Internet or network. This attack program connects through port 139, but other ports are also vulnerable if they are open. Upon receiving OOB data, the victim Windows machine cannot handle it and results with an exhibition of odd behavior.

Command: root@monowar-TravelMate-5744Z:/targa♯**./targa2 172.16.153.15 172.16.25.231 -t9 5**

Meaning: It runs *1234* attack and repeats five times in the specified IP range provided in the command. 1234 attack is launched by sending an oversize ping packet which cannot be handled by the network software. As a result the victim machine becomes very slow and ultimately it hangs. It may also result with loss of data.

Command: root@monowar-TravelMate-5744Z:/targa♯**./targa2 172.16.153.15 172.16.25.231 -t10 5**
Meaning: It runs *saihyousen* attack and repeats five times in the specified IP range provided in the command. This attack is responsible for some firewalls to crash. It is launched by sending a stream of UDP packets. *Saihyousen* attack can consume all of the available resources and eventually cause a very messy reboot if this occurs continuously for about 10–30 s after the machine is frozen.

Command: root@monowar-TravelMate-5744Z:/targa♯**./targa2 172.16.153.15 172.16.25.231 -t11 5**
Meaning: It runs *oshare* attack and repeats five times in the specified IP range provided in the command. *Oshare* is a DoS attack. It is caused by sending a novel packet structure. The consequences of these attacks can be different such as complete system crash, CPU load increasing, or momentary delays, depending upon the configuration of the computer. This will cause effect almost all versions of Windows 98 and NT-based systems with varying degrees related to the involved hardware.

3.3.3.2 *Scenario 2*: Probing Using NMAP

In this scenario, we attempt to acquire information about the target host and then launch the attack by exploiting the vulnerabilities found using the nmap[9] tool. nmap is also known as network mapper, a free open-source port scanning tool for both attacker and network administrator. It is basically used to identify the host alive and services they are offering. It supports a large number of scans such as UDP, TCP connect, TCP Syn, FTP proxy, ICMP, FIN, ACK sweep, Xmas tree, SYN sweep, IP protocol, and NULL scan. nmap accepts several commands to discover host or services, but a few of them are explained below.

Command: root@monowar-TravelMate-5744Z:/nmap♯**nmap -v 172.16.5.19**
Meaning: Scan a host or an IP address (IPv4) with more information.

Command: root@monowar-TravelMate-5744Z:/nmap♯**nmap -sS -PN -p 1-65535 172.16.*.***
Meaning: It scans whole IP range (IPv4) with port range (1-65535). It skips ping scan due to -PN option and assume that host is up.

[9] http://nmap.org/

Command: root@monowar-TravelMate-5744Z:/nmap♯**nmap -sP 172.16.*.***
Meaning: It scans whole IP range to discover alive host.

3.3.3.3 *Scenario 3*: Coordinated Scan Using RNMAP

This scenario starts with a goal to perform coordinated port scans to single and multiple targets. Tasks are distributed among multiple hosts for individual actions which may be synchronized. We use the rnmap[10] tool to launch coordinated scans in our testbed network during the collection of traffic. rnmap is also known as remote nmap that contains both client and server programs. Clients are connected with centralized server to launch port scanning using nmap tool. Client version of rnmap is shown in Fig. 3.4

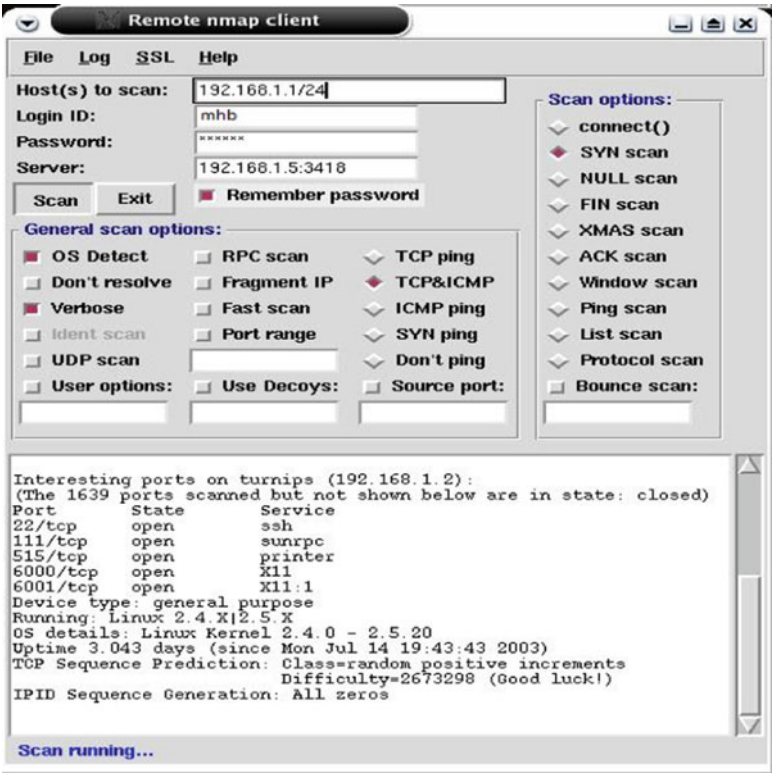


Fig. 3.4 Client version of rnmap(remote nmap) tool

[10] http://rnmap.sourceforge.net/

3.3.3.4 *Scenario 4*: User to Root Using Brute Force SSH

These attacks are very common against networks as they tend to break into accounts with weak username and password combinations. This attack has been designed with the goal of acquiring an SSH account by running a dictionary brute force attack against our central server. We use the brutessh[11] tool and a customized dictionary list. The dictionary consists of over 6,100 alphanumeric entries of varying length. We executed the attack for 60 min, during which superuser credentials were returned from the server. This ID and password combination was used to download other users' credentials immediately. It combines the username and password to get into the SSH server. It is a multi-threading program that uses 12 threads by default to accomplish the task. It has four different options:
-h: destination host
-u: username to force
-d: password file
-t: number of threads (default 12)
Command: root@monowar-TravelMate-5744Z:/brutessh♯**./brutessh.py -h 172.16.5.19 -u root -d mypass.txt**
Meaning: It tries to authenticate superuser password of the target host by matching with mypass.txt.

3.3.3.5 *Scenario 5*: Distributed Denial of Service Using Agent-Handler Network

This scenario mainly attempts to exploit an agent-handler network to launch the DDoS attack in the TUIDS testbed network. The agent-handler network consists of clients, handlers, and agents. The handlers are software packages that are used by the attacker to communicate indirectly with the agents. The agent software exists in compromised systems that will eventually carry out the attack on the victim system. The attacker may communicate with any number of handlers, thus making sure that the agents are up and running. We use Trinity v3, TFN2K, Trinoo, and DDoS ping 2.0 to launch the attacks in our testbed.

Trinity v3 [6] is a DDoS attack launching tool that usually controlled by IRC bots with the help of agents. The agent binary is installed in the Linux system and connects to the IRC server through port number 6,667. It has an agent known as x-force. Trinity v3 can be launched by using the following command.

Command: root@monowar-TravelMate-5744Z:/brutessh♯**./trinity trnd usr1 172.16.23.44 1000**
Meaning: It sends random flags flooding traffic to the target 172.16.23.44 for 1000 s. usr1 is the agent's password.

[11] http://www.securitytube-tools.net/

Trinity can use any of the following flood types during execution based on the command request:

tudp: UDP flood
tfrag: fragment flood
tsyn: SYN flood
trst: RST flood
trnd: random flags flood
tack: ACk flood
testab: establish flood
tnull: null flood

TFN2K [6] is a tool for launching DDoS attacks in distributed mode. It was written by Mixter and available for download at http://packetstormsecurity.com/. It has two parts: server and client. The server runs on a host that waiting for accepting commands from clients. To see the different options, type the following command:

Command: root@monowar-TravelMate-5744Z:/tfn2k♯**./tfn**
usage: ./tfn <options>
[-P protocol] Protocol for server communication. Can be ICMP, UDP or TCP. Uses a random protocol as default
[-D n] Send out n bogus requests for each real one to decoy targets
[-S host/ip] Specify your source IP. Randomly spoofed by default, you need to use your real IP if you are behind spoof filtering routers
[-f hostlist] Filename containing a list of hosts with TFN servers to contact
[-h hostname] To contact only a single host running a TFN server
[-i target string] Contains options/targets separated by "@"; see below
[-p port] A TCP destination port can be specified for SYN floods
<-c command ID> 0 - Halt all current floods on server(s) immediately
1 - Change IP antispoof-level (evade rfc2267 filtering)
usage: -i 0 (fully spoofed) to -i 3 (/24 host bytes spoofed).
2 - Change packet size, usage: -i <packet size in bytes>
3 - Bind root-shell to a port, usage: -i <remote port>
4 - UDP flood, usage: -i victim@victim2@victim3@...
5 - TCP/SYN flood, usage: -i victim@... [-p destination port]
6 - ICMP/PING flood, usage: -i victim@...
7 - ICMP/SMURF flood, usage: -i victim@broadcast@broadcast2@...
8 - MIX flood (UDP/TCP/ICMP interchanged), usage: -i victim@...
9 - TARGA3 flood (IP stack penetration), usage: -i victim@...
10 - Blindly execute remote shell command, usage -i command

Command: root@monowar-TravelMate-5744Z:/tfn2k♯**./td**
Meaning: Run the server host and make ready server for accepting command from client host.

Command: root@monowar-TravelMate-5744Z:/tfn2k♯**./tfn -h 14.243.26.30 -c8 23.246.65.23**
Meaning: TFN2K sends mixed flood traffic UDP/TCP/ICMP interchangeably to the target host.

Following command stops the attack from further sending malicious traffic to the target.
Command: root@monowar-TravelMate-5744Z:/tfn2k♯**./tfn -h 14.243.26.30 -c0**
Meaning: Stops the attack

In this tool, it verifies password when it tries to start and stop the attacks. This password is same with the password given during installation.

Trinoo [6] is another tool for launching DDoS attacks. TFN2K is stealthy in nature because it uses ICMP protocol. So, there are no ports to detect compromised host. Trinoo uses TCP and UDP protocols and it uses the following ports:

- Attacker to master: 27665/TCP
- Master to daemon: 27444/UDP
- Daemon to master: 31335/UDP

In trinoo, daemons are stored on the systems that actually launch the attack; however, master controls the daemon systems. At this point, it states below how to control master and daemon. The following are the commands used to control the master:

- Die: Halts the master
- Quit: Logs off of the master
- Mtimer N: DoS time sets for n number of seconds. It may be between 1 and 1999, if the value is less than one, it defaults to 300, and if it is greater than 2000, it defaults to 500.
- Dos IP: It launches a DoS attack against the specified IP address
- Die pass: Disables all broadcast hosts
- Mping: Sends a ping message to every active host on the broadcast address
- Mdos <ip1:ip2:ip3>: Similar to DoS IP, but it sends multiple DoS attack commands to each host.
- Info: Shows the version number and information about the program
- Msize: Sets the size of the buffer used during the DoS attacks
- Nslookup host: Performs a name server lookup of the specified host
- Killdead: Sends a message to all hosts with the goal of finding hosts that do not respond and removing them from the list
- Usebackup: Switches the program to use the file created by the kill dead command, which contains only the active hosts
- Bcast: Lists all active hosts
- Help [cmd]: Specifies additional information about a given command
- Mstop: Attempts to stop a DoS attack. This feature is listed in the help command.

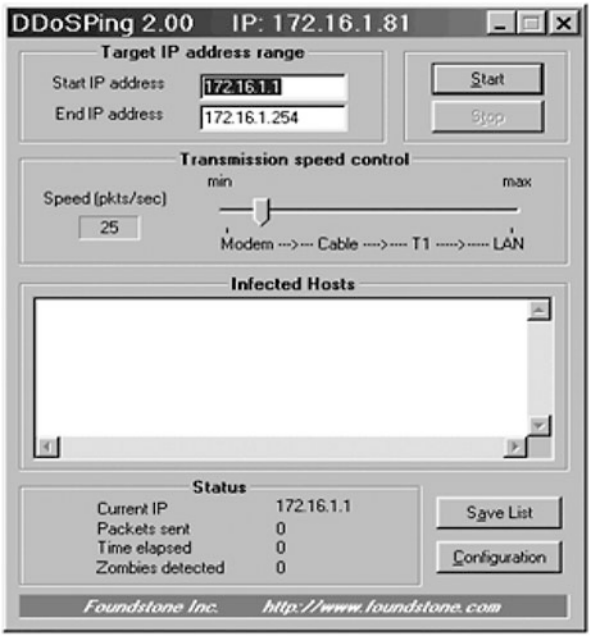


Fig. 3.5 DDoSPing 2.0 tool

The following are the commands used to access the trinoo daemons:

- aaa pass IP: Performs a DoS attack against the specified IP address
- bbb pass N: Sets the time limit for the DoS attack
- d1e pass: Used to shut down the daemons
- rsz N: Sets the size of the buffer that is used for the DoS attacks
- xyz pass 123:ip1:ip2:ip3: Performs DoS attacks against multiple IP addresses

DDoSping 2.0 [6] is a tool for scanning various DDoS agents that runs on the windows platform. Examples of DDoS agents are Trinoo, TFN2K, and Stacheldraht. The GUI for DDoSPing 2.0 is shown in Fig. 3.5.

Once it detects DDoS agents, the network administrator or network defender can take immediate action against those agents to keep always secure entire network.

3.3.3.6 *Scenario 6*: Distributed Denial of Service Using IRC Botnet

Botnets are emerging threats to all organizations because they can compromise a network and steal important information and distribute malware. Botnets combine individual malicious behaviors into a single platform by simplifying the actions needed to be performed by users to initiate sophisticated attacks against computers or networks around the world. These behaviors include coordinated scanning, distributed denial of service (DDoS) activities, direct attacks, indirect attacks, and other deceitful activities taking place across the Internet.

The main goal of this scenario is to perform distributed attacks using infected hosts on the testbed. Internet relay chat (IRC) bot network allows users to create public, private, and secret channels. For this, we use a LOIC[12] and an IRC-based DDoS attack generation tool. The IRC systems have several other significant advantages for launching DDoS attacks. Among the three important benefits are (i) they afford a high degree of anonymity, (ii) they are difficult to detect, and (iii) they provide a strong, guaranteed delivery system. Furthermore, the attacker no longer needs to maintain a list of agents, since he can simply log on to the IRC server and see a list of all available agents. The IRC channels receive communications from the agent software regarding the status of the agents (i.e., up or down) and participate in notifying the attackers regarding the status of the agents.

LOIC is a power tool for launching DDoS attacks through IRC botnet. It was developed by Praetox Technologies. It can send large sequence of UDP, TCP, or HTTP requests to the target server. It uses anonymous group of attacks and has an option to count the client to the IRC. So, it controls through IRC protocol. GUI for LOIC is shown in Fig. 3.6.

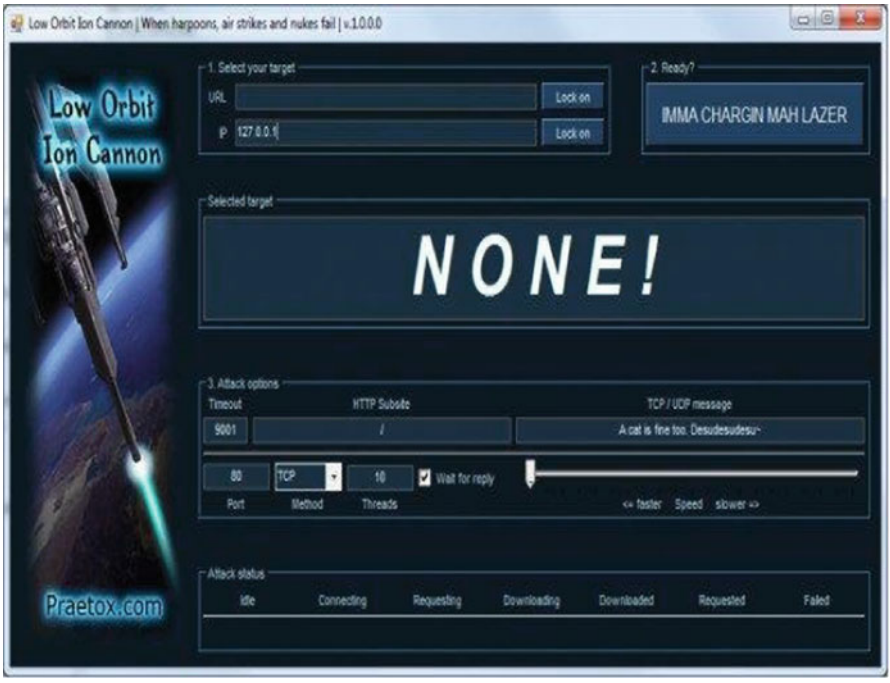


Fig. 3.6 LOIC tool

[12] http://sourceforge.net/projects/loic/

It works in three steps:

- Run the tool.
- Enter the URL of the website in the URL field and click on Lock O. Then, select attack method (TCP, UDP, or HTTP).
- Change other parameters as per choice or leave it to the default. Now click on the Big Button labeled as IMMA CHARGIN MAH LAZER. Then it immediately mounts the attack to the target.

3.3.4 Capturing Traffic

The key tasks in network traffic monitoring are lossless packet capturing and precise time stamping. Therefore, software or hardware is required with a guarantee that all traffic is captured and stored. The real network traffic is captured by using Libpcap [13] library, an open-source C library offering an interface for capturing link-layer frames over a wide range of system architectures. It provides a high-level common Application Programming Interface (API) to the different packet capture frameworks of various operating systems. The offered abstraction layer allows programmers to rapidly develop highly portable applications. A hierarchy of network traffic capturing components is given in Fig. 3.7 [7].

Libpcap defines a common standard format for files in which captured frames are stored, also known as the *tcpdump* format, currently a de facto standard used widely in public network traffic archives. Modern kernel-level capture frameworks on UNIX operating systems are mostly based on the BSD (or Berkeley) Packet Filter (BPF) [20]. The BPF is a software device that taps network interfaces,

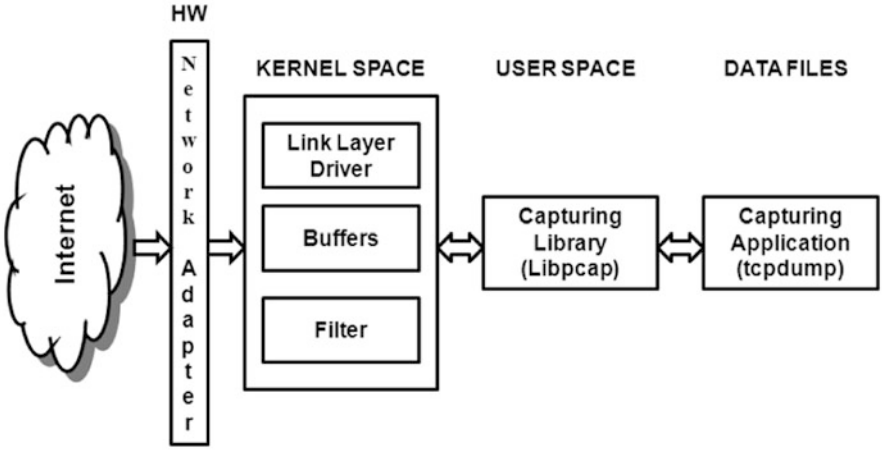


Fig. 3.7 Hierarchy of network traffic capturing components

copying packets into kernel buffers and filtering out unwanted packets directly in interrupt context. Definitions of packets to be filtered can be written in a simple human readable format using Boolean operators and can be compiled into a pseudo-code to be passed to the BPF device driver by a system call. The pseudo-code is interpreted by the BPF Pseudo-Machine, a lightweight, high-performance, state machine specifically designed for packet filtering. *Libpcap* also allows programmers to write applications that transparently support a rich set of constructs to build detailed filtering expressions for most network protocols. A few *Libpcap* system calls can be read (these Boolean expressions) directly from user's command line, compiled into pseudo-code, and passed to the Berkeley Packet Filter. *Libpcap* and the BPF interact to allow network packet data to traverse several layers to finally be processed and transformed into capture files (i.e., *tcpdump* format) or into samples for statistical analysis.

With the goal of preparing both packet and flow-level datasets, we capture both packet and NetFlow traffic from different locations in the TUIDS testbed. The capturing period started at 08:00:05am on Monday February 21, 2011, and continuously ran for an exact duration of 7 days, ending at 08:00:05am on Sunday February 27. Attacks were executed during this period for the TUIDS intrusion and the coordinated scan datasets. DDoS traffic was also collected for the same amount of time but during October 2012 with several variations of real-time DDoS attacks. Figure 3.8 illustrates the protocol composition and the average throughput during the last hour of data capture for the TUIDS intrusion dataset seen in our lab.

We use a tool known as Lossless Gigabit Remote Packet Capture with Linux (Gulp[13]) for capturing packet level traffic in a mirror port as shown in the TUIDS testbed architecture. Gulp reads packets directly from the network card and writes to the disk at a high rate of packet capture without dropping packets. For low-rate packets, Gulp flushes the ring buffer if it has not written anything in the last second. Gulp writes into even block boundaries for excellent writing performance when the data rate increases. It stops filling the ring buffer after receiving an interrupt, but it would write into the disk whatever remains in the ring buffer. Example commands of Gulp are given below.

Command: **gulp -i eth1 > pcapfile1**
Meaning: It captures the raw traffic from default ethernet and storing into pcapfile1.

Command: **gulp -i eth1 -d > pcapfile1**
Meaning: Capture and GRE-decapsulate an ERSPAN feed and save the result to disk.

Command: **gulp -i eth1 -d | tcpdump -r - -s0 -w pcapfile1**
Meaning: Capture, decapsulate, and then filter with tcpdump before saving.

[13] http://staff.washington.edu/corey/gulp/

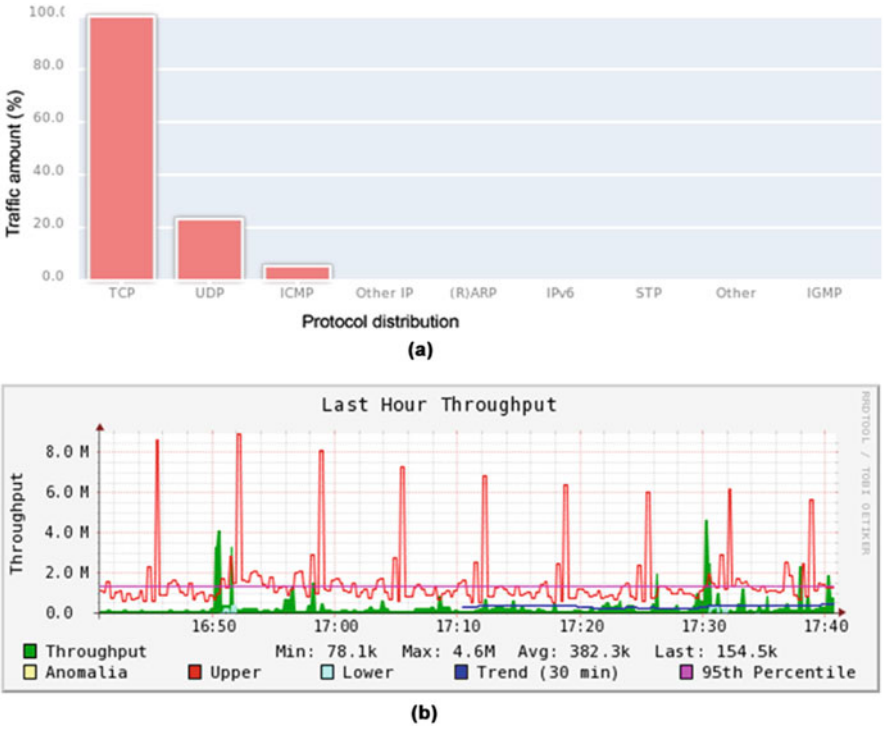


Fig. 3.8 (**a**) Composition of protocols and (**b**) Average throughput during last hour of data capture for the TUIDS intrusion dataset seen in our lab's traffic

Command: **gulp -i eth1 -d | taskset -c 2 tcpdump -r - -s0 -w pcapfile1.**
Meaning: If you have more than 2 CPUs, run tcpdump and gulp on different ones.

Command: **gulp -i eth1 > pcapfile1; gulp -d -i - < pcapfile1 > pcapfile2**
Meaning: Capture everything to disk; then decapsulate offline.

Command: **gulp -i eth1 -d | /usr/sbin/wireshark -i - -k**
Meaning: Capture, decapsulate, and feed into wireshark.

Command: **gulp -i eth1 -C 10 -W 10 -o pcapdir**
Meaning: Capture to 1000MB files, keeping just the most recent 10 (files).

In the last few years, NetFlow has become the most popular approach for IP network monitoring, since it helps cope with the scalability issues introduced by increasing network speeds. Now major vendors offer flow-enabled devices. An example is a Cisco router with NetFlow. A NetFlow is a stream of packets that arrives on a source interface with the key values shown in Fig. 3.9. A key is an identified value for a field within the packet. Cisco routers have NetFlow features

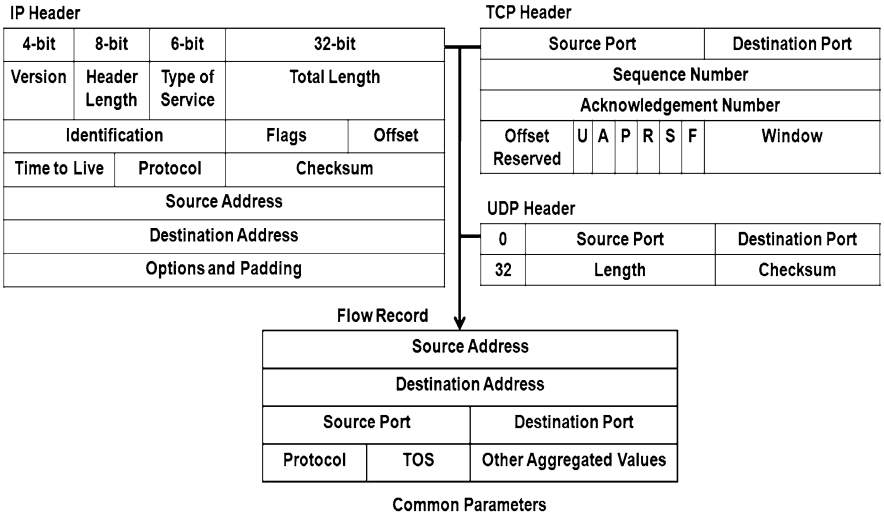


Fig. 3.9 Common NetFlow parameters

that can be enabled to generate NetFlow records. The principle of NetFlow is as follows. When the router receives a packet, its NetFlow module scans the source IP address, the destination IP address, the source port number, the destination port number, the protocol type, the type of service (ToS) bit in IP header, and the input or output interface number on the router of the IP packet, to judge whether it belongs to a NetFlow record that already exists in the cache. If so, it updates the NetFlow record; otherwise, a new NetFlow record is generated in the cache. The expired NetFlow records in the cache are exported periodically to a destination IP address using a UDP port.

For capturing NetFlow traffic, we need a NetFlow collector that can listen to a specific UDP port to collect traffic. The NetFlow collector captures exported traffic from multiple routers and periodically stores it in summarized or aggregated format into a round robin database (RRD). The following tools are used to capture and visualize the NetFlow traffic.

(a) *NFDUMP*: This tool captures and displays NetFlow traffic. All versions of nfdump support NetFlow v5, v7, and v9. nfcapd is a NetFlow capture daemon that reads the NetFlow data from the routers and stores the data into files periodically. It automatically rotates files every n minutes (by default it is 5 min). We need one nfcapd process for each NetFlow stream. Nfdump reads the NetFlow data from the files stored by nfcapd. The syntax is similar to that of tcpdump. Nfdump displays NetFlow data and can create top N statistics for flows based on the parameters selected. The main goal is to analyze NetFlow data from the past as well as to track interesting traffic patterns continuously from high-speed networks. The amount of time from the past is limited only by the disk space available for all NetFlow data. The internal architecture of nfdump is given in Fig. 3.10.

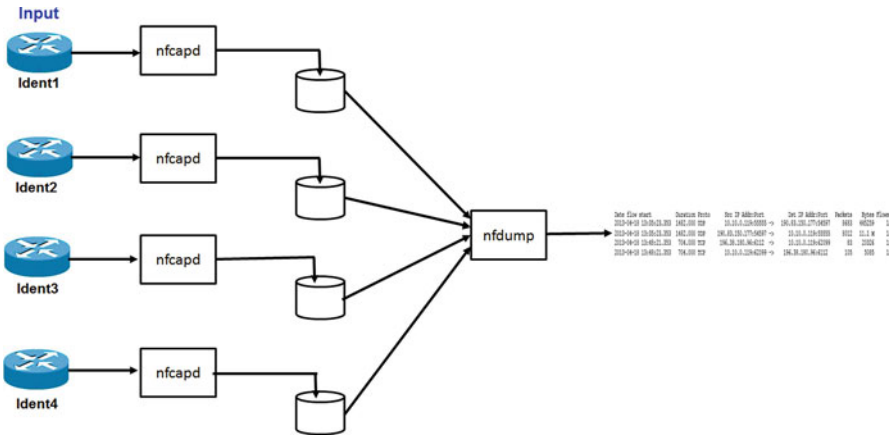


Fig. 3.10 Internal architecture of nfdump

Nfdump has four fixed output formats: *raw*, *line*, *long*, and *extended*. In addition, the user may specify any desired output format by customizing it. The default format is line, unless specified. The raw format displays each record in multiple lines and prints any available information in the traffic record. Example commands for nfdump are given below.

Command: **nfdump -r /and/dir/nfcapd.200407110845 -c 100 'tcp and (src ip 172.16.17.18 or dst ip 172.16.17.19)'**
Meaning: Stores the first 100 NetFlow records that matches the given filter.

Command: **nfdump -r /and/dir/nfcapd.200407110845 -S -n 20**
Meaning: It generates top 20 statistics of NetFlow records.

Command: **nfdump -M /to/and/dir1:dir2 -R**
nfcapd.200407110845:
nfcapd.200407110945 -S -n 20
Meaning: It generates the top 20 statistics from 08:45 to 09:45 from 3 sources.

(b) *NFSEN*: nfsen is a graphical Web-based front end tool for visualization of NetFlow traffic. nfsen facilitates the visualization of several traffic statistics, e.g., flow-wise statistics for various features, navigation through the NetFlow traffic, processes within a time span, and continuous profiles. It can also add own plugins to process NetFlow traffic in a customized manner at a regular time interval (Fig. 3.11).

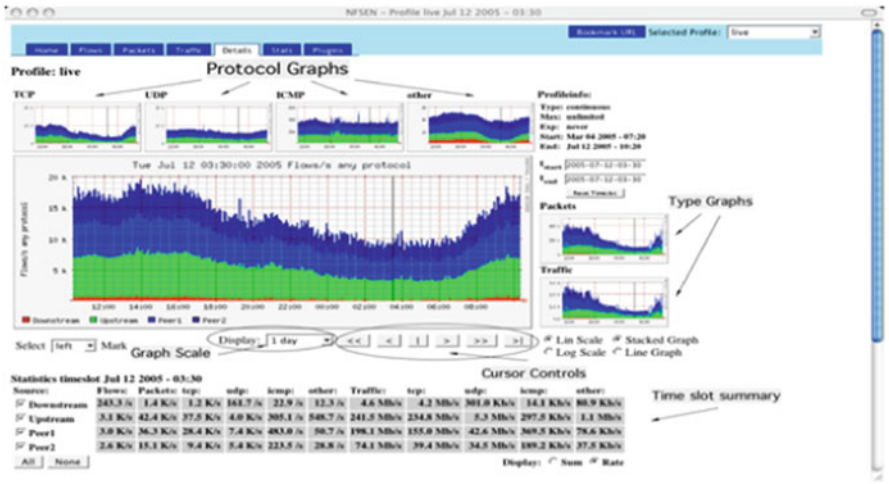


Fig. 3.11 NFSEN web interface

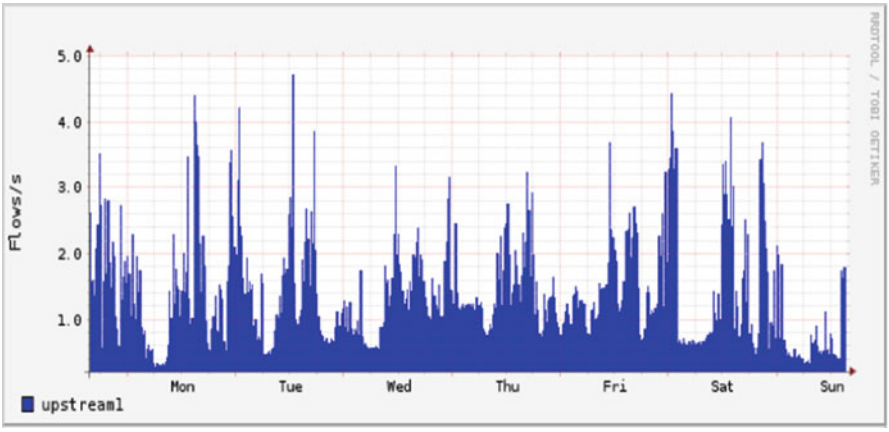


Fig. 3.12 Number of flows per second in TUIDS intrusion datasets during the capture period

Normal traffic is captured by restricting it to the internal networks, where 80% of the hosts are connected to the router, including wireless networks. We assume that normal traffic follows the normal probability distribution. Attack traffic is captured as we launch various attacks in the testbed for a week. For DDoS attacks, we used packet-craft[14] to generate customized packets. Figures 3.12 and 3.13 show the number of flows per second and also the protocol-wise distribution of flows during the capturing period, respectively.

[14] http://www.packet-craft.net/

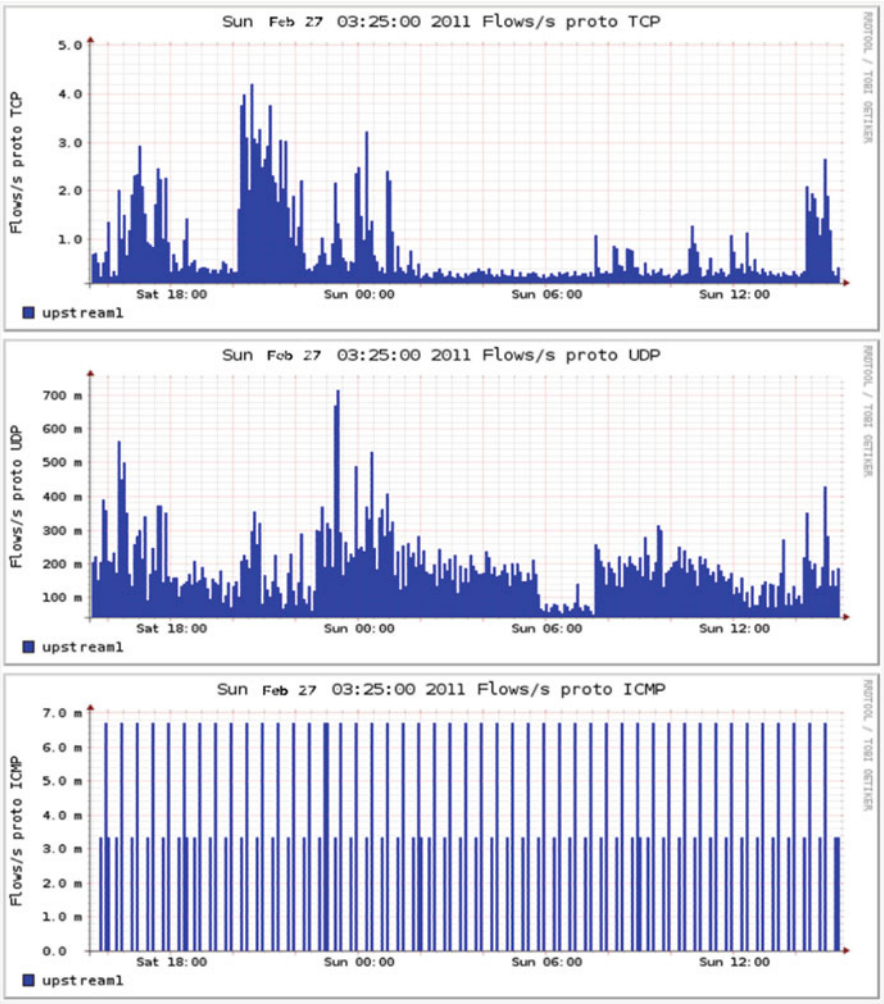


Fig. 3.13 Protocol-wise distribution of flow per second in TUIDS intrusion dataset during the capture period

3.3.5 Feature Extraction

We use wireshark and Java routines for filtering unwanted packets (such as packets with routing protocols, and packets with application layer protocols) as well as irrelevant information from the captured packets. Finally, we retrieve all relevant information from each packet using Java routines and store it in comma-separated form in a text file. The details of parameters identified for packet level data are shown in Table 3.9.

Table 3.9 Parameters identified for packet level data

Sl. No.	Parameter name	Description
1	Time	Time since occurrence of first frame
2	Frame-no	Frame number
3	Frame-len	Length of a frame
4	Capture-len	Capture length
5	TTL	Time to live
6	Protocol	Protocols (such as, TCP, UDP, ICMP etc.)
7	Src-ip	Source IP address
8	Dst-ip	Destination IP address
9	Src-port	Source port
10	Dst-port	Destination port
11	Len	Data length
12	Seq-no	Sequence number
13	Header-len	Header length
14	CWR	Congestion window record
15	ECN	Explicit congestion notification
16	URG	Urgent TCP flag
17	ACK	Acknowledgment flag
18	PSH	Push flag
19	RST	Reset flag
20	SYN	TCP syn flag
21	FIN	TCP fin flag
22	Win size	Window size
23	MSS	Maximum segment size

We developed several *C* routines and used them for filtering NetFlow data and for extracting features from the captured data. A detailed list of parameters identified for flow-level data is given in Table 3.10.

We capture raw traffic (both packet and flows), preprocess, and extract various types of features. We introduce a framework for fast, distributed feature extraction from raw network traffic, correlation computation, and data labeling, as shown in Fig. 3.14. We extract four types of features, *basic*, *content*-based, *time*-based, and *connection*-based, from the raw network traffic. We use $T = 5$ seconds as the time window for extraction of both time-based and connection-based traffic features. S_1 and S_2 are servers used for preprocessing, attack labeling, and profile generation. WS_1 and WS_2 are high-end workstations used for basic feature extraction and merging packet and NetFlow traffic. $N_1, N_2, \cdots N_6$ are independent nodes used for protocol-specific feature extraction. The lists of extracted features at both packet and flow levels for the intrusion datasets are presented in Table 3.11 and Table 3.12, respectively. The list of features available in the KDDcup99 intrusion dataset is also shown in Table 3.13.

Table 3.10 Parameters identified for flow-level data

Sl. No.	Parameter name	Description
1	Flow-start	Starting of flow
2	Duration	Total life time of a flow
3	Proto	Protocol, i.e., TCP, UDP, ICMP, etc.
3	Src-ip	Source IP address
4	Src-port	Source port
5	Dest-ip	Destination IP address
6	Dest-port	Destination port
7	Flags	TCP flags
8	ToS	Type of Service
9	Packets	Packets per flow
10	Bytes	Bytes per flow
11	Pps	Packet per second
12	Bps	Bit per second
13	Bpp	Byte per packet

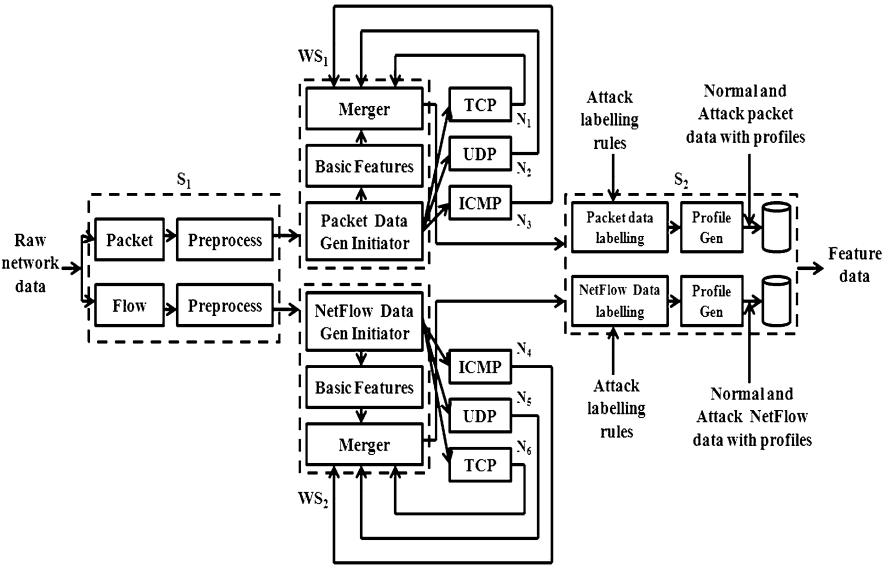


Fig. 3.14 Distributed, protocol-specific feature extraction, correlation, and labeling framework

3.3.6 Data Processing and Labeling

As mentioned in the previous section, both packet and flow-level traffic features are extracted separately within a time interval when features are extracted. So, it is important to correlate each feature (i.e., basic, content-based, time-based, and connection-based) to a time interval. Once correlation is performed for both packet

Table 3.11 List of packet level features in TUIDS intrusion dataset

Label/feature name	Type	Description
Basic features		
1. Duration	C	Length (number of seconds) of the connection
2. Protocol-type	D	Type of protocol, e.g., tcp, udp, etc.
3. Src-ip	C	Source host IP address
4. Dest-ip	C	Destination IP address
5. Src-port	C	Source host port number
6. Dest-port	C	Destination host port number
7. Service	D	Network service at the destination, e.g., http, telnet, etc.
8. num-bytes-src-dst	C	The number of data bytes flowing from source to destination
9. num-bytes-dst-src	C	The number of data bytes flowing from destination to source
10. Fr-no	C	Frame number
11. Fr-len	C	Frame length
12. Cap-len	C	Captured frame length
13. Head-len	C	Header length of the packet
14. Frag-off	D	Fragment offset: "1" for the second packet overwrite everything; "0" otherwise
15. TTL	C	Time to live: "0" discards the packet
16. Seq-no	C	Sequence number of the packet
17. CWR	D	Congestion window record
18. ECN	D	Explicit congestion notification
19. URG	D	Urgent TCP flag
20. ACK	D	Acknowledgment flag value
21. PSH	D	Push TCP flag
22. RST	D	Reset TCP flag
23. SYN	D	Syn TCP flag
24. FIN	D	Fin TCP flag
25. Land	D	1 if connection is from/to the same host/port; 0 otherwise
Content-based features		
26. Mss-src-dest-requested	C	Maximum segment size from source to destination requested
27. Mss-dest-src-requested	C	Maximum segment size from destination to source requested
28. Ttt-len-src-dst	C	Time to live length from source to destination
29. Ttt-len-dst-src	C	Time to live length from destination to source
30. Conn-status	C	Status of the connection (e.g., "1" for complete, "0" for reset)

(continued)

Table 3.11 (continued)

Label/feature name	Type	Description
Time-based features		
31. count-fr-dest	C	Number of frames received by unique destinations in the last *T* seconds from the same source
32. count-fr-src	C	Number of frames received from unique sources in the last *T* seconds from the same destination
33. count-serv-src	C	Number of frames from the source to the same destination port in the last *T* seconds
34. count-serv-dest	C	Number of frames from destination to the same source port in the last *T* seconds
35. num-pushed-src-dst	C	The number of pushed packets flowing from source to destination
36. num-pushed-dst-src	C	The number of pushed packets flowing from destination to source
37. num-SYN-FIN-src-dst	C	The number of SYN/FIN packets flowing from source to destination
38. num-SYN-FIN-dst-src	C	The number of SYN/FIN packets flowing from destination to source
39. num-FIN-src-dst	C	The number of FIN packets flowing from source to destination
40. num-FIN-dst-src	C	The number of FIN packets flowing from destination to source
Connection-based features		
41. count-dest-conn	C	Number of frames to unique destinations in the last N packets from the same source
42. count-src-conn	C	Number of frames from unique sources in the last N packets to the same destination
43. count-serv-srcconn	C	Number of frames from the source to the same destination port in the last N packets
44. count-serv-destconn	C	Number of frames from the destination to the same source port in the last N packets
45. num-packets-src-dst	C	The number of packets flowing from source to destination
46. num-packets-dst-src	C	The number of packets flowing from destination to source
47. num-acks-src-dst	C	The number of acknowledgement packets flowing from source to destination
48. num-acks-dst-src	C	The number of acknowledgement packets flowing from destination to source
49. num-retransmit-src-dst	C	The number of retransmitted packets flowing from source to destination
50. num-retransmit-dst-src	C	The number of retransmitted packets flowing from destination to source

C Continuous, *D* Discrete

Table 3.12 List of flow-level features in TUIDS intrusion dataset

Label/feature name	Type	Description
Basic features		
1. Duration	C	Length (number of seconds) of the flow
2. Protocol-type	D	Type of protocol, e.g., TCP, UDP, ICMP
3. Src-ip	C	Source host IP address
4. Dest-ip	C	Destination IP address
5. Src-port	C	Source host port number
6. Dest-port	C	Destination host port number
7. ToS	D	Type of service
8. URG	D	TCP urgent flag
9. ACK	D	TCP acknowledgment flag
10. PSH	D	TCP push flag
11. RST	D	TCP reset flag
12. SYN	D	TCP SYN flag
13. FIN	D	TCP FIN flag
14. Src-bytes	C	Number of data bytes transfered from source to destination
15. Dest-bytes	C	Number of data bytes transfered from destination to source
16. Land	D	1 if connection is from/to the same host/port; 0 otherwise
Time-based features		
17. count-dest	C	Number of flows to unique destination IPs in the last T seconds from the same source
18. count-src	C	Number of flows from unique source IPs in the last T seconds to the same destination
19. count-serv-src	C	Number of flows from the source to the same destination port in the last T seconds
20. count-serv-dest	C	Number of flows from the destination to the same source port in the last T seconds
Connection-based features		
21. count-dest-conn	C	Number of flows to unique destination IPs in the last N flows from the same source
22. count-src-conn	C	Number of flows from unique source IPs in the last N flows to the same destination
24. count-serv-srcconn	C	Number of flows from the source IP to the same destination port in the last N flows
25. count-serv-destconn	C	Number of flows to the destination IP to the same source port in the last N flows

C Continuous, *D* Discrete

and flow-level traffic, labeling of each feature data as normal or anomalous is important. The labeling process enriches the feature data with information such as (i) the type and structure of malicious or anomalous data and (ii) dependencies among different isolated malicious activities. The correlation and labeling of each feature traffic as normal or anomalous is made using Algorithm 1. However, both

Table 3.13 List of features in the KDDcup99 intrusion dataset

Label/feature name	Type	Description
Basic features		
1. Duration	C	Length (number of seconds) of the connection
2. Protocol-type	D	Type of protocol, e.g., tcp, udp, etc.
3. Service	D	Network service at the destination, e.g., http, telnet, etc.
4. Flag	D	Normal or error status of the connection
5. Src-bytes	C	Number of data bytes from source to destination
6. Dst-bytes	C	Number of data bytes from destination to source
7. Land	D	1 if connection is from/to the same host/port; 0 otherwise
8. Wrong-fragment	C	Number of "wrong" fragments
9. Urgen	C	Number of urgent packets
Content-based features		
10. Hot	C	Number of "hot" indicators (hot: number of directory accesses, create and execute program)
11. Num-failed-logins	C	Number of failed login attempts
12. Logged-in	D	1 if successfully logged-in; 0 otherwise
13. Num-compromised	C	Number of "compromised" conditions (compromised condition: number of file/path not found errors and jumping commands)
14. Root-shell	D	1 if root-shell is obtained; 0 otherwise
15. Su-attempted	D	1 if "su root" command attempted; 0 otherwise
16. Num-root	C	Number of "root" accesses
17. Num-file-creations	C	Number of file creation operations
18. Num-shells	C	Number of shell prompts
19. Num-access-files	C	Number of operations on access control files
20. Num-outbound-cmds	C	Number of outbound commands in an ftp session
21. Is-host-login	D	1 if login belongs to the "hot" list; 0 otherwise
22. Is-guest-login	D	1 if the login is a "guest" login; 0 otherwise
Time-based features		
23. Count	C	Number of connections to the same host as the current connection in the past 2 seconds
24. Srv-count	C	Number of connections to the same service as the current connection in the past 2 seconds (same-host connections)
25. Serror-rate	C	% of connections that have "SYN" errors (same-host connections)
26. Srv-serror-rate	C	% of connections that have "SYN" errors (same-service connections)
27. Rerror-rate	C	% of connections that have "REJ" errors (same-host connections)
28. Srv-rerror-rate	C	% of connections that have "REJ" errors (same-service connections)
29. Same-srv-rate	C	% of connections to the same service (same-host connections)
30. Diff-srv-rate	C	% of connections to different services (same-host connections)
31. Srv-diff-host-rate	C	% of connections to different hosts (same-service connections)

Table 3.13 (continued)

Label/feature name	Type	Description
Connection-based features		
32. Dst-host-count	C	Count of destination hosts
33. Dst-host-srv-count	C	Srv_count for destination host
34. Dst-host-same-srv-rate	C	Same_srv_rate for destination host
35. Dst-host-diff-srv-rate	C	Diff_srv_rate for destination host
36. Dst-host-same-src-port-rate	C	Same_src_port_rate for destination host
37. Dst-host-srv-diff-host-rate	C	Diff_host_rate for destination host
38. Dst-host-serror-rate	C	Serror_rate for destination host
39. Dst-host-srv-serror-rate	C	Srv_serror_rate for destination host
40. Dst-host-rerror-rate	C	Rerror_rate for destination host
41. Dst-host-srv-rerror-rate	C	Srv_rerror_rate for destination host

C Continuous, *D* Discrete

Algorithm 1 : FC and labeling ($\mathbb{F}$)

Require: extracted feature set, $\mathbb{F} = \{\alpha_1, \beta_1, \gamma_1, \delta_1\}$
Ensure: correlated and labelled feature data, $\mathbb{X}$
1: initialize X
2: call *FeatureExtraction*(), $F \leftarrow \{\alpha_1, \beta_1, \gamma_1, \delta_1\}$, ▷ the procedure *FeatureExtraction*() extracts the features separately for all cases
3: **for** $i \leftarrow 1$ to $|\mathbb{N}|$ **do** ▷ $\mathbb{N}$ is the total traffic instances
4: **for** $i \leftarrow 1$ to $|\mathbb{F}|$ **do** ▷ $\mathbb{F}$ is the total traffic features
5: **if** ($unique(src.ip \wedge dst.ip)$) **then**
6: store $X[ij] \leftarrow \alpha_{1(ij)}, \beta_{1(ij)}$
7: **end if**
8: **if** $((\mathbb{T} == 5s) \wedge (LnP == 100))$ **then** ▷ $\mathbb{T}$ is the time window, *LnP* is the last *n* packets
9: Store $X[ij] \leftarrow \gamma_{1(ij)}, \delta_{1(ij)}$
10: **end if**
11: **end for**
12: $X[ij] \leftarrow \{normal, attack\}$ ▷ label each traffic feature instance based on the duration of the collected traffic
13: **end for**

normal and anomalous traffic are collected separately in several sessions within a week. We remove normal traffic from anomalous traces as much as possible.

The overall traffic composition with protocol distribution in the generated datasets is summarized in Table 3.14. The traffic includes the TUIDS intrusion dataset, the TUIDS coordinated scan dataset, and the TUIDS DDoS dataset. The final labeled feature datasets for each category with the distribution of normal and attack information are summarized in Table 3.15. All datasets are prepared at both packet and flow levels and are presented in terms of training and testing in Table 3.15.

Table 3.14 TUIDS dataset traffic composition

Protocol	Size (MB)	(%)
(a) Total traffic composition		
IP	66,784.29	99.99
ARP	3.96	0.005
IPv6	0.00	0.00
IPX	0.00	0.00
STP	0.00	0.00
Other	0.00	0.00
(b) TCP/UDP/ICMP traffic composition		
TCP	49,049.29	73.44
UDP	14,940.53	22.37
ICMP	2,798.43	4.19
ICMPv6	0.00	0.00
Other	0.00	0.00

3.3.7 Comparison with Other Public Datasets

Several real network traffic traces are readily available to the research community as reported in Sect. 3.2. Although these traffic traces are invaluable to the research community, most, if not all, fail to satisfy one or more requirements described in Sect. 3.1. This book attempts to resolve the issues seen in other datasets by presenting a systematic approach to generate real-life network intrusion datasets. Table 3.16 summarizes a comparison between the prior datasets and the dataset generated through the application of our systematic approach to fulfill the principal objectives outlined for qualifying datasets.

Most datasets are unlabeled as labeling is laborious and requires a comprehensive search to tag anomalous traffic. Although an IDS helps by reducing the work, there is no guarantee that all anomalous activities are labeled. This has been a major issue with all datasets and one of the reasons behind the post-insertion of attack traffic in the DARPA 1999 dataset, so that anomalous traffic can be labeled in a deterministic manner. Having seen the inconsistencies produced by traffic merging, this chapter has adopted a different approach to provide the same level of deterministic behavior with respect to anomalous traffic by conducting anomalous activity within the capturing period using available network resources. Through the use of logging, all ill-intended activities can be effectively labeled.

The extent and scope of network traffic capture becomes relevant in situations where the information contained in the traces may breach the privacy of individuals or organizations. In order to prevent privacy issues, almost all publicly available datasets remove any identifying information such as payload, protocol, destination, and flags. In addition, the data is anonymized where necessary header information is cropped or flows are just summarized.

Table 3.15 Distribution of normal and attack connection instances in real-life packet and flow-level TUIDS datasets

Connection type	Dataset type			
	Training dataset		Testing dataset	
(a) TUIDS intrusion dataset				
Packet level				
Normal	71,785	58.87%	47,895	55.52%
DoS	42,592	34.93%	30,613	35.49%
Probe	7,550	6.19%	7,757	8.99%
Total	121,927	–	86,265	–
Flow level				
Normal	23,120	43.75%	16,770	41.17%
DoS	21,441	40.57%	14,475	35.54%
Probe	8,282	15.67%	9,480	23.28%
Total	52,843	–	40,725	–
(b) TUIDS coordinated scan dataset				
Packet level				
Normal	65,285	90.14%	41,095	84.95%
Probe	7,140	9.86%	7,283	15.05%
Total	72,425	–	48,378	–
Flow level				
Normal	20,180	73.44%	15,853	65.52%
Probe	7,297	26.56%	8,357	34.52%
Total	27,477	–	24,210	–
(c) TUIDS DDoS dataset				
Packet level				
Normal	46,513	68.62%	44,328	60.50%
Flooding attacks	21,273	31.38%	28,936	39.49%
Total	67,786	–	73,264	–
Flow level				
Normal	27,411	57.67%	28,841	61.38%
Flooding attacks	20,117	42.33%	18,150	38.62%
Total	47,528	–	46,991	–

In addition to anomalous traffic, traces must contain background traffic. Most captured datasets have little control over the anomalous activities included in the traces. However, a major concern with evaluating anomaly-based detection approaches is the requirement that anomalous traffic must be present on a certain scale. Anomalous traffic also tends to become outdated with the introduction of more sophisticated attacks. So, we have generated more up-to-date datasets that reflect the current trends and are tailored to evaluate certain characteristics of detection mechanisms which are unique to themselves.

Table 3.16 Comparison of existing datasets and their characteristics

Dataset	u	v	w	No. of instances	No. of attributes	x	y	z	Some references
Synthetic	No	No	Yes	user dependent	user dependent	Not known	any	user dependent	[1, 2]
KDDcup99	Yes	No	Yes	805,050	41	BCTW	P	C_1	[24, 26, 40, 41]
NSL-KDD	Yes	No	Yes	148,517	41	BCTW	P	C_1	[36]
DARPA 2000	Yes	No	No	Huge	Not known	Raw	Raw	C_2	[30]
DEFCON	No	No	No	Huge	Not known	Raw	P	C_2	[30]
CAIDA	Yes	Yes	No	Huge	Not known	Raw	P	C_1	[30]
LBNL	Yes	Yes	No	Huge	Not known	Raw	P	C_2	[39]
ISCX-UNB	Yes	Yes	Yes	Huge	Not known	Raw	P	A	[30]
KU	Yes	Yes	No	Huge	24	BTW	P	C_1	[32]
TUIDS	Yes	Yes	Yes	Huge	50,24	BCTW	P,F	C_1	[1, 2]

u realistic network configuration, *v* indicates realistic traffic, *w* describes the label information, *x* types of features extracted as basic features (B), content-based features (C), time-based features (T), and window-based features(W), *y* explains the types of data as packet based (P) or flow based (F) or hybrid (H) or others (O), *z* represents the attack category as C_1-all attacks, C_2-denial of service, C_3-probe, C_4-user to root, C_5-remote to local, and *A*-application layer attacks

3.4 Observations and Chapter Summary

Several questions may be raised with respect to what constitutes a perfect dataset when dealing with the dataset generation task. These include qualities of normal, anomalous, or realistic traffic included in the dataset. We provide a path and a template to generate a dataset that simultaneously exhibits the appropriate levels of normality, anomalousness, and realism while avoiding the various weak points of currently available datasets, pointed out earlier. Quantitative measurements can be obtained only when specific methods are applied to the dataset.

The following are the major observations and requirements when generating an unbiased real-life dataset for intrusion detection:

- The dataset should not exhibit any unintended property in both normal and anomalous traffic.
- The dataset should be labeled correctly.
- The dataset should cover all possible current network scenarios.
- The dataset should be entirely non-anonymized.
- In most benchmark datasets, the two basic assumptions described in Sect. 3.1 are valid, but this biasness should be avoided as much as possible.
- Several datasets lack traffic features, although it is important to extract traffic features with their relevancy for a particular attack.

Despite the enormous efforts needed to create unbiased datasets, there will always be deficiencies in any one particular dataset. Therefore, it is very important to generate dynamic datasets which not only reflect the traffic compositions and intrusions types of the time but are also modifiable, extensible, and reproducible. Therefore, new datasets must be generated from time to time for the purpose of analysis, testing, and evaluation of network intrusion detection methods and systems from multiple perspectives.

In this chapter, we have discussed a systematic hands-on approach to generate real-life network intrusion datasets using both packet and flow-level traffic information. Three different categories of datasets have been generated using the TUIDS testbed. They are (i) TUIDS intrusion dataset, (ii) TUIDS coordinated scan dataset, and (iii) TUIDS DDoS dataset. We incorporate maximum number of possible attacks and scenarios during the generation of the datasets in our testbed network. These datasets will immensely help the network security research community to evaluate the performance of newly developed methods for network intrusion detection. Once the dataset is built, it is necessary to use it for validation of different mechanisms for the detection of network attacks. So, we discuss different types of detection mechanisms in the next chapter.

References

1. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: RODD: an effective reference-based outlier detection technique for large datasets. In: Advanced Computing, vol. 133, pp. 76–84. Springer (2011)
2. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Surveying port scans and their detection methodologies. Comp. J. **54**(10), 1565–1581 (2011)
3. CACE Technologies: Winpcap. http://www.winpcap.org
4. CAIDA: The cooperative analysis for internet data analysis. http://www.caida.org (2011)
5. Cemerlic, A., Yang, L., Kizza, J.: Network intrusion detection based on Bayesian networks. In: Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering, SEKE'08, pp. 791–794. KSI, San Francisco (2008)
6. Cole, E.: Hackers Beware: Defending Your Network from the Wiley Hacker. New Riders Publishing, Thousand Oaks (2001)
7. Dainotti, A., Pescape, A.: Plab: a packet capture and analysis architecture (2004). Http://www.grid.unina.it/software/ITG/D-ITGpublications/TR-DIS-122004.pdf
8. Defcon: The Shmoo group. http://cctf.shmoo.com/ (2011)
9. Delooze, L.: Applying soft-computing techniques to intrusion detection. Ph.D. thesis, Computer Science Department, University of Colorado, Colorado Springs (2005)
10. Denning, D.E.: An intrusion-detection model. IEEE Trans. Softw. Eng. **13**(2), 222–232 (1987)
11. Ghorbani, A.A., Lu, W., Tavallaee, M.: Network Intrusion Detection and Prevention: Concepts and Techniques. Advances in Information Security. Springer, US (2009)
12. Gogoi, P., Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Packet and flow-based network intrusion dataset. In: Proc. of the 5th International Conference on Contemporary Computing, vol. LNCS-CCIS 306, pp. 322–334. Springer (2012)
13. Jacobson, V., Leres, C., McCanne, S.: tcpdump. URL ftp://ftp.ee.lbl.gov/tcpdump.tar.gz
14. KDDcup99: Knowledge discovery in databases DARPA archive. http://www.kdd.ics.uci.edu/databases/kddcup99/task.html (1999)

15. Kendall, K.: A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, MIT (1999)
16. Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., Srivastava, J.: A comparative study of anomaly detection schemes in network intrusion detection. In: Proceedings of the 3rd SIAM International Conference on Data mining. SIAM (2003)
17. LBNL: Lawrence Berkeley National Laboratory and ICSI, LBNL/ICSI Enterprise Tracing Project. http://www.icir.org/enterprise-tracing/ (2005)
18. Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., Wyschogrod, D., Cunningham, R.K., Zissman, M.A.: Evaluating intrusion detection systems: the 1998 DARPA offline intrusion detection evaluation. In: DARPA Information Survivability Conference and Exposition, vol. 2, pp. 12–26 (2000)
19. Mahoney, M.V., Chan, P.K.: An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection. In: Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection, pp. 220–237. Springer (2003)
20. McCanne, S., Jacobson, V.: The BSD packet filter: a new architecture for user level packet capture. In: Proceedings of the Winter 1993 USENIX Conference, pp. 259–269. USENIX Association (1993)
21. McHugh, J.: Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Trans. Inf. Syst. Secur. **3**(4), 262–294 (2000)
22. Mell, P., Hu, V., Lippmann, R., Haines, J., Zissman, M.: An overview of issues in testing intrusion detection systems. http://citeseer.ist.psu.edu/621355.html (2003)
23. MIT Lincoln Lab, Information Systems Technology Group: DARPA intrusion detection data sets. http://www.ll.mit.edu/mission/communications/ist/\corpora/ideval/data/2000data.html (2000)
24. Muda, Z., Yassin, W., Sulaiman, M.N., Udzir, N.I.: A K-means and naive-bayes learning approach for better intrusion detection. Inf. Technol. J. **10**(3), 648–655 (2011)
25. NSL-KDD: NSL-KDD data set for network-based intrusion detection systems. http://iscx.cs.unb.ca/NSL-KDD/ (2009)
26. Otey, M.E., Ghoting, A., Parthasarathy, S.: Fast distributed outlier detection in mixed-attribute data sets. Data Min. Knowl. Disc. **12**(2–3), 203–228 (2006)
27. Pang, R., Allman, M., Bennett, M., Lee, J., Paxson, V., Tierney, B.: A first look at modern enterprise traffic. In: Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement, pp. 2–2. USENIX Association, Berkeley (2005)
28. Pang, R., Allman, M., Paxson, V., Lee, J.: The devil and packet trace anonymization. SIGCOMM Comput. Commun. Rev. **36**(1), 29–38 (2006)
29. Portnoy, L., Eskin, E., Stolfo, S.: Intrusion detection with unlabeled data using clustering. In: Proceedings of the ACM CSS Workshop on on Data Mining Applied to Security, Philadelphia, pp. 5–8 (2001)
30. Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A.: Towards developing a systematic approach to generate benchmark datasets for intrusion detection. Comput. Secur. **31**(3), 357–374 (2012)
31. Song, J., Takakura, H., Okabe, Y.: Description of Kyoto University Benchmark Data. http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v3.pdf (2006)
32. Song, J., Takakura, H., Okabe, Y., Nakao, K.: Toward a more practical unsupervised anomaly detection system. Inf. Sci. **231** (2011). http://dx.doi.org/10.1016/j.ins.2011.08.011
33. Sperotto, A., Sadre, R., Vliet, F., Pras, A.: A labeled data set for flow-based intrusion detection. In: Proceedings of the 9th IEEE International Workshop on IP Operations and Management, IPOM '09, pp. 39–50. Springer, Venice (2009)
34. Stolfo, S.J., Fan, W., Lee, W., Prodromidis, A., Chan, P.K.: Cost-based modeling for fraud and intrusion detection: results from the JAM project. In: Proceedings of the DARPA Information Survivability Conference and Exposition, vol. 2, pp. 130–144. IEEE CS (2000)
35. symantec.com: Symantec security response. http://securityresponse.symantec.com/avcenter

36. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the 2nd IEEE International Conference on Computational Intelligence for Security and Defense Applications, pp. 53–58. IEEE Press (2009)
37. Thomas, C., Sharma, V., Balakrishnan, N.: Usefulness of DARPA dataset for intrusion detection system evaluation. In: Proceedings of the Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security, 6973. SPIE, Orlando (2008)
38. UNIBS: University of Brescia Dataset. http://www.ing.unibs.it/ntw/tools/traces/ (2009)
39. Xu, J., Shelton, C.R.: Intrusion detection using continuous time Bayesian networks. J. Artif. Intell. Res. **39**, 745–774 (2010)
40. Zhang, C., Zhang, G., Sun, S.: A mixed unsupervised clustering-based intrusion detection model. In: Proceedings of the 3rd International Conference on Genetic and Evolutionary Computing, pp. 426–428. IEEE CS (2009)
41. Zhang, Y.F., Xiong, Z.Y., Wang, X.Q.: Distributed intrusion detection based on clustering. In: Proceedings of the International Conference on Machine Learning and Cybernetics, vol. 4, pp. 2379–2383 (2005)

Chapter 4
Network Traffic Anomaly Detection Techniques and Systems

To develop a network traffic anomaly detection technique and system, it is indeed necessary to know the basic properties of network-wide traffic. This chapter starts with a discussion of the basic properties of network-wide traffic with an example. This chapter is organized into six major sections to describe different network anomaly detection techniques and systems. They are statistical techniques and systems, classification-based techniques and systems, clustering and outlier-based techniques and systems, soft computing-based techniques and systems, knowledge-based techniques and systems, and techniques and systems based on combination learners. Finally, it presents the strengths and weaknesses of each category of detection techniques and systems with a detailed comparison.

4.1 Network-Wide Traffic: An Overview

Due to increasing use of cloud-based services, it is indeed necessary to protect an enterprise network from large-scale network anomalies. Detection of exceptional patterns, i.e., anomalies is really an interesting problem that needs to be addressed in an enterprise network. Network-wide traffic anomalies disrupt the transmission of legitimate traffic over the network or Internet. Network-wide traffic is voluminous, high dimensional, and noisy. Therefore, it is difficult to detect anomalies in such voluminous data in real-time. Within a short time interval, the traffic may change its characteristics, this being a crucial property of network-wide traffic. An example of network-wide traffic is given in Fig. 4.1.

M.H. Bhuyan et al., *Network Traffic Anomaly Detection and Prevention*,
Computer Communications and Networks, DOI 10.1007/978-3-319-65188-0_4

0.272506000,7362,74,74,20,0,64,TCP,115.251.222.63,172.16.15.13,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,14040,14040,14102,53749,14040,14040,14102,13489,6960,0,0,0,278400,0,6959,0,0,0,0,0,0,0,0,0,0,1,syndrop
0.272513000,7363,60,60,20,24,64,TCP,115.251.222.63,172.16.15.13,16189,24939,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,14040,14040,13978,7019,14040,14040,13978,7019,6960,0,0,0,278400,0,6959,0,0,0,0,0,0,0,0,0,1,syndrop
0.272519000,7364,60,60,20,24,64,TCP,115.251.222.63,172.16.15.13,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,14040,14040,14102,53749,14040,14040,14102,13489,6960,0,0,0,278400,0,6959,0,0,0,0,0,0,0,0,0,1,syndrop
0.272528000,7365,74,74,20,0,63,TCP,115.251.222.63,172.16.15.13,16189,24939,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,14040,14040,13978,7019,14040,14040,13978,7019,6960,0,0,0,278400,0,6959,0,0,0,0,0,0,0,0,0,1,syndrop
0.442154000,12351,70,70,20,0,63,UDP,88.115.73.45,172.16.15.13,25247,3895,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,11824,11824,11812,5971,3448,3448,3440,1783,5912,0,-1,-1,-1,-1,212832,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,teardrop
0.442162000,12352,70,70,20,0,63,UDP,88.115.73.45,172.16.15.13,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,11824,11824,11815,8378,3448,3448,3444,2100,5912,0,-1,-1,-1,-1,212832,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,teardrop
0.442169000,12353,60,60,20,24,63,UDP,88.115.73.45,172.16.15.13,25247,3895,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,11824,11824,11812,5971,3448,3448,3440,1783,5912,0,-1,-1,-1,-1,212832,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,teardrop
0.442176000,12354,60,60,20,24,63,UDP,88.115.73.45,172.16.15.13,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,11824,11824,11815,8378,3448,3448,3444,2100,5912,0,-1,-1,-1,-1,212832,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,teardrop
0.442321000,12355,70,70,20,0,64,UDP,88.115.73.45,172.16.15.12,25247,3895,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,11803,11803,11812,5961,3436,3436,3440,1777,5902,0,-1,-1,-1,-1,212472,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,teardrop
0.455833000,68598,60,60,20,65528,255,UDP,42.19.101.238,172.16.15.13,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,2,2,2,41401,2,2,2,11719,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,oshare
0.455840000,68599,60,60,20,65528,255,UDP,34.149.39.84,172.16.15.13,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,2,2,2,41401,2,2,2,11719,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,oshare
0.455847000,68600,60,60,20,65528,255,UDP,34.149.39.84,172.16.15.13,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,2,2,2,41401,2,2,2,11719,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,oshare
0.455854000,68601,60,60,20,65528,255,UDP,88.115.113.3,172.16.15.12,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,2,2,2,38114,2,2,2,8181,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,oshare
0.455861000,68602,60,60,20,65528,255,UDP,88.115.113.3,172.16.15.12,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,2,2,2,38114,2,2,2,8181,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,oshare
189.961277000,20236,1514,1514,20,0,62,TCP,202.141.129.30,172.16.12.13,3128,43284,144037,0,0,0,1,0,0,0,0,28992,26115,26115,3113,26115,9188,9188,1075,9188,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,normal
189.961283000,20237,1514,1514,20,0,62,TCP,202.141.129.30,172.16.12.13,3128,43284,142589,0,0,0,1,0,0,0,0,28992,26115,26115,3113,26115,9188,9188,1075,9188,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,normal
189.961291000,20238,66,66,20,0,64,TCP,172.16.12.13,202.141.129.30,43285,3128,9048,0,0,0,1,0,0,0,0,64128,16855,16855,16855,848,5608,5608,5608,408,24,24,12,8,2624,6484,9,12,4,4,4,4,4,4,1460,1460,656,1621,0,normal
189.961293000,20239,66,66,20,0,64,TCP,172.16.12.13,202.141.129.30,43285,3128,9048,0,0,0,1,0,0,0,0,64128,16855,16855,16855,848,5608,5608,5608,408,24,24,12,8,2624,6484,9,12,4,4,4,4,4,4,1460,1460,656,1621,0,normal
189.961296000,20240,1514,1514,20,0,62,TCP,202.141.129.30,172.16.12.13,3128,43284,144037,0,0,0,1,0,0,0,0,28992,26115,26115,3113,26115,9188,9188,1075,9188,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,normal
189.961301000,20241,1266,1266,20,0,62,TCP,202.141.129.30,172.16.12.13,3128,43284,145485,0,0,0,1,1,0,0,0,28992,26115,26115,3113,26115,9188,9188,1075,9188,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,normal
189.961305000,20242,1514,1514,20,0,62,TCP,202.141.129.30,172.16.12.13,3128,43284,144037,0,0,0,1,0,0,0,0,28992,26115,26115,3113,26115,9188,9188,1075,9188,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,normal
189.961348000,20243,1266,1266,20,0,62,TCP,202.141.129.30,172.16.12.13,3128,43284,145485,0,0,0,1,1,0,0,0,28992,26115,26115,3113,26115,9188,9188,1075,9188,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,normal

Fig. 4.1 Network-wide traffic: an example

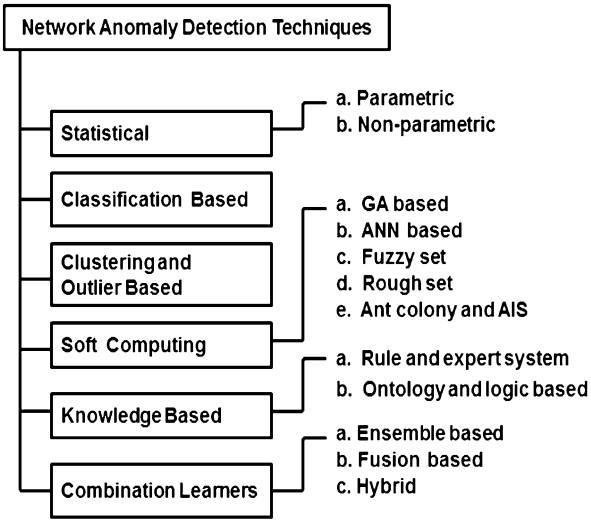


Fig. 4.2 Classification of network anomaly detection techniques (*GA* genetic algorithm, *ANN* artificial neural network, *AIS* artificial immune system)

4.2 Classification of Network Anomaly Detection Techniques and Systems

A classification scheme for network anomaly detection techniques and systems based on the nature of algorithms used is shown in Fig. 4.2. It is not straightforward to come up with appropriate classification for network anomaly detection techniques and systems, primarily because there is substantial overlap among the techniques used in the various classes in any particular scheme we may adopt. We have decided on six distinct classes of techniques and systems. We call them *statistical*, *classification-based*, *clustering and outlier-based*, *soft computing*, *knowledge-based*, and *combination learners*. Most techniques have subclasses as given in Fig. 4.2.

Here, we distinguish between network anomaly detection techniques and systems, although such a distinction is difficult to make sometimes. A network intrusion detection system (NIDS) usually integrates a network intrusion detection technique within an architecture that comprises other associated subsystems to build a standalone practical system that can perform the entire gamut of activities needed for intrusion detection. We present several NIDSs with their architectures and components as we discuss various anomaly detection categories.

4.3 Statistical Techniques and Systems

Statistically speaking, an anomaly is an observation which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed [7]. Normally, statistical techniques fit a statistical model (usually for normal behavior) to the given data and then apply a statistical inference test to determine if an unseen instance belongs to this model. Instances that have a low probability to be generated from the learnt model based on the applied test statistic are declared as anomalies.

Of the many statistical techniques and NIDSs [27, 33, 49, 91, 96, 97, 139, 153, 154, 161], only a few are described below in brief.

4.3.1 Statistical Techniques

Statistical techniques are usually designed based on network traffic distribution, assuming the network traffic follows a certain distribution. The simplest way to build a statistical model is to compute the parameters of a probability density function for each known class of network traffic and then test an unknown sample to determine which class it belongs to [98]. When generating a probability density function, it usually builds two different profiles, one for normal and the other for attack traffic. It checks whether input traffic belongs to the existing classes or not.

In practice, to estimate probability density function, there are two main techniques, parametric and nonparametric [41]. Parametric techniques assume knowledge of the underlying distribution and estimate the parameters from the given data [49]. The Gaussian distribution is usually assumed for the normal traffic data. Parametric techniques estimate certain parameters to fit into the existing distribution. However, distributions of major real-world data are not similar. So, such techniques are not used much in reality. Nonparametric techniques do not generally assume knowledge of the underlying distribution [41]. The density function is derived based on the network traffic distribution as well as parameter estimation. As a result, such techniques provide more flexibility than parametric techniques. A few parametric and nonparametric techniques for network traffic anomaly detection are described below.

Bayesian networks [54] are capable of detecting anomalies in a multi-class setting. Several variants of the basic technique have been proposed for network intrusion detection and for anomaly detection in text data [28]. The basic technique assumes independence among different attributes. Several variations that capture the conditional dependencies among different attributes using more complex Bayesian networks have also been proposed. For example, Kruegel et al. [81] introduce an event classification-based intrusion detection scheme using Bayesian networks. The Bayesian decision process improves detection decision to significantly reduce the rate of false alarms. Manikopoulos and Papavassiliou [97] introduce a hierarchical multitier multi-window statistical anomaly detection system to operate automatically, adaptively, and proactively. It can work with both wired and wireless ad hoc networks. This system uses statistical modeling and neural network classification to detect network anomalies and faults. The system achieves high detection rate along with low misclassification rate when the anomaly traffic intensity is at 5% of the background traffic, but the detection rate is lower at lower attack intensity levels such as 1% and 2%.

Association rule mining [3], conceptually a simple method based on counting of co-occurrences of items in transactions databases, has been used for one-class anomaly detection by generating rules from the data in an unsupervised fashion. The most difficult and dominating part of an association rule discovery algorithm is to find the itemsets that have strong support. Mahoney and Chan [96] present an algorithm known as LERAD that learns rules for finding rare events in time-series data with long-range dependencies and finds anomalies in network packets over TCP sessions. LERAD uses an a priori-like algorithm [3] that finds conditional rules over nominal attributes in a time series, e.g., a sequence of inbound client packets. The antecedent of a created rule is a conjunction of equalities, and the consequent is a set of allowed values, e.g., *if port=80 and word3=HTTP/1.0, then word1=GET or POST.* A value is allowed if it is observed in at least one training instance satisfying the antecedent. The idea is to identify rare anomalous events: those which have not occurred for a long time and which have high anomaly score. LERAD is a two-pass algorithm. In the first pass, a candidate rule set is generated from a random sample of training data comprised of attack-free network traffic. In the second pass, rules are trained by obtaining the set of allowed values for each antecedent. Li et al. [88] introduce a multivariate probabilistic calibration model for network anomaly detection and localization. It uses t-distribution to model the normal traffic. The algorithm's effectiveness demonstrated in terms of theoretical and experimental analysis. It works well even in the presence of high noise and is sensitive to parameters.

Song et al. [139] propose a conditional anomaly detection method for computing differences among attributes and present three different expectation-maximization algorithms for learning the model. They assume that the data attributes are partitioned into *indicator* attributes and *environmental* attributes based on the decision taken by the user regarding which attributes indicate an anomaly. The method learns the typical indicator attribute values and observes subsequent data points and labels them as anomalous or not, based on the degree the indicator attribute values differ

from the usual indicator attribute values. However, if the indicator attribute values are not conditioned on environmental attributes values, the indicator attributes are ignored effectively. The precision and recall of this method are greater than 90%.

Lu and Ghorbani [91] present a network signal modeling technique for anomaly detection by combining wavelet approximation and system identification theory. They define and generate 15 relevant traffic features as input signals to the system and model daily traffic based on these features. The output of the system is the deviation of the current input signal from the normal or regular signal behavior. Residuals are passed to the IDS engine to take decisions and obtain 95% accuracy in the daily traffic. In addition, a nonparametric adaptive cumulative sum (CUSUM) method for detecting network intrusions is discussed in [161].

4.3.2 Statistical Systems

As mentioned earlier, a NIDS includes one or more intrusion detection techniques that are integrated with other required subsystems necessary to create a practical system. An example of a statistical IDS is HIDE [169]. HIDE is an anomaly-based network intrusion detection system that uses statistical models and neural network classifiers to detect intrusions. HIDE is a distributed system, which consists of several tiers with each tier containing several intrusion detection agents (IDAs). IDAs are IDS components that monitor the activities of a host or a network. The probe layer (i.e., top layer as shown in Fig. 4.3) collects network traffic at a host or in a network, abstracts the traffic into a set of statistical variables to reflect network status, and periodically generates reports to the event preprocessor. The event preprocessor layer receives reports from both the probe and IDAs of lower tiers and converts the information into the format required by the statistical model.

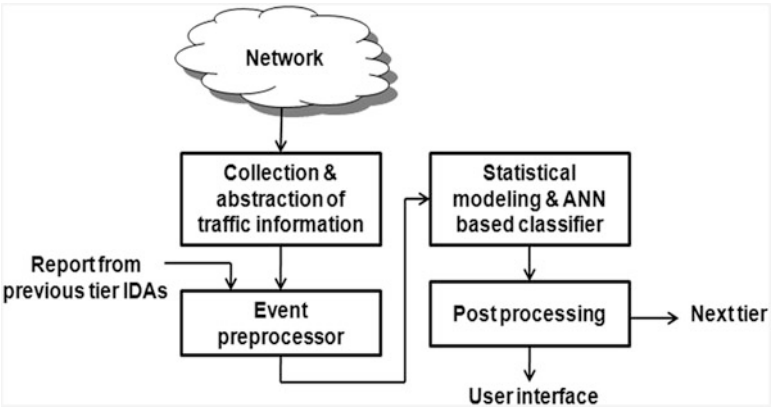


Fig. 4.3 Architecture of HIDE system

The statistical processor maintains a reference model of typical network activities, compares reports from the event preprocessor with the reference models, and forms a stimulus vector to feed into the neural network classifier. The neural network classifier analyzes the stimulus vector from the statistical model to decide whether the network traffic is normal. The post-processor generates reports for the agents at higher tiers. A major attraction of HIDE is its ability to detect UDP flooding attacks even with attack intensity as low as 10% of background traffic.

A payload-based anomaly detector for intrusion detection known as PAYL is proposed in [153]. PAYL attempts to detect the first occurrence of a worm, either at a network system gateway or with an internal network from a rogue device and to prevent its propagation. It employs a language-independent *n*-gram-based statistical model of sampled data streams. In fact, PAYL uses only a 1-gram model (i.e., it looks at the distribution of values contained within a single byte) which requires a linear scan of the data stream and a small 256-element histogram. In other words, for each ASCII character in the range 0–255, it computes its mean frequency as well as the variance and standard deviation. Since payloads (i.e., arriving or departing contents) at different ports differ in length, PAYL computes these statistics for each specific observed payload length for each port open in the system. It first observes many exemplar payloads during the training phase and computes the payload profiles for each port for each payload length. During detection, each incoming payload is scanned, and statistics are computed. The new payload distribution is compared against the model created during training. If there is a significant difference, PAYL concludes that the packet is anomalous and generates an alert. The authors found that this simple approach works surprisingly well.

N@G (Network at Guard) [143] is a hybrid IDS that exploits both misuse and anomaly approaches. N@G has both network and host sensors. Anomaly-based intrusion detection is pursued using the chi-square technique on various network protocol parameters. It has four detection methodologies, viz., traffic capturing, signature-based detection, network access policy violation, and protocol anomaly detection as a part of its network sensor. It includes audit trails, log analysis, statistical analysis, and host access policies as components of the host sensor. The system has a separate IDS server, i.e., a management console to aggregate alerts from the various sensors with a user interface, a middle-tier, and a data management component. It provides real-time protection against malicious changes to network settings on client computers, which includes unsolicited changes to the Windows Hosts file and Windows Messenger service.

Flow-based statistical aggregation scheme (FSAS) [138] is a flow-based statistical IDS. It comprises of two modules: *feature generator* and *flow-based detector*. In the feature generator, the event preprocessor module collects the network traffic of a host or a network. The event handlers generate reports to the flow management module. The flow management module efficiently determines if a packet is part of an existing flow or it should generate a new flow key. By inspecting flow keys, this module aggregates flows together and dynamically updates per-flow accounting measurements. The event time module periodically calls the feature

extraction module to convert the statistics regarding flows into the format required by the statistical model. The neural network classifier classifies the score vectors to prioritize flows with the amount of maliciousness. The higher the maliciousness of a flow, the higher is the possibility of the flow being an attacker.

Advantages of statistical network anomaly detection include the following:

- They do not require prior knowledge of normal activities of the target system. Instead, they have the ability to learn the expected behavior of the system from observations.
- Statistical techniques can provide accurate notification or alarm generation for malicious activities occurring over long periods of time, subject to setting of appropriate thresholding or parameter tuning.
- They analyze the traffic based on the theory of abrupt changes, i.e., they monitor the traffic for a long time and report an alarm if any abrupt change (i.e., significant deviation) occurs.

Drawbacks of the statistical model for network anomaly detection include the following:

- They are susceptible to being trained by an attacker in such a way that the network traffic generated during the attack is considered normal.
- Setting the values of the different parameters or metrics is a difficult task, especially because the balance between false positives and false negatives is an issue. Moreover, a statistical distribution per variable is assumed, but not all behaviors can be modeled using stochastic methods. Furthermore, most schemes rely on the assumption of a quasi-stationary process [112], which is not always realistic.
- It takes a long time to report an anomaly for the first time because the building of the models requires extended time.
- Several hypothesis testing statistics can be applied to detect anomalies. Choosing the best statistic is often not straightforward. In particular, as stated in [154], constructing hypothesis tests for complex distributions that are required to fit high-dimensional datasets is nontrivial.
- Histogram-based techniques are relatively simple to implement, but a key shortcoming of such techniques for multivariate data is that they are not able to capture interactions among the attributes.

A comparison of a few statistical network anomaly detection techniques is given in Table 4.1.

4.4 Classification-Based Techniques and Systems

Classification is the problem of identifying which of a set of categories a new observation belongs to, on the basis of a training set of data containing observations whose category is known. Assuming we have two classes whose instances are shown

Table 4.1 Comparison of statistical network anomaly detection techniques

Author (s)	Year of publication	No. of parameters	w	x	y	Data types	Dataset used	z	Detection method
Eskin [49]	2000	2	O	N	P	Numeric	DARPA99	C_4	Probability model
Manikopoulos and Papavassiliou [97]	2002	3	D	N	P	Numeric	real-life	C_2, C_5	Neural network
Mahoney and Chan [96]	2003	2	C	N	P	–	DARPA99	C_1	LERAD algorithm
Chan et al. [27]	2003	2	C	N	P	Numeric	DARPA99	C_1	Learning rules
Wang and Stolfo [153]	2004	3	C	N	P	Numeric	DARPA99	C_1	Payload-based algorithm
Song et al. [139]	2007	3	C	N	P	Numeric	KDDcup99	Synthetic intrusive pattern	Gaussian mixture model
Chhabra et al. [33]	2008	2	D	N	P	Numeric	Real-time	C_6	FDR method
Lu and Ghorbani [91]	2009	3	C	N	P,F	Numeric	DARPA99	C_1	Wavelet analysis
Wattenberg et al. [154]	2011	4	C	N	P	Numeric	Real-time	C_2	GLRT model
Yu [161]	2012	1	C	N	P	Numeric	Real-time	C_2	Adaptive CUSUM
Li et al. [88]	2015	2	C	N	P	Numeric	Real-time	C_2	RMPCM

w indicates centralized (C) or distributed (D) or others (O), *x* the nature of detection as real time (R) or non-real time (N), *y* characterizes packet-based (P) or flow-based (F) or hybrid (H) or others (O), *z* represents the list of attacks handled: C_1 all attacks, C_2 denial of service, C_3 probe, C_4 user to root, C_5 remote to local, and C_6 anomalous

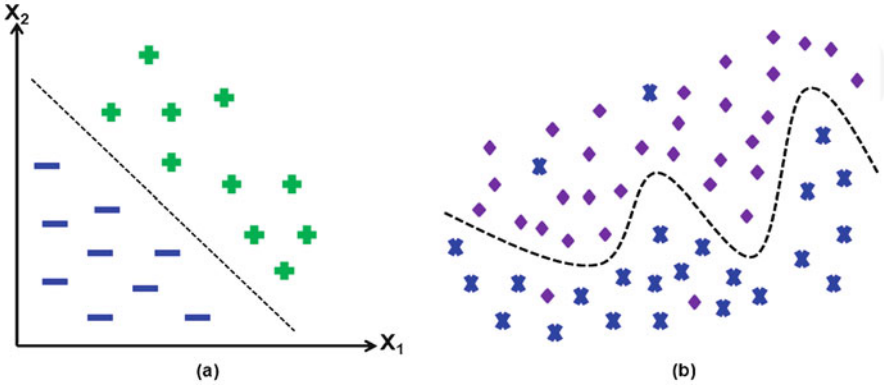


Fig. 4.4 Linear and nonlinear classification in 2-D. (**a**) Linear separation. (**b**) Nonlinear separation

as + and − and each object can be defined in terms of two attributes or features x_1 and x_2, linear classification tries to find a line between the classes as shown in Fig. 4.4a. The classification boundary may be nonlinear as in Fig. 4.4b. In anomaly detection, the data is high dimensional, not just two. The attributes are usually mixed, numeric, and categorial as discussed earlier.

4.4.1 Classification-Based Techniques

Classification techniques are based on establishing an explicit or implicit model that enables categorization of network traffic patterns into several classes [37, 55, 75, 92, 122, 146]. A singular characteristic of these techniques is that they need labeled data to train the behavioral model, a procedure that places high demands on resources [149]. In many cases, the applicability of machine learning principles such as classification coincides with that of statistical techniques, although the former technique is focused on building a model that improves its performance on the basis of previous results [58]. Several classification-based techniques (e.g., k-nearest neighbor, support vector machines, and decision trees) have been applied to network-wide traffic anomaly detection.

Network defenders and researchers have developed many techniques to detect anomalies in network-wide traffic. Port scan is an information gathering technique that searches for hostile open door or port, through which an attacker gains access to computers [39]. Coordinated scans are distributed in nature, where the attackers try to gain access from multiple machines at the same time. To keep secure an enterprise network, detection of such scanning is really important at early stage, i.e., before losing control of a machine. This is because cyber threats are becoming more sophisticated and more numerous, leading to more substantial damages to

systems within short periods of time [35, 100]. Two types of correlations are used in a coordinated scan attack, viz., *action correlation* and *task correlation* [21, 74]. How actions performed by one user affects another user is obtained during action correlation. For example, a particular action performed by one user may facilitate another user who performs the actual attack. In the other type of correlation, tasks divided among multiple users are discovered.

Network administrators or defenders are interested in detecting coordinated scan attacks for a system in an enterprise network due to the following reasons:

- To detect coordinated scan attacks just like the detection of other attacks
- To make it difficult for an attacker who may think it is easy and who wants to remain undetected
- To obviate the potential seriousness of the actual attacks

A *coordinated port scan* is a part of a coordinated attack. Here, tasks are distributed among multiple hosts for individual actions, which may be synchronized. A port scan is an information gathering method used by an opponent to gain information about responding computers and open ports on a target network host. An opponent initiates the exploration of multiple hosts to scan a portion of the target network, which they want to compromise after getting relevant information from the target host. Intrusion Detection Systems (IDSs) are normally configured to recognize and report single source port scan activity. So, they cannot usually detect multiple source scans that collaborate with several hosts during scanning.

A computer contains 65,536 standardly defined ports [99]. They can be classified into three large ranges: (a) well-known ports (0–1,023), (b) registered ports (1,024–49,151), and (c) dynamic and/or private ports (49,152–65,535). Normally, a port scan helps the attacker in finding those ports that are available to launch attacks, but it does not directly harm the system. Essentially, a port scan sends a packet with a message to the target host one at a time and listens for an answer. The response indicates whether the port is being used. This is a probe for weaknesses to launch future attacks. TCP and UDP ports are usually used for port scanning, but only TCP port scanning returns good feedback to the attacker because it is a connection-oriented protocol. UDP port scanning may not readily give relevant information to the attacker because it is a connectionless protocol. In addition, a UDP port may be easily blocked by network defenders or network administrators. The following are the various types of port scans [16] which are used to probe weaknesses from a networked host (shown in Fig. 4.5):

- *Stealth scan*: Auditing tools cannot detect this type of scanning because of complicated design architectures. Such a scan sends TCP packets to the destination host with stealth flags. Some of the flags are SYN, FIN, and NULL.
- *SOCKS port probe*: It allows sharing of Internet connections on multiple hosts. Attackers scan these ports because a large percentage of users misconfigure SOCKS ports, potentially permitting arbitrarily chosen sources and destinations to communicate. It also allows the attackers to access other Internet hosts while hiding their true locations.

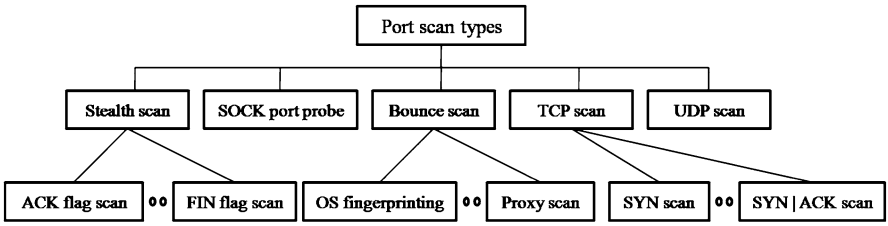


Fig. 4.5 Types of port scans

- *Bounce scan*: An FTP bounce scan attack takes advantage of a vulnerability of the FTP protocol itself. Email servers and HTTP proxies are the common applications that allow bounce scans.
- *TCP scan*: This type of scanning is used by a smart attacker because it never establishes a connection permanently. The attacker can launch an attack immediately if a remote port accepts the connection request. Normally, this type of connection request cannot be logged by a server's logging system due to its smart connection attempt. Some TCP scans are TCP Connect(), reverse identification, Internet protocol (IP) header dump scan, SYN, FIN, ACK, XMAS, NULL, and TCP fragment.
- *UDP scan*: A UDP scan attempts to discover open ports related to the UDP protocol. However, UDP is a connectionless protocol, and, thus, it is not often used by attackers since it can be easily blocked.

The list of *port scan* types discussed above along with firewall detection possibilities during the scanning process is given in Table 4.2. As seen from the table that most of scans are not detected at firewall level.

As discussed earlier, a *coordinated port scan* is composed of multiple scans from multiple sources where there is a single instigator behind the set of sources. The task of distributed information gathering is accomplished using either a many-to-one or a many-to-many model [52, 71]. The attacker uses multiple hosts to execute information-gathering techniques in two ways: rate limited and random or nonlinear. In a rate-limited information-gathering technique, the number of packets sent by a host to scan is limited [16, 47, 164]. This is based on the Berkeley Software Distribution (FreeBSD) implementation of UNIX where separate rate limits are maintained for open ports as well as closed ports. For example, TCP RST is rate limited. "ICMP port unreachable" is also rate limited. On the other hand, a random or nonlinear gathering technique refers to randomization of the destination IP-port pairs among the sources, as well as randomization of the time delay for each probe packet. A coordinated attack has a more generic form of a distributed scan [141]. It is defined as multistep exploitation using parallel sessions with the objective of obscuring the unified nature of the attack, allowing the attackers to proceed more quickly. Bhuyan et al. [18] introduce a coordinated scan detection technique known as AOCD(adaptive outlier-based coordinated scan detection). A framework of AOCD is given in Fig. 4.6.

Table 4.2 Port scan types and their firewall level detection possibilities

Port scanning technique	Protocol	TCP flag	Target reply (open port)	Target reply (closed port)	Firewall level detection possibility
TCP *Connect()*	TCP	SYN	ACK	RST	Yes
Reverse ident	TCP	No	No	No	No
SYN scan	TCP	SYN	ACK	RST	Yes
IP header dump scan	TCP	No	No	No	No
SYN|ACK scan	TCP	SYN|ACK	RST	RST	Yes
FIN scan	TCP	FIN	No	RST	No
ACK scan	TCP	ACK	No	RST	No
NULL scan	TCP	No	No	RST	No
XMAS scan	TCP	All flags	No	RST	No
TCP fragment	TCP	No	No	No	No
UDP scan	UDP	No	No	Port unreachable	No
FTP bounce scan	FTP	Arbitrary flag set	No	No	No
Ping scan	ICMP	No	Echo reply	No	Yes
List scan	TCP	No	No	No	No
Protocol scan	IP	No	–	–	No
TCP window scan	TCP	ACK	RST	RST	No

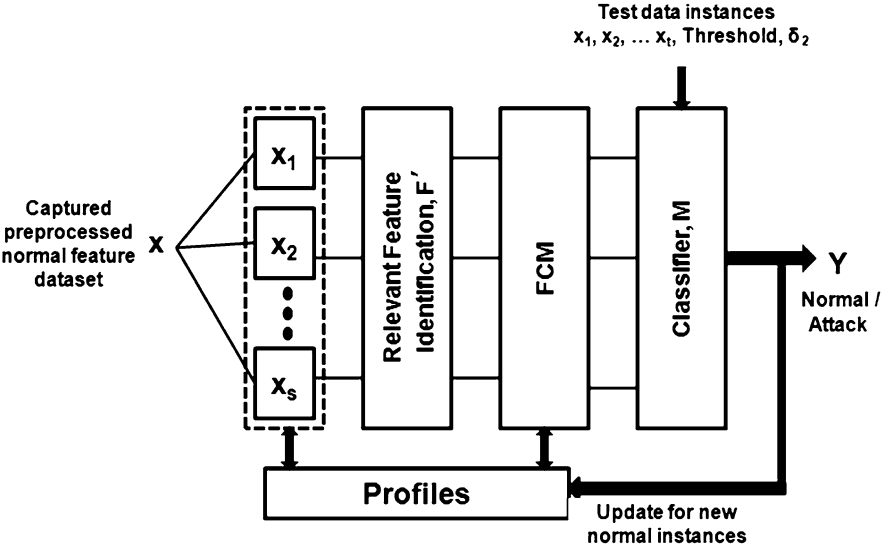


Fig. 4.6 A framework for AOCD: FCM is the fuzzy C-means clustering algorithm for sample clustering and F' is the PCA-based feature selection technique for each sample as well as testing instances

AOCD aims to detect anomalous patterns, i.e., coordinated port scans using an adaptive outlier-based approach with reference to profiles. Initially, they select random samples, $x_1, x_2, \cdots x_s$ using a linear congruential generator from the dataset x for training purpose. It is a maximum length pseudo random sequence generator [132] and can be defined as $x_n = (ax_{n-1} + b)\ mod\ m$, where x_n is the n^{th} number of the sequence, x_{n-1} is the previous number of the sequence. a, b, and m are secrets; a is the multiplier, b is the increment, and m is the period length when m is prime, the maximum period length is $(m-1)$.

They cluster each sample into k classes using the Fuzzy C-means [12] clustering technique. They obtain the following clusters from all samples: $C_{11}, C_{12}, C_{13}, \cdots C_{1k}$, $C_{21}, C_{22}, C_{23}, \cdots C_{2k}, \cdots\ C_{s1}, C_{s2}, C_{s3}, \cdots C_{sk}$. The method compares a range-based profile for each cluster and matches each profile with others to remove redundancy. These profiles are used as references during score computation. Finally, the method computes score for each candidate object and reports as normal or outlier (i.e., attack) w.r.t. a threshold, δ_2. This technique demonstrates that AOCD can detect coordinated scan attacks effectively at early stage.

Abbes et al. [1] introduce an approach that uses decision trees with protocol analysis for effective intrusion detection. They construct an adaptive decision tree for each application layer protocol. Detection of anomalies classifies data records into two classes: benign and anomalous or malignant. The anomalies include a large variety of types such as DoS, scans, and botnets. Thus, multi-class classifiers are a natural choice, but like any classifier they require expensive hand-labeled datasets and are also not able to identify unknown attacks. Wagner et al. [152] use *one-class classifiers* that can detect new anomalies, i.e., data points that do not belong to the learned class. In particular, they use a one-class SVM classifier proposed by Schölkopf et al. [129]. In such a classifier, the training data is presumed to belong to only one class, and the learning goal during training is to determine a function which is positive when applied to points on the circumscribed boundary around the training points and negative outside. This is also called *semi-supervised classification*. Such an SVM classifier can be used to identify outliers and anomalies. The authors develop a special kernel function that projects data points to a higher dimension before classification. Their kernel function takes into consideration properties of NetFlow data and enables determination of similarity between two windows of IP flow records. They obtain 92% accuracy on average for all attacks classes.

Leckie and Kotagiri [84] present an algorithm based on a probabilistic model. For each IP address in the monitored network, the algorithm generates a probability $P(d|s)$ that represents how likely it is that a source will contact that particular destination IP, where d is the destination IP and s is the source, based on how commonly that destination IP is contacted by other sources, $P(d)$. Similarly, it also computes a probability for each port that represents how likely a source will contact a particular destination port, $P(p|s)$ where p is the destination port. A limitation of this approach is that $P(d)$ is based on the prior distribution of sources that have accessed that IP address. This implies that if the probabilities for this approach are generated based on a sample of network data and if the monitored network is scanned, the resulting distributions may include scans as well as normal traffic.

Another limitation of this approach is that it assumes that an attacker accesses the destinations at random; this may not be always true. Kim et al. [79] aim to detect network port scans using anomaly detection. First, the method performs statistical tests to analyze traffic rates. Then, it makes use of two dynamic *chi-square* tests to detect anomalous packets. It models network traffic as a marked point process and introduces a general port scan model. The authors present simulation results to detect *ten* malicious vertical scans with true positive rate greater than 90% and false-positive rate smaller than 15% for both the static and dynamic tests using the port scan model and statistical tests.

Gates et al. [59] analyze Cisco NetFlow data for port scan attacks. The method extracts the events (bursts of network activities surrounded by quiescent periods) for each source, and the flows in each event are then sorted according to destination IP and destination port. It attempts to calculate six characteristics for each event based on statistical analysis of port scans. It estimates a probability using logistic regression with these six characteristics as input variables to predict whether the events contain a scan or not. The main drawback of the method is that it is non-real time. Udhayan et al. [150] report a heuristic approach for detecting port scan attacks. One possible solution to curb a zombie army or a malicious botnet attack is by detecting and blocking or dropping reconnaissance scans, i.e., port scans. They derive a set of heuristics to detect these scans, some quite crafty. The hierarchies are written into the firewall and are triggered immediately after a port scan is detected, to drop packets with the IP address of the source of port scan for a predetermined period. This detection approach is more user friendly than other approaches like SNORT [125].

Gyorgy et al. [65] propose a model known as off-the-shelf classifier. Initially, it transforms network trace data into a feature dataset with label information. Then, it selects Ripper, a fast rule-based classifier, which is capable of learning rules from multimodal datasets, with results that are easy to interpret. The authors successfully demonstrate that data mining models can encapsulate expert knowledge to create an adaptive algorithm that can substantially outperform the state of the art for heuristic-based scan detection in both precision and recall. This technique is also capable of detecting the scanners at an early stage. Treurniet [148] introduces a new scan detection technique that improves the understanding of Internet traffic. The author creates a session model using the behavior of packet level data between host pairs identified and activities between them. In a dataset collected over 24 h, 78% of the instances were identified as reconnaissance activities, out of which 80% were slower scans.

Heberlein et al. [67] present a system known as Network Security Monitor(NSM), which pioneered the implementation of threshold-based scan detection [74]. This tool has three parts: data capture, data analysis, and support. The data analysis is the core part of the NSM. It collects data in different forms such as statistical, session, full content, and alert. Statistical data represent the aggregation of network traffic, protocol breakdowns, and distribution. Session data represent the connection pairs and conversation between two hosts. Full content data represent the log of every single bit of network traffic. Alert data represent the data collected

by an IDS. It recognizes a source as anomalous and potentially malicious if it is found to contact more than 15 other IP addresses during an unspecified period of time. It also identifies a source as anomalous if it tries to contact an IP address that does not contain a responding computer on the monitored network. With this last heuristic, it assumes that an external source would contact an internal IP address only for a reason backed by knowledge of the existence of a service at an internal IP address such as an FTP server or a mail server. NSM is neither a security event management system nor an intrusion prevention system. Roesch [125] presents a signature-based intrusion detection system known as SNORT. It uses a preprocessor that extracts port scans, based on either invalid flag combination (e.g., NULL scans, Xmas scans, and SYN-FIN scans) or on exceeding a threshold. SNORT uses a preprocessor, called *portscan* that watches connections to determine whether a scan is occurring. By default, SNORT is configured to generate an alarm only if it has detected SYN packets sent to at least five different IP addresses within 60 s or 20 different ports within 60 s, although this can be adjusted manually. By having such a high threshold, the number of false positives is reduced. However, a careful scan at a rate lower than the threshold can easily go undetected.

Paxson [114] introduces a detection system known as Bro that also attempts to detect scans based on a thresholding approach. Network scans are detected when a single source contacts multiple destinations (>some threshold). It also detects vertical scans when a single source contacts too many different ports. It assumes that the external site has initiated the conversation in both cases. However, a major limitation of this method is the increased number of false positives. Bro uses payload as well as packet level information. Jung et al. [74] describe an approach called Threshold Random Walk (TRW) based on sequential hypothesis testing. It detects port scans using an Oracle database that contains the assigned IP addresses and ports inside a network after performing an analysis of return traffic. When a connection request is received, the source IP is entered into a list, along with each destination to which this source has attempted a connection. If the current connection is to a destination which is already in the list, the connection is ignored. If it is to a new destination, it is added to the list, and a measure that determines whether the connection is scanning or not is computed and updated based on the status of the connection. The entire source is flagged as either scanning or not scanning depending on whether the measure has exceeded the maximum threshold or has dropped below the minimum threshold, respectively. It has been observed that benign activity rarely results in connections to hosts or services that are not available, whereas scanning activity often makes such connections, with the probability of connecting to a legitimate service dependent on the density of the target network.

Romig [127] develops a flow analysis tool called *flow-dscan*. This tool examines flows for floods and port scans. Floods are identified by an excessive number of packets per flow. Port scans are identified by a source IP address contacting more than a certain threshold number of destination IP addresses or destination ports (only ports less than 1,024 are examined) on a single IP address. To minimize the false alarm rate, this approach makes use of a suppress list consisting of IP addresses.

Zhang and Fang [167] propose a new port scan detection approach known as time-based flow size distribution sequential hypothesis testing (TFDS) for high-speed transit networks where only unidirectional flow information is available. TFDS uses the main ideas of sequential hypothesis testing to detect scanners that exhibit abnormal access patterns in terms of flow size distribution (FSD) entropy. This work makes a comparison with the state-of-the-art backbone port scan detection method TAPS [140] in terms of efficiency and effectiveness using real backbone packet trace and finds that TFDS performs much better than TAPS.

Gadge and Patil [56] propose a method to identify possible port scans and try to gather additional information about the scanner or attacker, such as probable location and operating system. The scan detection system collects all the information and stores it to generate reports in terms of bar graphs. Analysis of the stored data can be performed in terms of time and day, by which type of scan was performed, from which IP the scan was performed, different ports, etc. Based on the analysis of the various parameters used, it can recognize and report the type of attack or scan performed during a time window. This method can detect scans coming from most common scanners such as Angry IP, nmap, and MegaPing.

Classification-based anomaly detection techniques usually give better results than unsupervised methods (e.g., clustering based) because of the use of labeled training examples. In traditional classification, new information can be incorporated by retraining with the entire dataset. However, this is time-consuming. Incremental classification algorithms [142] make such training more efficiently. Although classification-based methods are popular, they cannot detect or predict an unknown attack or event until relevant training information is fed for retraining. For a comparison of several classification-based network anomaly detection techniques, see Table 4.3.

Several authors have used a combination of classifiers and clustering for network intrusion detection leveraging the advantages of the two techniques. For example, Gaddam et al. [55] present a method to detect anomalous activities based on a combined approach that uses the k-means clustering algorithm and the ID3 algorithm for decision tree learning [123]. In addition to descriptive features, each data instance includes a label saying whether the instance is normal or anomalous. The first stage of the algorithm partitions the training data into k clusters using Euclidean distance similarity. Obviously, the clustering algorithm does not consider the labels on instances. The second stage of the algorithm builds a decision tree on the instances in a cluster. It does so for each cluster so that k separate decision trees are built. The purpose of building decision trees is to overcome two problems that k-means faces: a) *forced assignment*, if the value of k is lower than the number of natural groups, dissimilar instances are forced into the same cluster, and b) *class dominance*, which arises when a cluster contains a large number of instances from one class and fewer numbers of instances from other classes. The hypothesis is that a decision tree trained on each cluster learns the subgroupings (if any) present within each cluster by partitioning the instances over the feature space. To obtain a final decision on classification of a test instance, the decisions of the k-means and ID3 algorithms are combined using two rules: (a) the nearest neighbor rule

Table 4.3 Comparison of classification-based network anomaly detection techniques

Author (s)	Year of publication	No. of parameters	w	x	y	Data types	Dataset used	z	Detection method
Tong et al. [146]	2005	4	O	N	P	Numeric	DARPA99, TCPSTAT	C_1	KPCC model
Gaddam et al. [55]	2007	3	C	N	P	Numeric	NAD, DED, MSD	C_1	k-means+ID3
Khan et al. [76]	2007	3	C	N	P	Numeric	DARPA98	C_1	DGSOT+SVM
Das et al. [37]	2008	3	O	N	P	Categorical	KDDcup99	C_1	APD algorithm
Lu and Tong [92]	2009	2	O	N	P	Numeric	DARPA99	C_1	CUSUM-EM
Qadeer et al. [122]	2010	–	C	R	P	–	Real time	C_2	Packet analysis tool
Wagner et al.[152]	2011	2	C	R	F	Numeric	Flow Traces	C_2	Kernel OCSVM
Muda et al. [102]	2011	2	O	N	O	Numeric	KDDcup99	C_1	KMNB algorithm
Kang et al. [75]	2012	2	O	N	P	Numeric	DARPA98	C_1	Differentiated SVDD

w indicates centralized (C) or distributed (D) or others (O), *x* the nature of detection as real time (R) or non-real time (N), *y* characterizes packet-based (P) or flow-based (F) or hybrid (H) or others (O), *z* represents the list of attacks handled: C_1 all attacks, C_2 denial of service, C_3 probe, C_4 user to root, and C_5 remote to local

and (b) the nearest consensus rule. The authors claim that the detection accuracy of the *k*-means+ID3 method is very high with an extremely low false-positive rate on network anomaly data.

Support vector machines (SVMs) are very successful maximum margin linear classifiers [160]. However, SVMs take a long time for training when the dataset is very large. Khan et al. [76] reduce the training time for SVMs when classifying large intrusion datasets by using a hierarchical clustering method called dynamically growing self-organizing tree (DGSOT) intertwined with the SVMs. DGSOT, which is based on artificial neural networks, is used to find the boundary points between two classes. The boundary points are the most qualified points to train SVMs. An SVM computes the maximal margins separating the two classes of data points. Only points closest to the margins, called support vectors, affect the computation of these margins. Other points can be discarded without affecting the final results. Khan et al. approximate the support vectors by using DGSOT. They use clustering in parallel with the training of SVMs, without waiting till the end of the building of the tree to start training the SVM. The authors find that their approach significantly improves training time for the SVMs without sacrificing generalization accuracy, in the context of network anomaly detection.

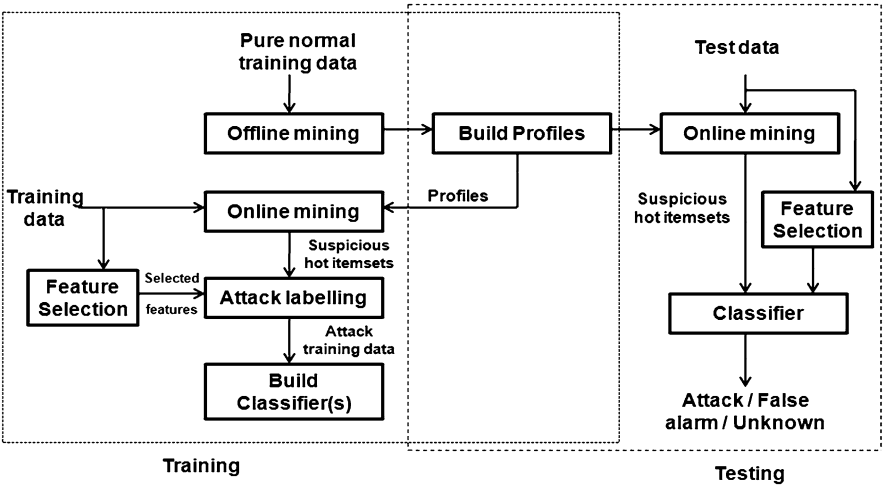


Fig. 4.7 Architecture of ADAM system

4.4.2 Classification-Based Systems

An example of a classification-based IDS is automated data analysis and mining (ADAM) [36] that provides a testbed for detecting anomalous instances. An architecture diagram of ADAM is shown in Fig. 4.7. ADAM exploits a combination of classification techniques and association rule mining to discover attacks in a tcpdump audit trail. First, ADAM builds a repository of "normal" frequent itemsets from attack-free periods. Second, ADAM runs a sliding window-based online algorithm that finds frequent itemsets in the connections and compares them with those stored in the normal itemset repository, discarding those that are deemed normal. ADAM uses a classifier which has been trained to classify suspicious connections as either a known type of attack or an unknown type or a false alarm.

Dependable Network Intrusion Detection System (DNIDS) [82] is a classification-based IDS. This IDS uses the combined strangeness and isolation measure of the *k*-nearest neighbor (CSI-KNN) algorithm developed as a part of the system. DNIDS can effectively detect network intrusion while providing continued service under attack. The intrusion detection algorithm analyzes characteristics of network data by employing two measures: strangeness and isolation. These measures are used by a correlation unit to raise intrusion alert along with the confidence information. For faster information, DNIDS exploits multiple CSF-KNN classifiers in parallel. It also includes an intrusion tolerance mechanism to monitor the hosts and the classifiers running on them, so that failure of any component can be handled carefully. Sensors capture network packets from a network segment and transform them into connection-based vectors. The detector is a collection of CSI-KNN classifiers that analyze the vectors supplied by the sensors. The manager,

alert agents, and maintenance agents are designed for intrusion tolerance and are installed on a secure administrative server called Station. The manager executes the tasks of generating mobile agents and dispatching them for task execution.

Some advantages of classification-based anomaly detection techniques are the following:

- These techniques are usually trained on the actual traffic data. Thus, the performance is usually good.
- Efficiency of training a classification algorithm depends on the algorithm itself. Algorithms like decision trees and random forests are quite efficient.
- They have a high detection rate for known attacks subject to appropriate threshold setting.

Though such techniques are popular, they have the disadvantages including the following:

- The techniques cannot detect or predict unknown attack or event until relevant training information is fed.
- All anomaly classes need to be known a priori. This may be difficult in practice.

4.5 Clustering and Outlier-Based Techniques and Systems

Clustering is the task of assigning a set of objects into groups called *clusters* so that the objects in the same cluster are more similar in some sense to each other than to those in other clusters. Clustering is used in exploratory data mining. For example, if we have a set of unlabeled objects in two dimensions, we may be able to cluster them into *five* clusters by drawing circles or ellipses around them, as in Fig. 4.8a. Outliers are those points in a dataset that are highly unlikely to occur, given a model of the data, as in Fig. 4.8b. Examples of outliers in a simple dataset are seen in [15]. Clustering and outlier finding are examples of unsupervised machine learning.

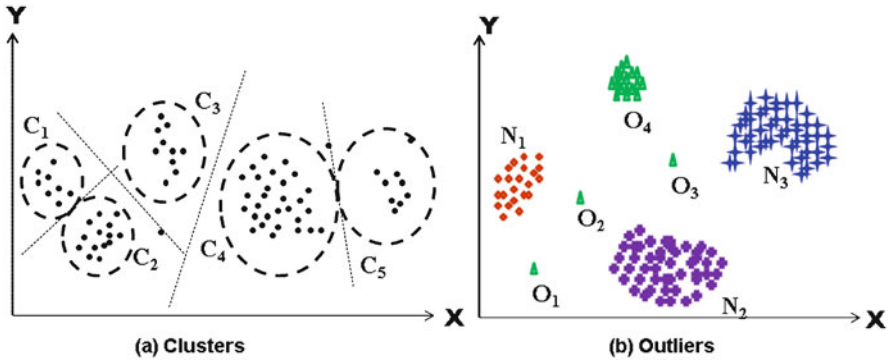


Fig. 4.8 Clustering and outliers in 2-D, where C_is are clusters in (**a**) and O_is are outliers in (**b**)

Clustering can be performed in network anomaly detection in an offline environment. Such an approach adds additional depth to the administrators' defenses and allows them to more accurately determine threats against their network through the use of multiple methods on data from multiple sources. Hence, the extensive amount of activities that may be needed to detect intrusion near real time in an online NIDS may be obviated, achieving efficiency [85].

4.5.1 Clustering-Based Techniques

Clustering techniques are frequently used in anomaly detection. These include single-link clustering algorithms, k-means (squared error clustering), and hierarchical clustering algorithms to mention a few [26, 87, 104, 125, 163, 168].

Sequeira and Zaki [135] present an anomaly-based intrusion detection system known as ADMIT that detects intruders by creating user profiles. It keeps track of the sequence of commands a user uses as he/she uses a computer. A user profile is represented by clustering the sequences of user commands. The data collection and processing are thus host-based. The system clusters a user's command sequence using LCS (the longest common subsequence) as the similarity metric. It uses a dynamic clustering algorithm that creates an initial set of clusters and then refines them by splitting and merging as necessary. When a new user types a sequence of commands, it compares the sequence to profiles of users it already has. If it is a long sequence, it is broken up to a number of smaller sequences. A sequence that is not similar to a normal user's profile is considered anomalous. One anomalous sequence is tolerated as noise, but a sequence of anomalous sequences typed by one single user causes the user to be marked as masquerader or concept drift. The system can also use incremental clustering to detect masqueraders.

Zhang et al. [168] report a distributed intrusion detection algorithm that clusters the data twice. The first clustering chooses candidate anomalies at agent IDSs, which are placed in a distributed manner in a network, and a second clustering computation attempts to identify true attacks at the central IDS. The first clustering algorithm is essentially the same as the one proposed by [50]. At each agent IDS, small clusters are assumed to contain anomalies, and all small clusters are merged to form a single candidate cluster containing all anomalies. The candidate anomalies from various agent IDSs are sent to the central IDS, which clusters again using a simple single-link hierarchical clustering algorithm. It chooses the smallest k clusters as containing true anomalies. They obtain 90% attack detection rate on test intrusion data.

Bhuyan et al. [17] present an unsupervised network anomaly detection method for large intrusion datasets. It exploits tree-based subspace clustering and an ensemble-based cluster labeling technique to achieve better detection rate over real-life network traffic data for the detection of known as well as unknown attacks. It can detect network anomalies without relying on existing signatures, training, or labeled data. The proposed approach runs in two consecutive phases for analyzing network

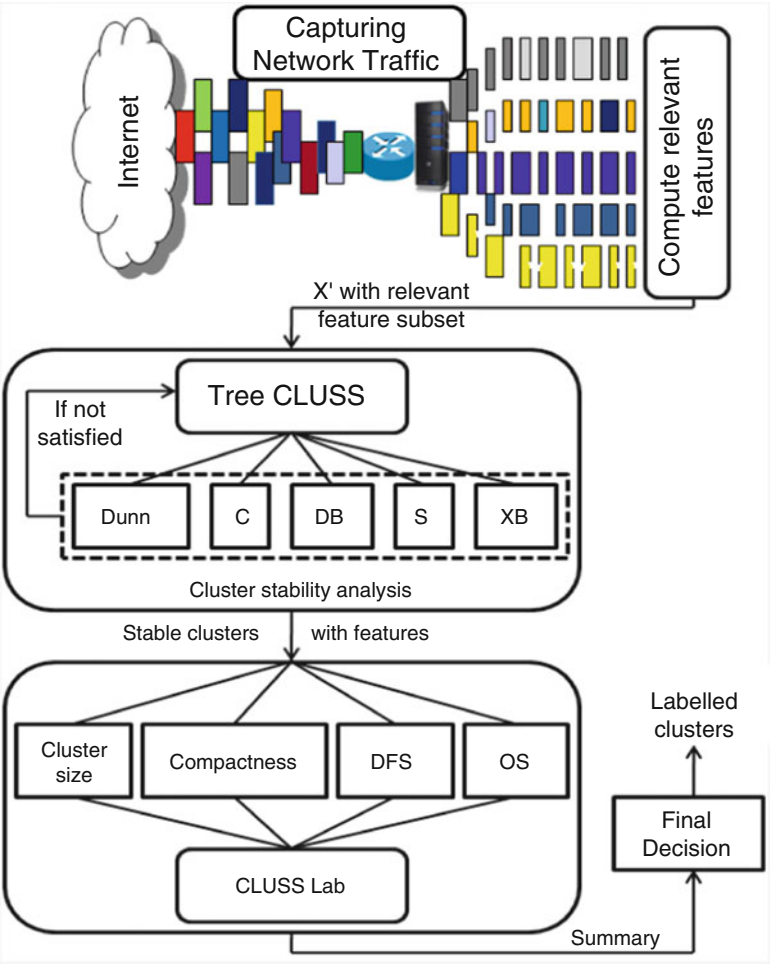


Fig. 4.9 High-level description of the unsupervised network anomaly detection method

traffic in contiguous time slots of fixed length. Figure 4.9 provides a conceptual framework of the proposed unsupervised network anomaly detection method.

In the *first* phase, a tree-based subspace clustering technique (TreeCLUSS) generates clusters in high-dimensional large datasets. It is well known that network intrusion datasets are high dimensional and large. The authors apply this technique over a subset of features. TreeCLUSS uses the MMIFS technique [5] for finding a highly relevant feature set. It uses a subset of features during cluster formation without using any class labels. They analyze the stability of the cluster results obtained. Cluster stability analysis for real-life data is not a trivial task. It is performed using an ensemble of several index measures, viz., Dunn index [45], C-index (C) [69], Davies-Bouldin index (DB) [38], Silhouette index (S) [128], and

Xie-Beni index (XB) [156]. They choose a stable set of clusters when a certain number of clusters produce better result after multiple execution of this module. In the *second* phase, they apply a cluster labeling technique (CLUSSLab) to label the stable clusters using a multi-objective approach. CLUSSLab takes into account the following features: cluster size, compactness obtained from the ensemble of five index measures, a dominating feature subset (DFS) obtained for each cluster based on an unsupervised feature clustering technique, and an outlier score (OS) obtained based on the RODD technique [15]. Finally, they label each cluster as normal or anomalous based on the described measures. They obtain 98% detection rate on average in detecting network anomalies.

Some advantages of using clustering are given below:

- For a partitioning approach, if k can be provided accurately, then the task is easy.
- Incremental clustering (in supervised mode) techniques are effective for fast response generation.
- In a large datasets, grouping first a number of clusters before detecting network anomalies is useful, because it reduces the computational complexity during intrusion detection.
- It provides stable performance in comparison to classifiers or statistical methods.

Drawbacks of clustering-based techniques include the following:

- Most techniques are able to handle continuous attributes only.
- In clustering-based intrusion detection techniques, an assumption is that the larger clusters are normal and smaller clusters are attack or intrusion [119]. Without this assumption, it is difficult to evaluate the technique.
- The use of an inappropriate proximity measure affects the detection rate negatively.
- Dynamic updation of profiles is time-consuming.

4.5.2 Outlier-Based Techniques

Several outlier-based network anomaly identification techniques are available in [63]. When we use outlier-based algorithms, the assumption is that anomalies are uncommon events in a network. Intrusion datasets usually contain mixed, numeric, and categorial attributes. Many early outlier detection algorithms worked with continuous attributes only. They ignored categorial attributes or modeled them in manners that caused considerable loss of information.

To overcome this problem, Otey et al. [108] develop a distance measure for data containing a mix of categorical and continuous attributes and use it for outlier-based anomaly detection. They define an anomaly score which can be used to identify outliers in mixed attribute space by considering dependencies among attributes of different types. Their anomaly score function is based on a global model of the data that can be easily constructed by combining local models built independently at

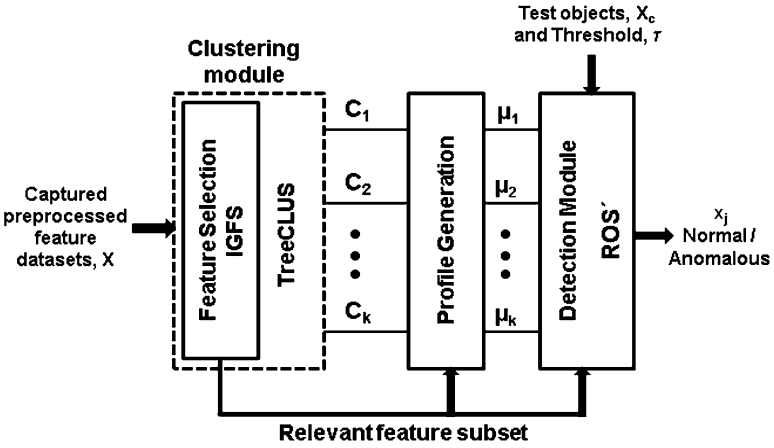


Fig. 4.10 A framework for outlier-based network anomaly detection. The framework takes feature dataset $\mathbb{X}$ as input. It initially selects a relevant and optimal feature subset using the IGFS algorithm. TreeCLUS uses this feature subset to form clusters, $C_1, C_2, \cdots, C_k$. Once cluster formation is over, it generates mean-based profiles, $\mu_1, \mu_2, \cdots, \mu_k$ for each cluster. Finally, ROS′ computes the outlier score for each test object, x_c, and reports as normal or anomalous based on a user-defined threshold, τ

each node. They develop an efficient one-pass approximation algorithm for anomaly detection that works efficiently in distributed detection environments with very little loss of detection accuracy. Each node computes its own outliers, and the internode communication needed to compute global outliers is not significant. In addition, the authors show that their approach works well in dynamic network traffic situations where data, in addition to being streaming, also changes in nature as time progresses leading to concept drift. Bhuyan et al. [13] introduce a multi-step outlier-based technique to detect network-wide traffic anomalies. It works by identifying reference points and by ranking based on outlier scores of candidate objects. The technique has four parts including feature selection, clustering, profile generation, and outlier detection. Figure 4.10 shows the framework for outlier-based technique to network anomaly detection.

The authors consider seven different classes of outliers the methods can detect as shown in Fig. 4.11. These seven classes were introduced in [14] by us, and they are defined formally as follows:

Definition 4.1 *Outlier Score*—An outlier score ROS′ with respect to a reference point is defined as a summarized value that combines distance and maximum frequency of class occurrence with respect to k'-nearest neighbors of each candidate data object (see formula in Eq. (4.1)).

Definition 4.2 *Distinct Inlierness*—An object O_i is defined as a distinct inlier if it conforms to normal objects, i.e., $\text{ROS}'(O_i, C_i) \ll \tau$ for all i.

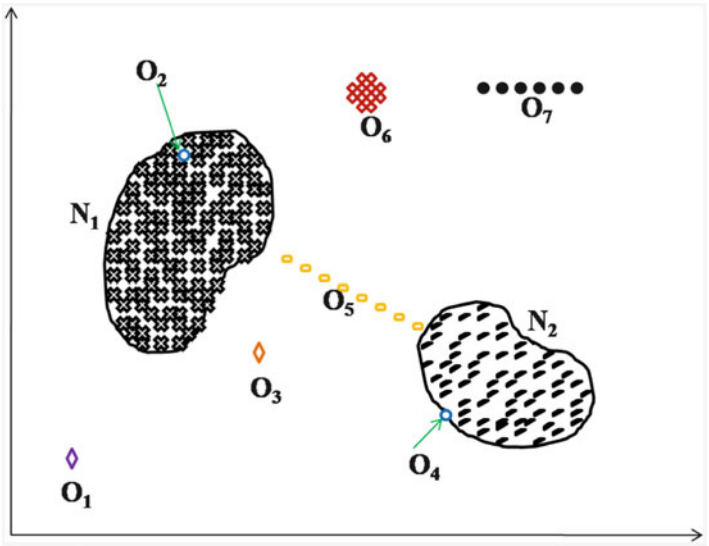


Fig. 4.11 Illustration of seven different cases: N_1 and N_2 are two normal clusters, O_1 is the distinct outlier, O_2, the distinct inlier, O_3, the equidistance outlier, O_4, the border inlier, O_5, a chain of outliers, O_6 is another set of outlier objects with higher compactness among the objects, and O_7 is an outlier case of "*stay together*"

Definition 4.3 *Border Inlierness*—An object O_i is defined as a border object in a class C_i, if $\text{ROS}'(O_i, C_i) < \tau$.

Definition 4.4 *Outlier*—An object O_i is defined as an outlier w.r.t. any normal class C_i and the corresponding profile R_i iff (i) $\text{ROS}'(O_i, C_i) \geq \tau$, and (ii) $\text{dist}(O_i, R_i) > \alpha'$, where α' is a proximity-based threshold and *dist* is proximity measure.

Definition 4.5 *Distinct Outlierness*—An object O_i is defined as a distinct outlier iff it deviates exceptionally from the normal objects, i.e., from the generic class C_i. In other words, $\text{ROS}'(O_i, C_i) \gg \tau$ for all i.

Definition 4.6 *Equidistant Outlierness*—An object O_i is defined as equidistant outlier from classes C_i and C_j, if $\text{dist}(O_i, C_i) = \text{dist}(O_i, C_j)$ but $\text{ROS}'(O_i, C_i) > \tau$.

Definition 4.7 *Chain Outlierness*—A set of objects, $O_i, O_{i+1}, O_{i+2} \cdots$ is defined as a chain of outliers if $\text{ROS}'(O_{i+l}, C_i) \geq \tau$, where $l = 0, 1, 2, \cdots, z$.

Let S_i be the number of classes to which each of k'-nearest neighbor data objects belongs. k' plays an important role in score computation. Let x_c be a data object in $\mathbb{X}_c$ and $dist(x_c, R_i)$ be the distance between the data object x_c and the reference points R_i, where $c = 1, 2, 3, \cdots n$, dist is a proximity measure used, and $\mathbb{X}_c$ represents the whole candidate dataset. The technique works well with any commonly used proximity measure. The outlier score for a data object x_c is given in Eq. (4.1).

$$ROS'(x_c) = \frac{\max_{1 \le i \le k'} S_i}{k'} \times \left(\frac{\left(1 - \min_{1 \le i \le k'} \text{dist}(x_c, R_i)\right) \times \left(\sum_{i=1}^{k'} \min_{1 \le i \le k'} \text{dist}(x_c, R_i)\right)}{\sum_{i=1}^{k'} \max_{1 \le i \le k'} \text{dist}(x_c, R_i)} \right) \quad (4.1)$$

Here, $\frac{\max_{1 \le i \le k'} S_i}{k'}$ is the maximum probability that a data object belongs to a particular class. The remaining part is the summarized value of the distance measure with k'-nearest neighbors over the relevant feature subset. The candidate data objects are ranked based on outlier score values. Objects with scores higher than a user-defined threshold τ are considered anomalous or outliers.

Some of the advantages of outlier-based anomaly detection are the following:

- It is easy to detect outliers when the datasets are smaller in size.
- Bursty and isolated attacks can be identified efficiently using this method.

Drawbacks of outlier-based anomaly detection include the following:

- Most techniques use both clustering and outlier detection. In such cases the complexity may be high in comparison to other techniques.
- The techniques are highly parameter dependent.

A comparison of a few clustering and outlier-based network anomaly detection techniques is given in Table 4.4.

4.5.3 Clustering and Outlier-Based Systems

Worms are often intelligent enough to hide their activities and evade detection by IDSs. Zhuang et al. [171] propose a method called PAIDS (proximity-assisted IDS) to identify the new worms as they begin to spread. PAIDS works differently from other IDSs and has been designed to work collaboratively with existing IDSs such as anomaly-based IDSs for enhanced performance. The goal of the designers of PAIDS is to identify new and intelligent fast-propagating worms and thwarting their spread, particularly as the worm is just beginning to spread. Neither signature-based nor anomaly-based techniques can achieve such capabilities. Zhuang et al.'s approach is based mainly on the observation that during the starting phase of a new worm, the infected hosts are clustered in terms of geography, IP address, and maybe, even DNSes used.

MINDS (Minnesota Intrusion Detection System) [48] is a data mining-based system for detecting network intrusions. The architecture of MINDS is given in Fig. 4.12. It accepts NetFlow data collected through flow tools as input. Flow tools only capture packet header information and build one way sessions of flows. The analyst uses MINDS to analyze these data files in batch mode. The reason for running the system in batch mode is not due to the time it takes to analyze these files but because it is convenient for the analyst to do so. Before data is fed into the anomaly detection module, a data filtering step is executed to remove network traffic in which the analyst is not interested.

Table 4.4 Comparison of clustering and outlier-based network anomaly detection techniques

Author (s)	Year of publication	No. of parameters	w	x	y	Data types	Dataset used	z	Detection method
Sequeira and Zaki [135]	2002	4	C	R	P	Numeric, Categorical	Real life	Synthetic intrusions	ADMIT
Zhang et al. [168]	2005	2	D	N	P	Numeric	KDDcup99	C_1	Cluster-based DIDS
Leung and Leckie [87]	2005	3	C	N	P	Numeric	KDDcup99	C_1	fpMAFIA algorithm
Otey et al. [108]	2006	5	C	N	P	Mixed	KDDcup99	C_1	FDOD algorithm
Jiang et al. [73]	2006	3	C	N	P	Mixed	KDDcup99	C_1	CBUID algorithm
Chen and Chen [32]	2008	–	O	N	–	–	–	C_3	AAWP model
Zhang et al. [163]	2009	2	O	N	P	Mixed	KDDcup99	C_1	KD algorithm
Zhuang et al. [171]	2010	2	R	C	P	–	Real time	C_6	PAIDS model
Bhuyan et al. [16]	2011	2	N	C	P,F	Numeric	KDDcup99	C_1	NADO algorithm
Casas et al. [26]	2012	2	N	C	F	Numeric	KDDcup99, Real time	C_1	UNIDS method

w-indicates centralized (C) or distributed (D) or others (O), *x*-the nature of detection as real time (R) or non-real time (N), *y*-characterizes packet-based (P) or flow-based (F) or hybrid (H) or others (O), *z*-represents the list of attacks handled: C_1 all attacks, C_2 denial of service, C_3 probe, C_4 user to root, C_5 remote to local, and C_6 worms

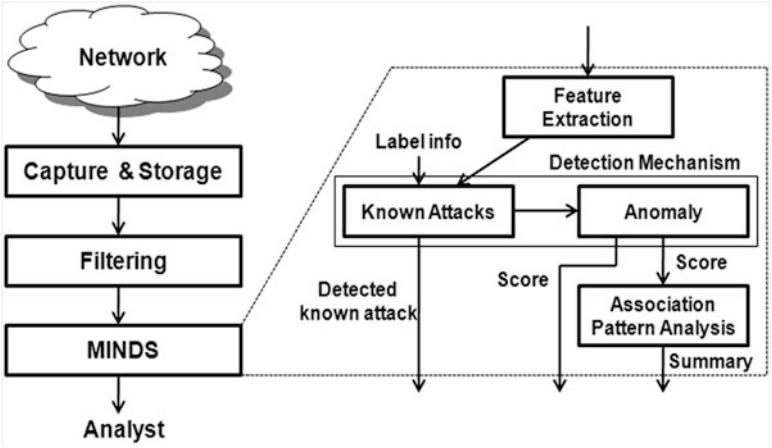


Fig. 4.12 Architecture of MINDS system

The first step of MINDS is to extract important features that are used. Then, it summarizes the features based on time windows. After the feature construction step, the known attack detection module is used to detect network connections that correspond to attacks for which signatures are available and to remove them from further analysis. Next, an outlier technique is activated to assign an anomaly score to each network connection. A human analyst then looks at only the most anomalous connections to determine if they are actual attacks or represent other interesting behavior. The association pattern analysis module of this system is dedicated to summarize the network connections as per the assigned anomaly rank. The analyst provides feedback after analyzing the summaries created and decides whether these summaries are helpful in creating new rules that may be used in known attack detection.

4.6 Soft Computing Techniques and Systems

Soft computing techniques are suitable for network anomaly detection because often one cannot find exact solutions. Soft computing is usually thought of as encompassing methods such as genetic algorithms (GA), artificial neural networks (ANN), fuzzy sets, rough sets, ant colony algorithms, and artificial immune systems. We describe several soft computing techniques and systems for network anomaly detection below.

4.6.1 GA-Based Techniques

Genetic algorithms (GA) are population-based adaptive heuristic search techniques based on evolutionary ideas. The approach begins with conversion of a problem into a framework that uses a chromosome-like data structure. Balajinath and Raghavan [11] present a genetic intrusion detector (GBID) based on learning of individual user behavior. User behavior is described as 3-tuple <matching index, entropy index, newness index> and is learnt using a genetic algorithm. This behavior profile is used to detect intrusion based on past behavior.

Khan [77] uses genetic algorithms to develop rules for network intrusion detection. A chromosome in an individual contains genes corresponding to attributes such as the service, flags, logged in or not, and super-user attempts. Khan concludes that attacks that are common can be detected more accurately compared to uncommon attributes.

4.6.2 ANN-Based Techniques

Artificial neural networks (ANN) are motivated by the recognition that the human brain computes in an entirely different way from the conventional digital computer [66]. The brain organizes its constituents, known as neurons, so as to perform certain computations (e.g., pattern recognition, perception, and motor control) many times faster than the fastest digital computer. To achieve good performance, real neural networks employ massive interconnections of neurons. Neural networks acquire knowledge of the environment through a process of learning, which systematically changes the interconnection strengths or synaptic weights of the network to attain a desired design objective.

Cannady's approach [24] autonomously learns new attacks rapidly using modified reinforcement learning. His approach uses feedback for signature update when a new attack is encountered and achieves satisfactory results. An improved approach to detect network anomalies using a hierarchy of neural networks is introduced in [86]. The neural networks are trained using data that spans the entire normal space and are able to recognize unknown attacks effectively.

Liu et al. [89] report a real-time solution to detect known and new attacks in network traffic using unsupervised neural nets. It uses a hierarchical intrusion detection model using principal components analysis (PCA) neural networks to overcome the shortcomings of single-level structures. Sun et al. [144] present a wavelet neural network (WNN)-based intrusion detection method. It reduces the number of the basic wavelet functions by analyzing the sparseness property of sample data to optimize the wavelet network to a large extent. The learning algorithm trains the network using gradient descent. Yong and Feng [159] use recurrent multilayered perceptrons (RMLP) [111], a dynamic extension of well-known feed-forward layered networks to classify network data into anomalous and

normal. An RMLP network has the ability to encode temporal information. They develop an incremental kernel principal components algorithm to preprocess the data that goes into the neural network and obtain effective results.

4.6.3 Fuzzy Set Theoretic Techniques

Fuzzy network intrusion detection systems exploit fuzzy rules to determine the likelihood of specific or general network attacks [42, 60]. A fuzzy input set can be defined for traffic in a specific network.

Tajbakhsh et al. [145] describe a novel method for building classifiers using fuzzy association rules and use it for network intrusion detection. The fuzzy association rule sets are used to describe different classes: normal and anomalous. Such fuzzy association rules are *class association rules* where the consequents are specified classes. Whether a training instance belongs to a specific class is determined using matching metrics proposed by the authors. The fuzzy association rules are induced using normal training samples. A test sample is classified as normal if the compatibility of the rule set generated is above a certain threshold; those with lower compatibility are considered anomalous. The authors also propose a new method to speed up the rule induction algorithm by reducing items from extracted rules.

Mabu et al. report a novel fuzzy class-association-rule mining method based on genetic network programming (GNP) for detecting network intrusions [93]. GNP is an evolutionary optimization technique, which uses directed graph structures instead of strings in standard genetic algorithms, leading to enhanced representation ability with compact descriptions derived from possible node reusability in a graph. Xian et al. [155] present a novel unsupervised fuzzy clustering method based on clonal selection for anomaly detection. The method is able to obtain global optimal clusters more quickly than competing algorithms with greater accuracy.

4.6.4 Rough Set-Based Techniques

A rough set is an approximation of a crisp set (i.e., a regular set) in terms of a pair of sets that are its lower and upper approximations. In the standard and original version of rough set theory [113], the two approximations are crisp sets, but in other variations the approximating sets may be fuzzy sets. The mathematical framework of rough set theory enables modeling of relationships with a minimum number of rules.

Rough sets have two useful features [23]: (i) enabling learning with small-size training datasets and (ii) overall simplicity. They can be applied to anomaly detection by modeling normal behavior in network traffic. For example, Chimphlee et al. [34] present a Fuzzy Rough C-means clustering technique for network intrusion detection by integrating fuzzy set theory and rough set theory to achieve high detection rate.

Adetunmbi et al. [2] use rough sets and a k-NN classifier to detect network intrusions with high detection rate and low false alarm rate. Chen et al. present a two-step classifier for network intrusion detection [31]. Initially, it uses rough set theory for feature reduction and then a support vector machine classifier for final classification. They obtain 89% accuracy on network anomaly data.

4.6.5 Ant Colony and Artificial Immune Systems

Ant colony optimization [43] and related algorithms are probabilistic techniques for solving computational problems which can be reformulated to find optimal paths through graphs. The algorithms are based on the behavior of ants seeking a path between their colony and a source of food.

Gao et al. [57] use ant colony optimization for feature selection for an SVM classifier for network intrusion detection. The features are represented as graph nodes with the edges between them denoting the addition of the next feature. Ants traverse the graph to add nodes until the stopping criterion is encountered.

Artificial immune systems (AIS) represent a computational method inspired by the principles of the human immune system. The human immune system is adept at performing anomaly detection. Visconti and Tahayori [151] present a performance-based AIS for detecting individual anomalous behavior. It monitors the system by analyzing the set of parameters to provide general information on its state. Interval type-2 fuzzy set paradigm is used to dynamically generate system status.

4.6.6 Soft Computing Systems

RT-UNNID [4] is an IDS developed based on ANN. This system is capable of intelligent real-time intrusion detection using unsupervised neural networks (UNN). The architecture of RT-UNNID is given in Fig. 4.13. The first module captures and preprocesses the real-time network traffic data for the TCP, UDP, and ICMP protocols. It also extracts the numeric features and converts them into binary or normalized form. The converted data is sent to the UNN-based detection engine that uses adaptive resonance theory(ART) and self-organizing map(SOM) [25, 80] neural networks. Finally, the output of the detection engine is sent to the responder for recording in the user's system log file and to generate alarm when detecting attacks. RT-UNNID can work in real time to detect known and unknown attacks in network traffic with high detection rate.

NSOM (network self-organizing maps) [83] is a network IDS developed using self-organizing maps (SOM). It detects anomalies by quantifying the usual or acceptable behavior and flags irregular behavior as potentially intrusive. To classify real-time traffic, it uses a structured SOM. It continuously collects network data from a network port, preprocesses that data, and selects the features necessary for

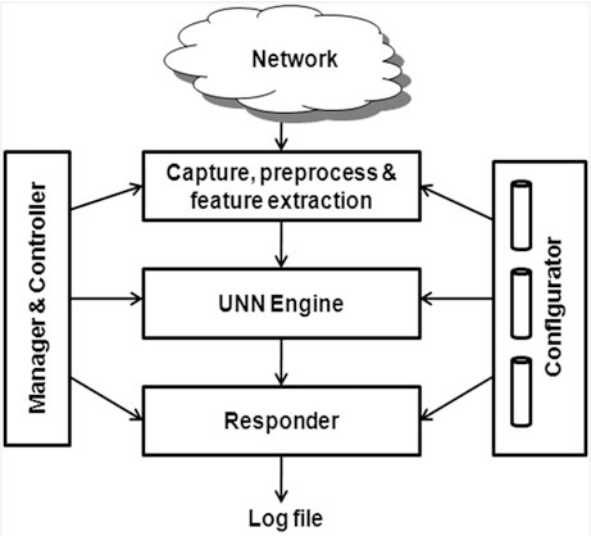


Fig. 4.13 Architecture of RT-UNNID system

classification. Then it starts the classification process—a chunk of packets at a time—and then sends the resulting classification to a graphical tool that portrays the activities that are taking place on the network port dynamically as it receives more packets. The hypothesis is that routine traffic that represents normal behavior would be clustered around one or more cluster centers, and any irregular traffic representing abnormal and possibly suspicious behavior would be clustered in addition to the normal traffic clustering. The system is capable of classifying regular vs. irregular and possibly intrusive network traffic for a given host.

POSEIDON (Payl Over Som for Intrusion DetectiON) [19] is a two-tier network intrusion detection system. The first tier consists of a self-organizing map (SOM) and is used exclusively to classify payload data. The second tier consists of a light modification of the PAYL system [153]. Tests using the DARPA99 dataset show a higher detection rate and lower number of false positives than PAYL and PHAD [95].

NFIDS [101] is a neuro-fuzzy anomaly-based network intrusion detection system. It comprises three tiers. Tier I contains several intrusion detection agents (IDAs). IDAs are IDS components that monitor the activities of a host or a network and report the abnormal behavior to Tier II. Tier II agents detect the network status of a LAN based on the network traffic that they observe as well as the reports from the Tier I agents within the LAN. Tier III combines higher-level reports, correlates data, and sends alarms to the user interface. There are four main types of agents in this system: TCPAgent, which monitors TCP connections between hosts and on the network; UDPAgent, which looks for unusual traffic involving UDP data; ICMPAgent, which monitors ICMP traffic; and PortAgent, which looks for unusual services in the network.

FIRE (fuzzy intrusion recognition engine) [42] is an anomaly-based intrusion detection system that uses fuzzy logic to assess whether malicious activity is taking place on a network. The system combines simple network traffic metrics with fuzzy rules to determine the likelihood of specific or general network attacks. Once the metrics are available, they are evaluated using a fuzzy set theoretic approach. The system takes on fuzzy network traffic profiles as inputs to its rule set and report maliciousness.

Advantages of soft computing-based anomaly detection techniques include the following:

- Due to the adaptive nature of ANNs, it is possible to train and test instances incrementally. Multilevel neural network techniques are more efficient than single-level neural networks.
- Unsupervised learning using competitive neural networks is effective in data clustering, feature extraction, and similarity detection.
- Rough sets are useful in resolving inconsistency in the dataset and to generate a minimal, nonredundant, and consistent rule set.

Some of the disadvantages of soft computing techniques are pointed below:

- Over-fitting may happen during neural network training.
- If a credible amount of normal traffic data is not available, the training of the techniques becomes very difficult.
- Most methods have scalability problems.
- Rough set-based rule generation suffers from proof of completeness.
- In fuzzy association rule-based techniques, reduced, relevant rule subset identification and dynamic rule updation at runtime are a difficult task.

Table 4.5 gives a comparison of several soft computing-based anomaly detection techniques.

4.7 Knowledge-Based Techniques and Systems

In knowledge-based techniques, network or host events are checked against predefined rules or patterns of attack. The goal is to represent the known attacks in a generalized fashion so that handling of actual occurrences becomes easier. Examples of knowledge-based methods are expert systems, rule-based, ontology-based, logic-based, and state-transition analysis [106, 120, 133, 157].

These techniques search for instances of known attacks, by attempting to match with predetermined attack representations. The search begins like other intrusion detection techniques, with a complete lack of knowledge. Subsequent matching of activities against a known attack helps acquire knowledge and enter into a region with higher confidence. Finally, it can be shown that an event or activity has reached maximum anomaly score.

A few prominent knowledge-based network anomaly detection techniques and NIDS are given below:

Table 4.5 Comparison of soft computing-based network anomaly detection techniques

Author (s)	Year of publication	No. of parameters	w	x	y	Data types	Dataset used	z	Detection method
Cannady [24]	2000	2	O	N	P	Numeric	Real-life	C_2	CMAC-based model
Balajinath and Raghavan [11]	2001	3	O	N	O	Categorical	User command	C_4	Behavior model
Lee and Heinbuch [86]	2001	3	C	N	P	–	Simulated data	C_2	TNNID model
Xian et al. [155]	2005	3	C	N	P	Numeric	KDDcup99	C_1	Fuzzy k-means
Amini et al. [4]	2006	2	C	R	P	Categorical	KDDcup99, Real-life	C_1	RT-UNNID system
Chimphlee et al. [34]	2006	3	C	N	P	Numeric	KDDcup99	C_1	Fuzzy rough C-means
Liu et al. [89]	2007	2	C	N	P	Numeric	KDDcup99	C_1	HPCANN model
Adetunmbi et al. [2]	2008	2	C	N	P	Numeric	KDDcup99	C_1	LEM2 and K-NN
Chen et al. [31]	2009	3	C	N	P	Numeric	DARPA98	C_2	RST-SVM technique
Mabu et al. [93]	2011	3	C	N	P	Numeric	KDDcup99	C_1	Fuzzy ARM-based on GNP
Visconti and Tahayori [151]	2011	2	O	N	P	Numeric	Real-life	C_2	Interval type-2 fuzzy set
Geramiraz et al. [60]	2012	2	O	N	P	Numeric	KDDcup99	C_1	Fuzzy rule-based model

w indicates centralized (C) or distributed (D) or others (O), *x* the nature of detection as real time (R) or non-real time (N), *y* characterizes packet-based (P) or flow-based (F) or hybrid (H) or others (O), *z* represents the list of attacks handled: C_1 all attacks, C_2 denial of service, C_3 probe, C_4 user to root, and C_5 remote to local

4.7.1 Expert Systems and Rule-Based Techniques

The expert system approach is one of the most widely used knowledge-based methods [6, 40]. An expert system, in the traditional sense, is a rule-based system, with or without an associated knowledge base. An expert system has a rule engine that matches rules against the current state of the system and depending on the results of matching, fires one or more rules.

To scale to large networks that collect flow statistics ubiquitously, Duffield et al. [44] use the machine learning algorithm called Adaboost [130] to translate packet level signatures to work with flow-level statistics. The algorithm is used to correlate packet and flow information. In particular, the authors associate packet level network alarms with a feature vector they create from flow records on the same traffic. They create a set of rules using flow information with features similar to those used in Snort rules. They also add numerical features such as the number of packets of a specific kind flowing within a certain time period. Duffield et al. train Adaboost on concurrent flow and packet traces. They evaluate the system using real-time network traffic data with more than a billion flows over 29 days and show that their performance is comparable to Snort's with flow data.

Prayote and Compton [121] present an approach to anomaly detection that attempts to address the brittleness problem in which an expert system makes a decision that human common sense would recognize as impossible. They use a technique called *prudence* [46], in which for every rule, the upper and lower bounds of each numerical variable in the data seen by the rule are recorded, as well as a list of values seen for enumerated variables. The expert system raises a warning when a new value or a value outside the range is seen in a data instance. They improve the approach by using a simple probabilistic technique to decide if a value is an outlier. When working with network anomaly data, the authors partition the problem space into smaller subspaces of homogeneous traffic, each of which is represented with a separate model in terms of rules. The authors find that this approach works reasonably well for new subspaces when little data has been observed. They claim 0% false-negative rate in addition to very low false-positive rate.

Scheirer and Chuah [131] report a syntax-based scheme that uses variable-length partition with multiple break marks to detect many polymorphic worms. The prototype is the first NIDS that provides semantics-aware capability and can capture polymorphic shell codes with additional stack sequences and mathematical operations.

4.7.2 Ontology and Logic-Based Techniques

It is possible to model attack signatures using expressive logic structure in real time by incorporating constraints and statistical properties. Naldurg et al. [103] present a framework for intrusion detection based on temporal logic specification. Intrusion

patterns are specified as formulae in an expressively rich and efficiently monitorable logic called EAGLE and evaluated using DARPA log files.

Estevez-Tapiador et al. [51] describe a finite state machine (FSM) methodology, where a sequence of states and transitions among them are used to model network protocols. If the specifications are complete enough, the model is able to detect illegitimate behavioral patterns effectively. Shabtai et al. [136] describe an approach for detecting previously un-encountered malware targeting mobile devices. Time-stamped security data is continuously monitored within the target mobile devices like smart phones and PDAs. Then it is processed by the knowledge-based temporal abstraction (KBTA) methodology. The authors evaluate the KBTA model using a lightweight host-based intrusion detection system, combined with central management capabilities for Android-based mobile phones.

Hung and Liu [70] use ontologies as a way of describing the knowledge of a domain, expressing the intrusion detection system much more in terms of the end user domain. Ontologies are used as a conceptual modeling tool, allowing a non-expert person to model an intrusion detection application using the concepts of intrusion detection more intuitively.

A comparison of knowledge-based anomaly detection techniques is given in Table 4.6.

The main advantages of knowledge-based anomaly detection methods include the following:

- These techniques are robust and flexible.
- These techniques have high detection rate, if a substantial knowledge base can be acquired properly about attacks as well as normal instances.

Some disadvantages of knowledge-based techniques are listed below:

Table 4.6 Comparison of knowledge-based network anomaly detection techniques

Author (s)	Year of publication	No. of parameters	w	x	y	Data types	Dataset used	z	Detection method
Noel et al. [106]	2002	–	O	N	O	–	–	–	Attack guilt model
Sekar et al. [133]	2002	3	O	N	P	Numeric	DARPA99	C_1	Specification-based model
Tapiador et al. [51]	2003	3	C	N	P	Numeric	Real-life	C_2	Markov chain model
Hung and Liu [70]	2008	–	O	N	P	Numeric	KDDcup99	C_1	Ontology-based
Shabtai et al. [136]	2010	2	O	N	O	–	Real-life	C_2	Incremental KBTA

w indicates centralized (C) or distributed (D) or others (O), *x* the nature of detection as real time (R) or non-real time (N), *y* characterizes packet-based (P) or flow-based (F) or hybrid (H) or others (O), *z* represents the list of attacks handled: C_1-all attacks, C_2-denial of service, C_3-probe, C_4-user to root, and C_5-remote to local

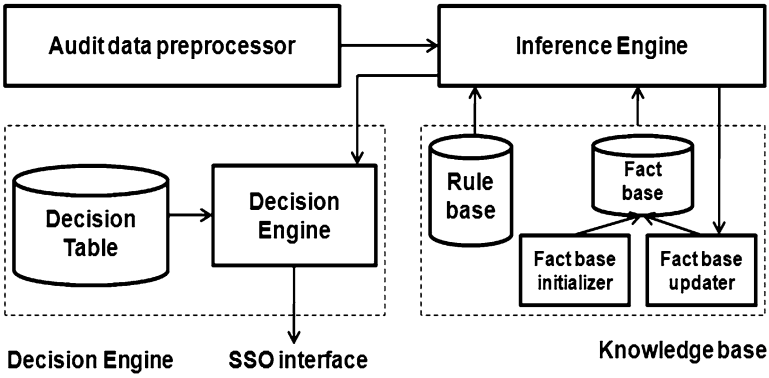


Fig. 4.14 Architecture of STAT system

- The development of high-quality knowledge is often difficult and time-consuming.
- Due to nonavailability of biased normal and attack data, such a method may generate a large number of false alarms.
- Such a method may not be able to detect rare or unknown attacks.
- Dynamic updation of rule or knowledge base is a costly affair.

4.7.3 Knowledge-Based Systems

An example knowledge-based system is STAT (state transition analysis tool) [72]. Its architecture is given in Fig. 4.14. It models traffic data as a series of state changes that lead from secure state to a compromised target state. STAT is composed of three main components: knowledge base, inference engine, and decision engine. The audit data preprocessor reformats the raw audit data to send as input to the inference engine. The inference engine monitors the state transitions extracted from the preprocessed audit data and then compares these states with the states available within the knowledge base. The decision engine monitors the improvement of the inference engine for matching accuracy of the state transitions. It also specifies the action(s) to be taken based on results of the inference engine and the decision table. Finally, the decision results are sent to the SSO (Site Security Officer) interface for action. It can detect cooperative attackers and attacks across user sessions well.

SNORT [125] is a quintessentially popular rule-based IDS. This open-source IDS matches each packet it observes against a set of rules. The antecedent of a Snort rule is a Boolean formula composed of predicates that look for specific values of fields present in IP headers, in transport headers, and in the payload. Thus, Snort rules identify attack packets based on IP addresses, TCP or UDP port numbers, ICMP codes or types, and contents of strings in the packet payload. Snort's rules

are arranged into priority classes based on potential impact of alerts that match the rules. Snort's rules have evolved over its history of 15 years. Each Snort rule has associated documentation with the potential for false positives and negatives, together with corrective actions to be taken when the rule raises an alert. Snort rules are simple and easily understandable. Users can contribute rules when they observe new types of anomalous or malicious traffic. Currently, Snort has over 20,000 rules, inclusive of those submitted by users.

An intrusion detection system like Snort can run on a general purpose computer and can try to inspect all packets that go through the network. However, monitoring packets comprehensively in a large network is obviously an expensive task since it requires fast inspection on a large number of network interfaces. Many hundreds of rules may have to be matched concurrently, making scaling almost impossible.

4.8 Combination Learners: Techniques and Systems

In this section, we present a few techniques and systems which use combinations of multiple techniques, usually classifiers.

4.8.1 Ensemble-Based Techniques

The idea behind the ensemble methodology is to weigh several individual classifiers and combine them to obtain an overall classifier that outperforms every one of them [20, 61, 107, 118, 126]. These techniques weigh the individual opinions and combine them to reach a final decision. The ensemble-based methods are categorized based on the approaches used. Three main approaches to develop ensembles are (i) bagging, (ii) boosting, and (iii) stack generalization. *Bagging* (bootstrap aggregating) increases classification accuracy by creating an improved composite classifier into a single prediction by combining the outputs of learnt classifiers. *Boosting* builds an ensemble incrementally by training misclassified instances obtained from the previous model. *Stack generalization* achieves the high generalization accuracy by using output probabilities for every class label from the base-level classifiers.

Chebrolu et al. [30] present an ensemble approach by combining two classifiers, Bayesian networks (BN) and Classification and Regression Trees (CART) [22, 54]. A hybrid architecture for combining different feature selection algorithms for real-world intrusion detection is also incorporated for getting better results. Perdisci et al. [117] construct a high-speed payload anomaly IDS using an ensemble of one-class SVM classifiers intended to be accurate and hard to evade.

Folino et al. [53] introduce a distributed data mining algorithm to improve detection accuracy when classifying malicious or unauthorized network activity using genetic programming (GP) extended with the ensemble paradigm. Their

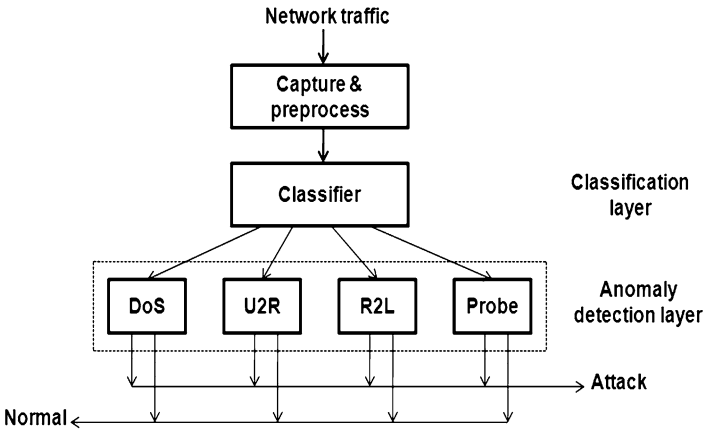


Fig. 4.15 Architecture of Octopus-IIDS system

data is distributed across multiple autonomous sites, and the learner component acquires useful knowledge from data in a cooperative way and uses network profiles to predict abnormal behavior with better accuracy. Nguyen et al. [105] build an individual classifier using both the input feature space and an additional subset of features given by k-means clustering. The ensemble combination is calculated based on the classification ability of classifiers on different local data segments given by k-means clustering.

4.8.2 Ensemble-Based Systems

The paradigm of multiple classifier systems (MCS) has also been used to build misuse detection IDSs. Classifiers trained on different feature subsets are combined to achieve better classification accuracy than the individual classifiers. In such a NIDS, network traffic is serially processed by each classifier. At each stage, a classifier may either decide for one attack class or send the pattern to another stage, which is trained on more difficult cases. Reported results show that an MCS improves the performance of IDSs based on statistical pattern recognition techniques. For example, Octopus-IIDS [94] is an ensemble-based IDS. The architecture of this system is shown in Fig. 4.15. It is developed using two types of neural networks (Kohonen and Support Vector Machines). The system is composed of two layers: classifier and anomaly detection. The classifier is responsible for capturing and preprocessing of network traffic data. It classifies the data into four main categories: DoS, probe, U2R, and R2L. A specific class of attack is identified in the anomaly detection layer. The authors claim that the IDS works effectively in small-scale networks.

CAMNEP [124] is a fast prototype agent-based NIDS designed for high-speed networks. It integrates several anomaly detection techniques and operates on a collective trust model within a group of collaborative detection agents. The anomalies are used as input for trust modeling. Aggregation is performed by extended trust models of generalized situated identities, represented by a set of observable features. The system is able to perform real-time surveillance of gigabit networks.

McPAD (multiple classifier payload-based anomaly detector) [116] is an effective payload-based anomaly detection system that consists of an ensemble of one-class classifiers. It is very accurate in detecting network attacks that bear some form of shell-code in the malicious payload. This detector performs well even in the case of polymorphic attacks. Furthermore, the authors tested their IDS with advanced polymorphic blending attacks and showed that even in the presence of such sophisticated attacks, it is able to obtain a low false-positive rate.

An ensemble technique is good because it obtains higher accuracy than the individual techniques. The following are the major advantages:

- Even if the individual classifiers are weak, the ensemble methods perform well by combining multiple classifiers.
- Ensemble methods can scale for large datasets.
- Ensemble classifiers need a set of controlling parameters that are comprehensive and can be easily tuned.
- Among existing approaches, Adaboost and Stack generalization are more effective because they can exploit the diversity in predictions by multiple base-level classifiers.

Here are some disadvantages of ensemble-based techniques:

- Selecting a subset of consistently performing and unbiased classifiers from a pool of classifiers is difficult.
- It is difficult to obtain real-time performance.

A comparison of ensemble-based network anomaly detection techniques is given in Table 4.7.

4.8.3 Fusion-Based Techniques

A classification technique may have to work with several disparate data sources. A suitable combination of these data sources may be used to train a classifier. This is known as data fusion. Several fusion-based techniques have been applied to network anomaly detection [62, 110, 137, 158, 170]. Such techniques may perform fusion at (i) data level, (ii) feature level, and (iii) decision level. Some methods only address the issue of operating in a space of high dimensionality with features divided into semantic groups. Others attempt to combine classifiers trained on different features divided based on hierarchical abstraction levels or the type of information contained.

Table 4.7 Comparison of ensemble-based network anomaly detection techniques

Author (s)	Year of publication	Combination strategy	w	x	y	Data types	Dataset used	z	Detection method
Chebrolu et al. [30]	2005	Weighted voting	O	N	P	Numeric	KDDcup99	C_1	Class specific ensemble model
Perdisci et al. [117]	2006	Majority voting	O	N	Pay	–	Operational points	Synthetic intrusions	One-class classifier model
Borji [20]	2007	Majority voting	O	N	P	Numeric	DARPA98	C_1	Heterogeneous classifiers model
Perdisci et al. [116]	2009	Min and Max probability	O	R	Pay	–	DARPA98	C_1	McPAD model
Folino et al. [53]	2010	Weighted majority voting	O	N	P	Numeric	KDDcup99	C_1	GEdIDS model
Noto et al. [107]	2010	Information theoretic	O	N	–	Numeric	UCI	None	FRaC model
Nguyen et al. [105]	2011	Majority voting	O	N	P	Numeric	KDDcup99	C_1	Cluster ensemble
Khreich et al. [78]	2012	Learn and combine	O	N	pay	Numeric	UNM	C_4	EoHMMs model

w indicates centralized (C) or distributed (D) or others (O), *x* the nature of detection as real time (R) or non-real time (N), *y* characterizes packet-based (P) or flow-based (F) or payload-based (pay) or hybrid (H) or others (O), *z* represents the list of attacks handled: C_1 all attacks, C_2 denial of service, C_3 probe, C_4 user to root, and C_5 remote to local

Giacinto et al. [62] develop a pattern recognition approach to network intrusion detection employing a fusion of multiple classifiers. Five different decision fusion methods are assessed by experiments, and their performance compared. Shifflet [137] discusses a platform that enables a multitude of techniques to work together toward creating a more realistic fusion model of the state of a network, able to detect malicious activity effectively. A heterogeneous data-level fusion for network anomaly detection is added by Chatzigiannakis et al. [29]. They use the Dempster-Shafer theory of evidence and principal components analysis for developing the technique.

dLEARNIN [110] is an ensemble of classifiers that combines information from multiple sources. It is explicitly tuned to minimize the cost of errors. dLEARNIN is shown to achieve state-of-the-art performance, better than competing algorithms. The cost minimization strategy, dCMS, attempts to minimize the cost to a significant level. Gong et al. [64] contribute a neural network-based data fusion method for intrusion data analysis and pruning to filter information from multiple sensors to get high detection accuracy.

4.8.4 Fusion-Based Systems

HMMPayl [8] is an example of fusion-based IDS, where the payload is represented as a sequence of bytes and the analysis is performed using hidden Markov models (HMM). The algorithm extracts features and uses HMM to guarantee the same expressive power as that of n-gram analysis while overcoming its computational complexity. HMMPayl follows the Multiple Classifiers System paradigm to provide better classification accuracy, to increase the difficulty of evading the IDS, and to mitigate the weaknesses due to a nonoptimal choice of HMM parameters.

Some advantages of fusion techniques are given below:

- Data fusion is effective in increasing timeliness of attack identification and in reducing false alarm rates.
- Decision-level fusion with appropriate training data usually yields high detection rate.

Some of the drawbacks are given below:

- The computational cost is high for rigorous training on the samples.
- Feature-level fusion is a time-consuming task. The biases of the base classifiers affect the fusion process.
- Building a single hypotheses for different classifiers is a difficult task.

A comparison of fusion-based network anomaly detection is given in Table 4.8.

Table 4.8 Comparison of fusion-based network anomaly detection techniques

Author (s)	Year of publication	Fusion level	w	x	y	Data types	Dataset used	z	Detection method
Giacinto et al. [62]	2003	Decision	O	N	P	Numeric	KDDcup99	C_1	MCS Model
Shifflet [137]	2005	Data	O	N	O	–	–	None	HSPT algorithm
Chatzigiannakis et al. [29]	2007	Data	C	N	P	–	NTUA, GRNET	C_2	D-S algorithm
Parikh and Chen [110]	2008	Data	C	N	P	Numeric	KDDcup99	C_1	dLEARNIN system
Gong et al. [64]	2010	Data	C	N	P	Numeric	KDDcup99	C_1	IDEA model
Ariu et al. [8]	2011	Decision	C	R	Pay	–	DARPA98, real-life	C_1	HMMPayl model
Yan and Shao [158]	2012	Decision	O	N	F	Numeric	Real time	C_2, C_3	EWMA model

w indicates centralized (C) or distributed (D) or others (O), *x* the nature of detection as real time (R) or non-real time (N), *y* characterizes packet-based (P) or flow-based (F) or payload-based (pay) or hybrid (H) or others (O), *z* represents the list of attacks handled: C_1-all attacks, C_2-denial of service, C_3-probe, C_4-user to root, and C_5-remote to local

4.8.5 Hybrid Techniques

Most current network intrusion detection systems employ either misuse detection or anomaly detection. However, misuse detection cannot detect unknown intrusions, and anomaly detection usually has high false-positive rate [9]. To overcome the limitations of the techniques, hybrid methods are developed by exploiting features from several network anomaly detection approaches [10, 109, 165]. Hybridization of several methods increases performance of IDSs.

A hybrid approach to host security that prevents binary code injection attacks known as the FLIPS (Feedback Learning IPS) model is proposed by [90]. It incorporates three major components: an anomaly-based classifier, a signature-based filtering scheme, and a supervision framework that employs Instruction Set Randomization (ISR). Capturing the injected code allows FLIPS to construct signatures for zero-day exploits. Peddabachigari et al. [115] present a hybrid approach that combines decision trees (DT) and SVMs as a hybrid hierarchical intelligent system model (DTSVM) for intrusion detection. It maximizes detection accuracy and minimizes computational complexity.

Zhang et al. [166] propose a systematic framework that applies a data mining algorithm called random forests to build a misuse, anomaly, and hybrid network-based IDS. The hybrid detection system improves detection performance by combining the advantages of both misuse and anomaly detection. Tong et al. [147] discuss a hybrid RBF/Elman neural network model that can be employed for both anomaly detection and misuse detection. It can detect temporally dispersed and collaborative attacks effectively because of its memory of past events.

4.8.6 Hybrid Systems

An intelligent hybrid IDS model based on neural networks is reported in [162]. The model is flexible, extended to meet different network environments, and improves detection performance and accuracy. Selim et al. [134] report a hybrid intelligent IDS to improve the detection rate for known and unknown attacks. It consists of multiple levels: hybrid neural networks and decision trees. The technique is evaluated using the NSL-KDD dataset and results were promising.

RT-MOVICAB-IDS, a hybrid intelligent IDS, is introduced in [68]. It combines ANN and CBR (case-based reasoning) within a multi-agent system (MAS) to detect intrusion in dynamic computer networks. The dynamic real-time multi-agent architecture allows the addition of prediction agents (both reactive and deliberative). In particular, two of the deliberative agents deployed in the system incorporate a temporal-bounded CBR system. This upgraded CBR is based on an anytime approximation, which allows the adaptation of this paradigm to real-time requirements.

Advantages of hybrid techniques include the following:

- Such a method exploits major features of both signature and anomaly-based network anomaly detection.
- Such methods can handle both known and unknown attacks.

Drawbacks include the following:

- Lack of appropriate hybridization may lead to high computational cost.
- Dynamic updation of rule or profile or signature still remains difficult.

Table 4.9 gives a comparison of a few hybrid network anomaly detection techniques.

We perform a comparison of the anomaly-based network intrusion detection systems that we have discussed throughout this chapter based on parameters such as mode of detection (host-based, network-based, or both), detection approach (misuse, anomaly, or both), nature of detection (online or offline), nature of processing (centralized or distributed), data gathering mechanism (centralized or distributed), and approach of analysis. A comparison chart is given in Table 4.10.

4.9 Observations and Chapter Summary

This chapter presents several network traffic anomaly detection techniques and systems under different categories. Though many techniques and systems have been developed for detection of network-wide traffic anomaly detection, it is still necessary to develop effective and efficient techniques and systems to handle the growing threats of cyber attacks. We observe the following few points:

- Each detection technique and system works well in certain scenarios. But no technique or system is capable of working well in all scenarios. It is because the nature of network traffic changes constantly, and the performance of a technique depends on the deployment point in the network.
- Statistical techniques and systems work well if the predicted model is built by observing the real-time traffic of an enterprise network. For example, normal traffic follows the Gaussian distribution, and attack traffic usually follows the Poisson distribution. But obtaining a density function that works well and finding parameter values that are optimal are difficult.
- Classification, clustering, and outlier-based techniques mostly all require an appropriate proximity measure. For high-dimensional datasets, it is difficult to compute proximity considering all features and get good results efficiently. Therefore it is necessary to reduce the number of features. Identification of a suitable subset of features in a large high-dimensional dataset is expensive and time-consuming. For real-time network traffic anomaly detection, time complexity is important and should be low. Training is a crucial in building a system that detects anomalies in network-wide traffic. If training is offline, it

Table 4.9 Comparison of hybrid network anomaly detection techniques

Author (s)	Year of publication	No. of parameters	w	x	y	Data types	Dataset used	z	Detection method
Locasto et al. [90]	2005	2	C	R	P	–	Real-life	C_2	FLIPS model
Zhang and Zulkernine [165]	2006	2	C	N	P	Numeric	KDDcup99	C_1	Random forest-based hybrid algorithm
Peddabachigari et al. [115]	2007	2	C	N	P	Numeric	KDDcup99	C_1	DT-SVM hybrid model
Zhang et al. [166]	2008	2	C	N	P	Numeric	KDDcup99	C_1	RFIDS model
Aydin et al. [10]	2009	3	C	N	P	–	DARPA98, IDEVAL	C_1	Hybrid signature-based IDS model
Tong et al. [147]	2009	1	C	N	P	Numeric	DARPA-BSM	C_1	Hybrid RBF/Elman NN
Yu [162]	2010	1	C	N	–	–	–	–	Hybrid NIDS
Arumugam et al. [9]	2010	–	C	N	P	Numeric	KDDcup99	C_1	Multi-stage hybrid IDS
Selim et al. [134]	2011	–	C	N	P	Numeric	KDDcup99	C_1	Hybrid multilevel IDS model
Panda et al. [109]	2012	2	C	N	P	Numeric	NSL-KDD, KDDcup99	C_1	DTFF and FFNN

w indicates centralized (C) or distributed (D) or others (O), *x* the nature of detection as real time (R) or non-real time (N), *y* characterizes packet-based (P) or flow-based (F) or hybrid (H) or others (O), *z*-represents the list of attacks handled: C_1 all attacks, C_2 denial of service, C_3 probe, C_4 user to root, and C_5 remote to local

Table 4.10 Comparison of existing NIDSs

Name of IDS	Year of publication	a	b	c	d	e	Approach
STAT [72]	1995	H	M	R	C	C	Knowledge-based
FIRE [42]	2000	N	A	N	C	C	Fuzzy Logic
ADAM [36]	2001	N	A	R	C	C	Classification
HIDE [169]	2001	N	A	R	C	D	Statistical
NSOM [83]	2002	N	A	R	C	C	Neural network
MINDS [48]	2003	N	A	R	C	C	Clustering and outlier-based
NFIDS [101]	2003	N	A	N	C	C	Neuro fuzzy logic
N@G [143]	2003	H	B	R	C	C	Statistical
FSAS [138]	2006	N	A	R	C	C	Statistical
POSEIDON [19]	2006	N	A	R	C	C	SOM & modified PAYL
RT-UNNID [4]	2006	N	A	R	C	C	Neural network
DNIDS [82]	2007	N	A	R	C	C	CSI-KNN based
CAMNEP [124]	2008	N	A	R	C	C	Agent-based trust and reputation
McPAD [116]	2009	N	A	N	C	C	Multiple classifier
Octopus-IIDS [94]	2010	N	A	N	C	C	Neural network & SVM
HMMPayl [8]	2011	N	A	R	C	C	HMM model
RT-MOVICAB-IDS [68]	2011	N	A	R	C	C	Hybrid IDS

a represents the types of detection such as host-based (H) or network-based (N) or hybrid (H), *b* indicates the class of detection mechanism as misuse (M) or anomaly (A) or both (B), *c* denotes the nature of detection as real time (R) or non-real time (N), *d* characterizes the nature of processing as centralized (C) or distributed (D), *e* indicates the data gathering mechanism as centralized (C) or distributed (D)

does not affect detection efficiency. Training usually involves high cost. Testing or detecting should be preferably online.

- Knowledge-based techniques and systems build a rule database based on the existing attack information, and then it detects anomalies with respect to the built rule set. Building an optimal, nonredundant, and consistent rule database is a challenging task, considering all types of attacks.
- Soft computing techniques are applied when the decision of identifying an element of network traffic as anomalous or normal is not certain. Developing a membership function to rank network traffic elements as anomaly or legitimate allows one to make decisions in uncertain situations.
- Finally, combination learners are techniques to combine different methods at different levels such as feature, decision, and data. But such approaches are time-consuming and are generally not suitable for real-time detection.

This chapter has discussed network-wide traffic anomaly detection techniques and systems with pros and cons. We analyze detection techniques in terms of various performance parameters. We also discuss several systems that have been built for deployment as a detection system in enterprise networks.

References

1. Abbes, T., Bouhoula, A., Rusinowitch, M.: Efficient decision tree for protocol analysis in intrusion detection. Int. J. Secur. Netw. **5**(4), 220–235 (2010)
2. Adetunmbi, A.O., Falaki, S.O., Adewale, O.S., Alese, B.K.: Network intrusion detection based on rough set and k-nearest neighbour. Int. J. Comput. ICT Res. **2**(1), 60–66 (2008)
3. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann, San Francisco (1994)
4. Amini, M., Jalili, R., Shahriari, H.R.: RT-UNNID: a practical solution to real-time network-based intrusion detection using unsupervised neural networks. Comput. Secur. **25**(6), 459–468 (2006)
5. Amiri, F., Yousefi, M.M.R., Lucas, C., Shakery, A., Yazdani, N.: Mutual information-based feature selection for intrusion detection systems. J. Netw. Comput. Appl. **34**(4), 1184–1199 (2011)
6. Anderson, D., Lunt, T.F., Javitz, H., Tamaru, A., Valdes, A.: Detecting Unusual Program Behaviour Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES). Tech. Rep. SRIO-CSL-95-06, Computer Science Laboratory, SRI International (1995)
7. Anscombe, F.J., Guttman, I.: Rejection of outliers. Technometrics **2**(2), 123–147 (1960)
8. Ariu, D., Tronci, R., Giacinto, G.: HMMPayl: an intrusion detection system based on hidden Markov models. Comput. Secur. **30**(4), 221–241 (2011)
9. Arumugam, M., Thangaraj, P., Sivakumar, P., Pradeepkumar, P.: Implementation of two class classifiers for hybrid intrusion detection. In: Proceedings of the International Conference on Communication and Computational Intelligence, pp. 486–490 (2010)
10. Aydin, M.A., Zaim, A.H., Ceylan, K.G.: A hybrid intrusion detection system design for computer network security. Comput. Electr. Eng. **35**(3), 517–526 (2009)
11. Balajinath, B., Raghavan, S.V.: Intrusion detection through learning behavior model. Comput. Commun. **24**(12), 1202–1212 (2001)
12. Bezdek, J.C., Ehrlich, R., Full, W.: FCM: the Fuzzy C-means clustering algorithm. Comput. Geosci. **10**(2-3), 191–203 (1984). doi:10.1016/0098-3004(84)90020-7
13. Bhuyan, M.H., Bhattacharyya, D., Kalita, J.: A multi-step outlier-based anomaly detection approach to network-wide traffic. Inf. Sci. **348**, 243–271 (2016)
14. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: NADO: network anomaly detection using outlier approach. In: Proceedings of the International Conference on Communication, Computing and Security, pp. 531–536. ACM, Odisha (2011)
15. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: RODD: an effective reference-based outlier detection technique for large datasets. In: Advanced Computing, vol. 133, pp. 76–84. Springer, Berlin (2011)
16. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Surveying port scans and their detection methodologies. Comp. J. **54**(10), 1565–1581 (2011)
17. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: An effective unsupervised network anomaly detection method. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics, pp. 533–539. ACM, New York (2012)
18. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: AOCD: an adaptive outlier based coordinated scan detection approach. Int. J. Netw. Secur. **14**(6), 339–351 (2012)
19. Bolzoni, D., Etalle, S., Hartel, P.H., Zambon, E.: POSEIDON: a 2-tier anomaly-based network intrusion detection system. In: Proceedings of the 4th IEEE International Workshop on Information Assurance, pp. 144–156 (2006)
20. Borji, A.: Combining heterogeneous classifiers for network intrusion detection. In: Proceedings of the 12th Asian Computing Science Conference on Advances in Computer Science: Computer and Network Security, pp. 254–260. Springer (2007)

21. Braynov, S., Jadliwala, M.: Detecting malicious groups of agents. In: Proceedings of the 1st IEEE Symposium on Multi-Agent Security and Survivability, pp. 90–99. IEEE CS (2004)
22. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and regression trees. Wadsworth and Brooks, Monterey (1984)
23. Cai, Z., Guan, X., Shao, P., Peng, Q., Sun, G.: A rough set theory based method for anomaly intrusion detection in computer network systems. Expert Syst. **20**(5), 251–259 (2003)
24. Cannady, J.: Applying CMAC-based on-line learning to intrusion detection. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, vol. 5, pp. 405–410 (2000)
25. Carpenter, G., Grossberg, S.: Adaptive resonance theory. In: The Handbook of Brain Theory and Neural Networks, pp. 87–90. MIT Press, Cambridge (2003)
26. Casas, P., Mazel, J., Owezarski, P.: Unsupervised network intrusion detection systems: detecting the unknown without knowledge. Comput. Commun. **35**(7), 772–783 (2012)
27. Chan, P.K., Mahoney, M.V., Arshad, M.H.: A Machine Learning Approach to Anomaly Detection. Tech. Rep. CS-2003-06, Department of Computer Science, Florida Institute of Technology (2003)
28. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**(3), 15:1–15:58 (2009)
29. Chatzigiannakis, V., Androulidakis, G., Pelechrinis, K., Papavassiliou, S., Maglaris, V.: Data fusion algorithms for network anomaly detection: classification and evaluation. In: Proceedings of the 3rd International Conference on Networking and Services, pp. 50–57. IEEE CS (2007)
30. Chebrolu, S., Abraham, A., Thomas, J.P.: Feature deduction and ensemble design of intrusion detection systems. Comput. Secur. **24**(4), 295–307 (2005)
31. Chen, R.C., Cheng, K.F., Chen, Y.H., Hsieh, C.F.: Using rough set and support vector machine for network intrusion detection system. In: Proceedings of the 1st Asian Conference on Intelligent Information and Database Systems, pp. 465–470. IEEE Computer Society, Washington, DC (2009)
32. Chen, Z., Chen, C.: A closed-form expression for static worm-scanning strategies. In: Proceedings of the IEEE International Conference on Communications, pp. 1573–1577. IEEE CS, Beijing (2008)
33. Chhabra, P., Scott, C., Kolaczyk, E.D., Crovella, M.: Distributed spatial anomaly detection. In: Proceedings of the 27th IEEE International Conference on Computer Communications, pp. 1705–1713 (2008)
34. Chimphlee, W., Abdullah, A.H., Noor, M.S.M., Srinoy, S., Chimphlee, S.: Anomaly-based intrusion detection using fuzzy rough clustering. In: Proceedings of the International Conference on Hybrid Information Technology, vol. 1, pp. 329–334. IEEE Computer Society, Washington, DC (2006)
35. Choo, K.K.R.: the cyber threat landscape: challenges and future research directions. Comput. Secur. **30**(8), 719–731 (2011)
36. Daniel, B., Julia, C., Sushil, J., Ningning, W.: ADAM: a testbed for exploring the use of data mining in intrusion detection. ACM SIGMOD Rec. **30**(4), 15–24 (2001)
37. Das, K., Schneider, J., Neill, D.B.: Anomaly pattern detection in categorical datasets. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 169–176. ACM (2008). doi:10.1145/1401890.1401915
38. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **1**(2), 224–227 (1979)
39. De Vivo, M., Carrasco, E., Isern, G., de Vivo, G.O.: A review of port scanning techniques. SIGCOMM Comput. Commun. Rev. **29**(2), 41–48 (1999)
40. Denning, D.E., Neumann, P.G.: Requirements and Model for IDES – A Real-time Intrusion Detection System. Tech. Rep. 83F83-01-00, Computer Science Laboratory, SRI International (1985)

41. Desforges, M.J., Jacob, P.J., Cooper, J.E.: Applications of probability density estimation to the detection of abnormal conditions in engineering. In: Proceedings of Institute of Mechanical Engineers, vol. 212, pp. 687–703 (1998)
42. Dickerson, J.E.: Fuzzy network profiling for intrusion detection. In: Proceedings of the 19th International Conference of the North American Fuzzy Information Processing Society, pp. 301–306. Atlanta (2000)
43. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. B Cybern. **26**(1), 29–41 (1996)
44. Duffield, N.G., Haffner, P., Krishnamurthy, B., Ringberg, H.: Rule-based anomaly detection on IP flows. In: Proceedings of the 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, pp. 424–432. IEEE press, Rio de Janeiro (2009)
45. Dunn, J.C.: Well separated clusters and optimal fuzzy partitions. J. Cybern. **4**(1), 95–104 (1974)
46. Edwards, G., Kang, B., Preston, P., Compton, P.: Prudent expert systems with credentials: managing the expertise of decision support systems. Int. J. Biomed. Comput. **40**(2), 125–132 (1995)
47. Ensafi, R., Park, J.C., Kapur, D., Crandall, J.R.: Idle port scanning and non-interference analysis of network protocol stacks using model checking. In: Proceedings of the 19th USENIX Security Symposium (2010)
48. Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P., Kumar, V., Srivastava, J.: Chapter 3: MINDS – Minnesota intrusion detection system. In: Next Generation Data Mining, pp. 1–21. CRC press (2004)
49. Eskin, E.: Anomaly detection over noisy data using learned probability distributions. In: Proceedings of the 7th International Conference on Machine Learning, pp. 255–262. Morgan Kaufmann Publishers Inc. (2000)
50. Eskin, E., Arnold, A., Prerau, M., Portnoy, L., Stolfo, S.: A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data. In: Applications of Data Mining in Computer Security. Kluwer Academic, Boston (2002)
51. Estevez-Tapiador, J.M., Garcya-Teodoro, P., Dyaz-Verdejo, J.E.: Stochastic protocol modeling for anomaly-based network intrusion detection. In: Proceedings of the 1st International Workshop on Information Assurance, pp. 3–12. IEEE CS (2003)
52. Falletta, V., Ricciato, F.: Detecting scanners: empirical assessment on 3G network. Int. J. Netw. Secur. **9**(2), 143–155 (2009)
53. Folino, G., Pizzuti, C., Spezzano, G.: An ensemble-based evolutionary framework for coping with distributed intrusion detection. Genet. Program. Evolvable Mach. **11**(2), 131–146 (2010)
54. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Mach. Learn. **29**(2–3), 131–163 (1997)
55. Gaddam, S.R., Phoha, V.V., Balagani, K.S.: K-Means+ID3: a novel method for supervised anomaly detection by cascading k-means clustering and id3 decision tree learning methods. IEEE Trans. Knowl. Data Eng. **19**(3), 345–354 (2007)
56. Gadge, J., Patil, A.A.: Port scan detection. In: Proceedings of 16th IEEE International Conference on Networks, pp. 1–6. IEEE Computer Society, Habitat World, IHC, New Delhi (2008)
57. Gao, H.H., Yang, H.H., Wang, X.Y.: Ant colony optimization based network intrusion feature selection and detection. In: Proceedings of the International Conference on Machine Learning and Cybernetics, vol. 6, pp. 3871–3875 (2005). doi:10.1109/ICMLC.2005.1527615
58. Garcia-Teodoro, P., Diaz-Verdejo, J., Macia-Fernandez, G., Vazquez, E.: Anomaly-based network intrusion detection: techniques, systems and challenges. Comput. Secur. **28**(1-2), 18–28 (2009)
59. Gates, C., McNutt, J.J., Kadane, J.B., Kellner, M.: Scan detection on very large networks using logistic regression modeling. In: Proceedings of the 11th IEEE Symposium on Computers and Communications, pp. 402–408. IEEE Computer Society, Pula-Cagliari, Sardinia (2006)

60. Geramiraz, F., Memaripour, A.S., Abbaspour, M.: Adaptive anomaly-based intrusion detection system using fuzzy controller. Int. J. Netw. Secur. **14**(6), 352–361 (2012)
61. Giacinto, G., Perdisci, R., Rio, M.D., Roli, F.: Intrusion detection in computer networks by a modular ensemble of one-class classifiers. Inf. Fusion **9**(1), 69–82 (2008)
62. Giacinto, G., Roli, F., Didaci, L.: Fusion of multiple classifiers for intrusion detection in computer networks. Pattern Recogn. Lett. **24**(12), 1795–1803 (2003)
63. Gogoi, P., Bhattacharyya, D.K., Borah, B., Kalita, J.K.: A survey of outlier detection methods in network anomaly identification. Comput. J. **54**(4), 570–588 (2011)
64. Gong, W., Fu, W., Cai, L.: A neural network based intrusion detection data fusion model. In: Proceedings of the 3rd International Joint Conference on Computational Science and Optimization, vol. 2, pp. 410–414. IEEE CS (2010)
65. Gyorgy, S.U., Gyorgy, J.S., Hui, X.: Scan detection: a data mining approach. In: Proceedings of the Sixth SIAM International Conference on Data Mining, pp. 118–129. SIAM, Sutton Place Hotel, Newport Beach (2005)
66. Haykin, S.: Neural Networks. Prentice Hall, New Jersey (1999)
67. Heberlein, T., Dias, G., Levitt, K., Mukherjee, B., Wood, J., Wolber, D.: A network security monitor. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, pp. 296–304. IEEE Computer Society, Oakland (1990)
68. Herrero, A., Navarro, M., Corchado, E., Julian, V.: RT-MOVICAB-IDS: addressing real-time intrusion detection. Futur. Gener. Comput. Syst. **29**(1), 250–261 (2011)
69. Hubert, L., Schultz, J.: Quadratic assignment as a general data analysis strategy. Br. J. Math. Stat. Psychol. **29**(2), 190–241 (1976)
70. Hung, S.S., Liu, D.S.M.: A user-oriented ontology-based approach for network intrusion detection. Comput. Stand. Interfaces **30**(1-2), 78–88 (2008)
71. hybrid@hotmail.com: Distributed information gathering. Phrack Mag. Article 9 **9**(55) (1999)
72. Ilgun, K., Kemmerer, R.A., Porras, P.A.: State transition analysis: a rule-based intrusion detection approach. IEEE Trans. Softw. Eng. **21**(3), 181–199 (1995)
73. Jiang, S., Song, X., Wang, H., Han, J.J., Li, Q.H.: A clustering-based method for unsupervised intrusion detections. Pattern Recogn. Lett. **27**(7), 802–810 (2006)
74. Jung, J., Paxson, V., Berger, A.W., Balakrishnan, H.: Fast portscan detection using sequential hypothesis testing. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 211–225. IEEE Computer Society, Oakland (2004)
75. Kang, I., Jeong, M.K., Kong, D.: A differentiated one-class classification method with applications to intrusion detection. Expert Syst. Appl. **39**(4), 3899–3905 (2012)
76. Khan, L., Awad, M., Thuraisingham, B.: A new intrusion detection system using support vector machines and hierarchical clustering. VLDB J. **16**(4), 507–521 (2007)
77. Khan, M.S.A.: Rule based network intrusion detection using genetic algorithm. Int. J. Comput. Appl. **18**(8), 26–29 (2011)
78. Khreich, W., Granger, E., Miri, A., Sabourin, R.: Adaptive ROC-based ensembles of HMMs applied to anomaly detection. Pattern Recogn. **45**(1), 208–230 (2012)
79. Kim, H., Kim, S., Kouritzin, M.A., Sun, W.: Detecting network portscans through anomaly detection. In: Proceedings of SPIE on Detecting Network Portscans Through Anomaly Detection, vol. 5429, pp. 254–263. SPIE, Orlando (2004)
80. Kohonen, T.: The self-organizing map. Proc. IEEE **78**(9), 1464–1480 (1990)
81. Kruegel, C., Mutz, D., Robertson, W., Valeur, F.: Bayesian event classification for intrusion detection. In: Proceedings of the 19th Annual Computer Security Applications Conference (2003)
82. Kuang, L.V.: DNIDS: a dependable network intrusion detection system using the CSI-KNN algorithm. Master's thesis, Queen's University Kingston, Ontario (2007)
83. Labib, K., Vemuri, R.: NSOM: A Tool to Detect Denial of Service Attacks Using Self-Organizing Maps. Tech. Rep., Department of Applied Science University of California, Davis (2002)

84. Leckie, C., Kotagiri, R.: A probabilistic approach to detecting network scans. In: Proceedings of the IEEE Network Operations and Management Symposium, pp. 359–372. IEEE Computer Society, Florence (2002)
85. Lee, W., Stolfo, S.J., Mok, K.W.: Adaptive intrusion detection: a data mining approach. Artif. Intell. Rev. **14**(6), 533–567 (2000)
86. Lee, W., Xiang, D.: Information-theoretic measures for anomaly detection. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 130–143. IEEE Computer Society, Washington, DC (2001)
87. Leung, K., Leckie, C.: Unsupervised anomaly detection in network intrusion detection using clusters. In: Proceedings of the 28th Australasian conference on Computer Science, vol. 38, pp. 333–342. Australian Computer Society, Inc., Darlinghurst (2005)
88. Li, Y., Luo, X., Qian, Y., Zhao, X.: Network-wide traffic anomaly detection and localization based on robust multivariate probabilistic calibration model. Math. Probl. Eng. **2015** (2015)
89. Liu, G., Yi, Z., Yang, S.: A hierarchical intrusion detection model based on the PCA neural networks. Neurocomputing **70**(7-9), 1561–1568 (2007)
90. Locasto, M.E., Wang, K., Keromytis, A.D., Stolfo, S.J.: FLIPS: hybrid adaptive intrusion prevention. In: Recent Advances in Intrusion Detection, pp. 82–101 (2005)
91. Lu, W., Ghorbani, A.A.: Network anomaly detection based on wavelet analysis. EURASIP J. Adv. Signal Process. **2009**(837601) (2009)
92. Lu, W., Tong, H.: Detecting network anomalies using CUSUM and EM clustering. In: Proceedings of the 4th International Symposium on Advances in Computation and Intelligence, pp. 297–308. Springer (2009). doi:http://dx.doi.org/10.1007/978-3-642-04843-2_32
93. Mabu, S., Chen, C., Lu, N., Shimada, K., Hirasawa, K.: An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming. IEEE Trans. Syst. Man Cybern. C Appl. Rev. **41**(1), 130–139 (2011)
94. Mafra, P.M., Moll, V., Fraga, J.D.S., Santin, A.O.: Octopus-IIDS: an anomaly-based intelligent intrusion detection system. In: Proceedings of the IEEE Symposium on Computers and Communications, pp. 405–410. IEEE CS (2010)
95. Mahoney, M.V., Chan, P.K.: PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic. Tech. Rep. cs-2001-04, Dept. of Computer Science, Florida Tech (2001)
96. Mahoney, M.V., Chan, P.K.: Learning rules for anomaly detection of hostile network traffic. In: Proceedings of the 3rd IEEE International Conference on Data Mining. IEEE CS, Washington (2003)
97. Manikopoulos, C., Papavassiliou, S.: Network intrusion and fault detection: a statistical anomaly approach. IEEE Commun. Mag. **40**(10), 76–82 (2002)
98. Markou, M., Singh, S.: Novelty detection: a review—part 1: statistical approaches. Signal Process. **83**(12), 2481–2497 (2003). doi:10.1016/j.sigpro.2003.07.018
99. Mateti, P.: Lecture Notes on Internet Security. Wright State University, Dayton, US (2010)
100. Mishra, B.K., Ansari, G.M.: Differential epidemic model of virus and worms in computer network. Int. J. Netw. Secur. **14**(3), 149–155 (2012)
101. Mohajerani, M., Moeini, A., Kianie, M.: NFIDS: a neuro-fuzzy intrusion detection system. In: Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems, vol. 1, pp. 348–351 (2003)
102. Muda, Z., Yassin, W., Sulaiman, M.N., Udzir, N.I.: A K-means and Naive-bayes learning approach for better intrusion detection. Inf. Technol. J. **10**(3), 648–655 (2011)
103. Naldurg, P., Sen, K., Thati, P.: A temporal logic based framework for intrusion detection. In: Proceedings of the 24th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems, pp. 359–376 (2004)
104. Neumann, B.: Knowledge management and assistance systems. http://kogs-www.informatik.uni-hamburg.de/ neumann/ (2007)
105. Nguyen, H.H., Harbi, N., Darmont, J.: An efficient local region and clustering-based ensemble system for intrusion detection. In: Proceedings of the 15th Symposium on International Database Engineering & Applications, pp. 185–191. ACM (2011)

106. Noel, S., Wijesekera, D., Youman, C.: Modern intrusion detection, data mining, and degrees of attack guilt. In: Proceedings of the International Conference on Applications of Data Mining in Computer Security. Springer (2002)
107. Noto, K., Brodley, C., Slonim, D.: Anomaly detection using an ensemble of feature models. In: Proceedings of the IEEE International Conference on Data Mining, pp. 953–958. IEEE CS (2010)
108. Otey, M.E., Ghoting, A., Parthasarathy, S.: Fast distributed outlier detection in mixed-attribute data sets. Data Min. Knowl. Disc. **12**(2-3), 203–228 (2006)
109. Panda, M., Abraham, A., Patra, M.R.: Hybrid intelligent systems for detecting network intrusions. Secur. Commun. Netw. **8**(16), 2741–2749 (2015). http://dx.doi.org/10.1002/sec.592
110. Parikh, D., Chen, T.: Data fusion and cost minimization for intrusion detection. IEEE Trans. Inf. Forensics Secur. **3**(3), 381–389 (2008)
111. Parlos, A., Chong, K., Atiya, A.: Application of the recurrent multilayer perceptron in modeling complex process dynamics. IEEE Trans. Neural Netw. **5**(2), 255–266 (1994)
112. Patcha, A., Park, J.M.: An overview of anomaly detection techniques: existing solutions and latest technological trends. Comput. Netw. **51**(12), 3448–3470 (2007)
113. Pawlak, Z.: Rough sets. Int. J. Parallel Prog. **11**(5), 341–356 (1982)
114. Paxson, V.: Bro: a system for detecting network intruders in real-time. In: Proceedings of the the 7th USENIX Security Symposium, pp. 2435–2463. Usenix Association, San Antonio (1998)
115. Peddabachigari, S., Abraham, A., Grosan, C., Thomas, J.: Modeling intrusion detection system using hybrid intelligent systems. J. Netw. Comput. Appl. **30**(1), 114–132 (2007)
116. Perdisci, R., Ariu, D., Fogla, P., Giacinto, G., Lee, W.: McPAD: a multiple classifier system for accurate payload-based anomaly detection. Comput. Netw. **53**(6), 864–881 (2009)
117. Perdisci, R., Gu, G., Lee, W.: Using an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems. In: Proceedings of the 6th International Conference on Data Mining, pp. 488–498. IEEE CS (2006)
118. Polikar, R.: Ensemble based systems in decision making. IEEE Circuits Syst. Mag. **6**(3), 21–45 (2006)
119. Portnoy, L., Eskin, E., Stolfo, S.: Intrusion detection with unlabeled data using clustering. In: Proceedings of the ACM CSS Workshop on on Data Mining Applied to Security, pp. 5–8. Philadelphia (2001)
120. Prayote, A.: Knowledge based anomaly detection. Ph.D. thesis, School of Computer Science and Egineering, The University of New South Wales, Australia (2007)
121. Prayote, A., Compton, P.: Detecting Anomalies and Intruders. Adv. Artif. Intell. AI 2006 1084–1088 (2006)
122. Qadeer, M.A., Iqbal, A., Zahid, M., Siddiqui, M.R.: Network traffic analysis and intrusion detection using packet sniffer. In: Proceedings of the 2nd International Conference on Communication Software and Networks, pp. 313–317. IEEE Computer Society, Washington, DC (2010)
123. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
124. Rehak, M., Pechoucek, M., Celeda, P., Novotny, J., Minarik, P.: CAMNEP: agent-based network intrusion detection system. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track, pp. 133–136. IFAAMS, Richland (2008)
125. Roesch, M.: Snort – lightweight intrusion detection for networks. In: Proceedings of the 13th USENIX Conference on System Administration, pp. 229–238. Usenix Association, Seattle (1999)
126. Rokach, L.: Ensemble-based classifiers. Artif. Intell. Rev. **33**(1–2), 1–39 (2010)
127. Romig, S.: The OSU flow-tools package and CISCO NetFlow logs. In: Proceedings of the 14th USENIX conference on System Administration, pp. 291–304. USENIX Association, Berklay (2000)

128. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**(1), 53–65 (1987)
129. SchAolkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. **13**(7), 1443–1471 (2001)
130. Schapire, R.E.: A brief introduction to boosting. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence, pp. 1401–1406. Morgan Kaufmann (1999)
131. Scheirer, W., Chuah, M.C.: Syntax vs. semantics: competing approaches to dynamic network intrusion detection. Int. J. Secur. Netw. **3**(1), 24–35 (2008)
132. Schneier, B.: Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd edn. Wiley, New York (1995)
133. Sekar, R., Gupta, A., Frullo, J., Shanbhag, T., Tiwari, A., Yang, H., et al.: Specification-based anomaly detection: a new approach for detecting network intrusions. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 265–274 (2002)
134. Selim, S., Hashem, M., Nazmy, T.M.: Hybrid multi-level intrusion detection system. Int. J. Comput. Sci. Inf. Secur. **9**(5), 23–29 (2011)
135. Sequeira, K., Zaki, M.: ADMIT: anomaly-based data mining for intrusions. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 386–395. ACM, New York (2002)
136. Shabtai, A., Kanonov, U., Elovici, Y.: Intrusion detection for mobile devices using the knowledge-based, temporal abstraction method. J. Syst. Softw. **83**(8), 1524–1537 (2010)
137. Shifflet, J.: A technique independent fusion model for network intrusion detection. In: Proceedings of the Midstates Conference on Undergraduate Research in Computer Science and Mathematics, vol. 3, pp. 13–19 (2005)
138. Song, S., Ling, L., Manikopoulo, C.: Flow-based statistical aggregation schemes for network anomaly detection. In: Proceedings of the IEEE International Conference on Networking, Sensing and Control, pp. 786–791. IEEE, Ft. Lauderdale (2006)
139. Song, X., Wu, M., Jermaine, C., Ranka, S.: Conditional anomaly detection. IEEE Trans. Knowl. Data Eng. **19**(5), 631–645 (2007)
140. Sridharan, A., Ye, T., Bhattacharyya, S.: Connectionless port scan detection on the backbone. In: Proceedings of the 25th IEEE International Conference on Performance, Computing, and Communications, pp. 567–576. IEEE Computer Society, Phoenix (2006)
141. Staniford-Chen, S., Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., Levitt, K., Wee, C., Yip, R., Zerkle, D.: GrIDS: a graph based intrusion detection system for large networks. In: Proceedings of the 19th National Information Systems Security Conference, pp. 361–370. NIST, CSRC, Baltimore (1996)
142. Su, M.Y., Yu, G.J., Lin, C.Y.: A real-time network intrusion detection system for large-scale attacks based on an incremental mining approach. Comput. Secur. **28**(5), 301–309 (2009)
143. Subramoniam, N., Pawar, P.S., Bhatnagar, M., Khedekar, N.S., Guntupalli, S., Satyanarayana, N., Vijayakumar, V.A., Ampatt, P.K., Ranjan, R., Pandit, P.S.: Development of a comprehensive intrusion detection system – challenges and approaches. In: Proceedings of the 1st International Conference on Information Systems Security, pp. 332–335. Kolkata (2005)
144. Sun, J., Yang, H., Tian, J., Wu, F.: Intrusion detection method based on wavelet neural network. In: Proceedings of the 2nd International Workshop on Knowledge Discovery and Data Mining, pp. 851–854. IEEE CS (2009)
145. Tajbakhsh, A., Rahmati, M., Mirzaei, A.: Intrusion detection using fuzzy association rules. Appl. Soft Comput. **9**(2), 462–469 (2009)
146. Tong, H., Li, C., He, J., Chen, J., Tran, Q.A., Duan, H.X., Li, X.: Anomaly internet network traffic detection by kernel principle component classifier. In: Proceedings of the 2nd International Symposium on Neural Networks, LNCS, vol. 3498, pp. 476–481 (2005)
147. Tong, X., Wang, Z., Yu, H.: A research using hybrid RBF/Elman neural networks for intrusion detection system secure model. Comput. Phys. Commun. **180**(10), 1795–1801 (2009)
148. Treurniet, J.: A network activity classification schema and its application to scan detection. IEEE/ACM Trans. Netw. **19**(5), 1396–1404 (2011)

149. Tsai, C.F., Hsu, Y.F., Lin, C.Y., Lin, W.Y.: Intrusion detection by machine learning: a review. Expert Syst. Appl. **36**(10), 11,994–12,000 (2009)
150. Udhayan, J., Prabu, M.M., Krishnan, V.A., Anitha, R.: Reconnaissance scan detection heuristics to disrupt the pre-attack information gathering. In: Proceedings of the International Conference on Network and Service Security, pp. 1–5. IEEE Computer Society, ESIEA-9, 75005 Paris (2009)
151. Visconti, A., Tahayori, H.: Artificial immune system based on interval type-2 fuzzy set paradigm. Appl. Soft Comput. **11**(6), 4055–4063 (2011)
152. Wagner, C., François, J., State, R., Engel, T.: Machine learning approach for IP-flow record anomaly detection. In: Proceedings of the 10th International IFIP TC 6 conference on Networking – Volume Part I, pp. 28–39 (2011)
153. Wang, K., Stolfo, S.J.: Anomalous payload-based network intrusion detection. In: Proceedings of the Recent Advances in Intrusion Detection, pp. 203–222. Springer (2004)
154. Wattenberg, F.S., Perez, J.I.A., Higuera, P.C., Fernandez, M.M., Dimitriadis, I.A.: Anomaly detection in network traffic based on statistical inference and α-stable modeling. IEEE Trans. Dependable Secure Comput. **8**(4), 494–509 (2011)
155. Xian, J.Q., Lang, F.H., Tang, X.L.: A novel intrusion detection method based on clonal selection clustering algorithm. In: Proceedings of the International Conference on Machine Learning and Cybernetics, vol. 6. IEEE Press (2005)
156. Xie, X.L., Beni, G.: A validity measure for fuzzy clustering. IEEE Trans. Pattern Anal. Mach. Intell. **13**(4), 841–847 (1991)
157. Xu, X.: Sequential anomaly detection based on temporal difference learning: principles, models and case studies. Appl. Soft Comput. **10**(3), 859–867 (2010)
158. Yan, R., Shao, C.: Hierarchical method for anomaly detection and attack identification in high-speed network. Inf. Technol. J. **11**(9), 1243–1250 (2012)
159. Yong, H., Feng, Z.X.: Expert system based intrusion detection system. In: Proceedings of the International Conference on Information Management, Innovation Management and Industrial Engineering, vol. 4, pp. 404–407 (2010)
160. Yu, H., Kim, S.: SVM tutorial – classification, regression and ranking. In: Handbook of Natural Computing. Springer, Berlin/Heidelberg (2003)
161. Yu, M.: A Nonparametric adaptive CUSUM method and its application in network anomaly detection. Int. J. Adv. Comput. Technol. **4**(1), 280–288 (2012)
162. Yu, X.: A new model of intelligent hybrid network intrusion detection system. In: Proceedings of the International Conference on Bioinformatics and Biomedical Technology, pp. 386–389. IEEE CS (2010)
163. Zhang, C., Zhang, G., Sun, S.: A mixed unsupervised clustering-based intrusion detection model. In: Proceedings of the 3rd International Conference on Genetic and Evolutionary Computing, pp. 426–428. IEEE CS (2009)
164. Zhang, H.L.: Agent-based open connectivity for decision support systems. Ph.D. thesis, School of Computer Science and Mathematics, Victoria University (2007)
165. Zhang, J., Zulkernine, M.: A hybrid network intrusion detection technique using random forests. In: Proceedings of the 1st International Conference on Availability, Reliability and Security, pp. 262–269. IEEE CS (2006). doi:10.1109/ARES.2006.7
166. Zhang, J., Zulkernine, M., Haque, A.: Random-forests-based network intrusion detection systems. IEEE Trans. Syst. Man Cybern. C **38**(5), 649–659 (2008)
167. Zhang, Y., Fang, B.: A novel approach to scan detection on the backbone. In: Proceedings of the Sixth International Conference on Information Technology: New Generations, pp. 16–21. IEEE Computer Society, Washington, DC (2009)
168. Zhang, Y.F., Xiong, Z.Y., Wang, X.Q.: Distributed intrusion detection based on clustering. In: Proceedings of the International Conference on Machine Learning and Cybernetics, vol. 4, pp. 2379–2383 (2005)

169. Zhang, Z., Li, J., Manikopoulos, C.N., Jorgenson, J., Ucles, J.: HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In: Proceedings of IEEE Man Systems and Cybernetics Information Assurance Workshop (2001)
170. Zhi-dong, L., Wu, Y., Wei, W., Da-peng, M.: Decision-level fusion model of multi-source intrusion detection alerts. J. Commun. **32**(5), 121–128 (2011)
171. Zhuang, Z., Li, Y., Chen, Z.: Enhancing intrusion detection system with proximity information. Int. J. Secur. Netw. **5**(4), 207–219 (2010)

Chapter 5
Alert Management and Anomaly Prevention Techniques

As an ANIDS (anomaly-based network intrusion detection system) or IDS (intrusion detection system) monitors network-wide traffic, it generates warning messages (i.e., alerts) that indicate attack or suspicious or legitimate events. Due to widespread deployment of IDSs, they may generate an overwhelming number of alerts with true alerts mixed with false alerts. So, management of such alerts is indeed necessary to get to the origin of an attack, so that survival measures may be taken at the earliest. This chapter focuses on alert management and network anomaly prevention techniques. Alert management contains several components, viz., alert clustering, alert merging, alert frequency, alert link, alert association, intention recognition, and alert correlation. However, network traffic anomaly prevention techniques include basic concepts of ANIPS (anomaly-based network intrusion prevention system), attack coverage, features of ANIPS, and selection of the right ANIPS for deployment. Finally, the chapter presents the pros and cons of both alert management and anomaly-based network intrusion prevention techniques.

5.1 Alert Management

Since an intrusion detection system (IDS) is deployed in an alive network, it is likely to generate alerts for mostly malicious events. But it may also generate alerts for events that are incorrectly detected as malicious. So, proper diagnosis of alerts is a major issue in network-wide traffic intrusion detection. An architecture for alert generation and management is given in Fig. 5.1 [31].

This architecture has four different levels, viz., improving IDSs, deployment environment, alert post-processing, and involvement of an analyst.

1. *Level 1—Improving IDSs*: In this step, most IDSs focus on how a detection engine can efficiently detect network anomalies with a low false-positive rate. In

M.H. Bhuyan et al., *Network Traffic Anomaly Detection and Prevention*,
Computer Communications and Networks, DOI 10.1007/978-3-319-65188-0_5

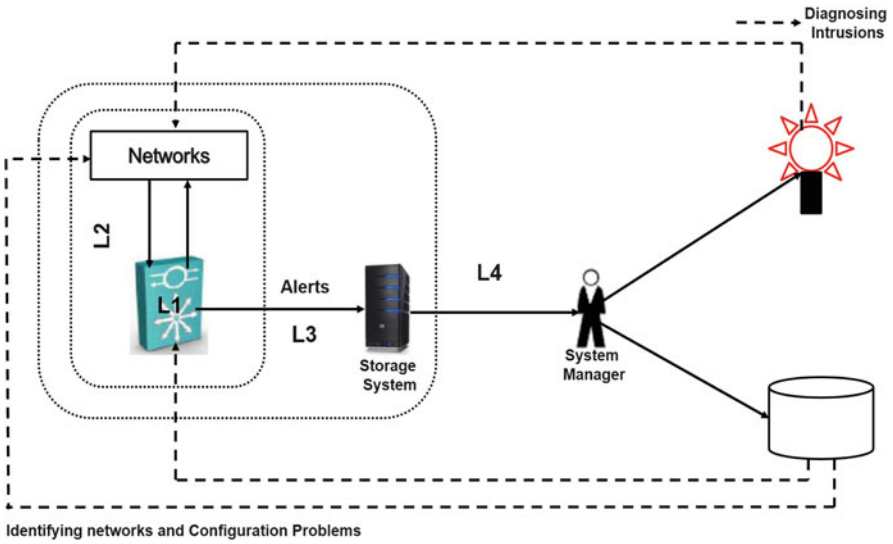


Fig. 5.1 An architecture for alert generation and management

anomaly-based network intrusion detection, the detection engine uses a simple pattern matching mechanism whose parameters have been optimized.

2. *Level 2—Deployment environment*: ANIDSs have limited detection ability, which may also depend on where exactly these are deployed. At the outset, an ANIDS may not be able to differentiate well between legitimate traffic and anomalous traffic with certainty. With the passage of time, an ANIDS may gain knowledge of the deployment environment and reduce the rate of false alarms are to better understanding of the environment.
3. *Level 3—Alert post-processing*: Alert post-processing examines alerts generated from the IDS and attempts to improve the function of the IDS. It evaluates the quality of the alerts based on different parameters that affect the normal activity of a network. It can be achieved using data mining and other techniques that correlate alerts at different levels, thus creating what is called an alert correlation system.
4. *Level 4—System manager*: A system security manager analyzes the alerts in real time or within a short-time interval. But system manager involvement is rare if the detection mechanisms are fast.

Different IDSs have been developed and deployed in different environments to provide robust defense of networks. An IDS is useful because it generates alert based on the abnormal activity in the network. In practice, in an organization's security operation center, the security manager audits the alerts typically 24 h a day, 7 days a week, so that the security manager can respond to intrusion at the earliest. The security manager uses a knowledge base to distinguish between false and true positives among intrusion alerts [46]. To build this knowledge base, it needs

to collect pre-classified examples of both categories of alerts (i.e., legitimate and malicious). But obtaining pure pre-classified alerts is difficult. A security manager reports security incidents and investigates intrusions or configuration problems based on the alert classes it knows about. It also may try to modify bad alert signatures or apply filters to remove alerts based on pre-defined criteria. Alert management has three major functioning components, viz., (i) alert correlation, (ii) alert merging, and (iii) alert clustering [36]. Each component is described below. Before an in-depth discussion of each component of alert management, we provide a few definitions for better understandability of contents that follow.

Definition 5.1 ***Alert***: An alert is a warning message generated by an IDS. It may indicate an attack or suspicious event.

Definition 5.2 ***Event***: An alert is an action directed at a target which is intended to result in a change of state of the target. An event may generate no alerts, a single alert, or many alerts.

Definition 5.3 ***Synthetic alert***: An alert that can be produced by the detection engine itself, when an attack signature is fired.

Definition 5.4 ***Alert feature***: A specific field of information contained within an alert representing an item of interest. Some examples of normal alert features are time, alert name, source, and destination.

Definition 5.5 ***Alert group***: A set of alerts which may be matched to an attack signature in any order.

Definition 5.6 ***Alert propagation***: The act of generating repeated alerts that match a signature which is not complete. It may be used to provide better overall attack detection.

Definition 5.7 ***Alert sequence***: An alert sequence is a set of alerts whose elements must be matched to an attack signature in a specific temporal order.

Definition 5.8 ***Alert stream***: The flow of alerts of all IDS as well as alerts that may be generated by the system itself, when a signature is fired. An alert stream consists of the alerts gathered from the database in an offline context.

Definition 5.9 ***Component alert***: An alert that matches a portion of a signature and is associated with a specific instance of a signature, possibly including IDS-generated alerts.

Definition 5.10 ***Alert cluster***: An alert cluster is a group of alerts that are related by one or more common features, such as time, source, or destination.

Definition 5.11 ***Alert correlation***: The discovery and/or establishment of either mutual or causal relationships between alerts, usually in order to logically group alerts based on relationship to attacks or actions on a system or network.

Definition 5.12 ***IDS sensor***: Sensors are the basic detection components of an IDS. Their basic role is to provide either alerts or intrusion-related data directly to a

higher-level analysis component within the same IDS. The distinction between a full IDS and sensors is normally defined in terms of information flow. Any component providing information to an analysis system can be termed a sensor for that analysis system. In a normal IDS analysis system, any IDS providing information can be classed a sensor.

5.1.1 Alert Correlation

Alert correlation is an analysis process for alerts obtained from different IDS sensors to provide a compact report on the security status of the network under surveillance. Correlation process provides a summary of security activities on the network where IDSs have been deployed with sensors. It consists several components that transform intrusion detection sensor alerts into intrusion reports. Because alerts may be caused by different types of attacks, such as single level and multilevel with different depths, the correlation process cannot treat all alerts equally with equal importance. Therefore, it is really necessary to provide different components to concentrate on different aspects of the overall correlation task.

There are several alert correlation mechanisms available in the literature [4, 46, 49]. Each mechanism contains a different set of components, a few of which are given in Table 5.1.

As the components of alert correlation vary for different approaches, we attempt to present a coherent of overall picture. We classify the components as alert normalization, preprocessing, correlation techniques, post-processing, and validation. In the next few sections, we discuss each stage.

Table 5.1 Alert correlation process

Valeur et al. [46]	Siraj [39]	Elshoush and Osman [15]	Ning et al. [29]	Maggi and Zanero [26]	Huballi and Suryanarayanan [20]
Normalization, Preprocessing, Alert Fusion, Alert Verification, Thread Reconstruction, Attack Session Reconstruction, Focus Recognition, Multi-step Correlation, Impact Analysis, Prioritization	Normaliz-ation/ Formatting, Reduction Severity/ Prioritization, Attack Scenario Contribution, Attack Prediction	Normalization, Preprocessing, Prioritization, Alert Verification, Alert Fusion	Adjustable Graph Reduction, Focused Analysis, Graph Decompo sition	Normalization, Prioritization, Aggregation, Correlation, Verification	Alert Normalization, Alert Clustering, Alert Correlation, Intention Recognition

5.1.1.1 Alert Normalization

IDS sensors generate alerts and encode them in different formats. These alerts are usually received by the correlation process from different sensors. The primary objective of alert normalization is to translate features of each sensor alert into a generic format to reduce the number of alerts to be correlated. The Internet Engineering Task Force (IETF) has proposed a generic representation of intrusion alerts to develop a standard known as Intrusion Detection Message Exchange Format (IDMEF) [11]. As defined at [11], IDMEF "defines data formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to the management systems that may need to interact with them" [11]. A simplified version of IDMEF is given in Fig. 5.2.

This IDMEF format defines the semantics of only a few attributes. The format is mostly concerned with the syntactic rules. Network defenders usually choose different names for the same attack to provide additional fields in the reports they generate. As a result, the same alerts may be filed differently.

An IDMEF message is classified into heartbeat and alert. A *heartbeat* message is sent by an analyzer in a periodic fashion indicating that it is up and running. An *alert* message is composed of nine different components:

- *CreateTime*: The time when the alert was generated.
- *DetectTime*: The time when the event(s) leading up to the alert was (were) detected. This could be different from *CreateTime* in specific situations.
- *AnalyzerTime*: Current time on the analyzer.
- *Analyzer*: Identification information for the analyzer that generated the alert.
- *Source*: The source that triggered the alert. It is also composed of four aggregate classes: node, user, process, and service.
- *Target*: The main target of the alert. It has the same aggregate classes as the source has with on an additional class named *FileList*.
- *Classification*: Information that describes the alert.
- *Assessment*: Impact, action and response against the generated alerts with proper evaluation.
- *Additional data*: Additional information that does not fit into the data model.

MessageId is used to uniquely identify itself. There are three subclasses of the alert class. They are ToolAlert, OverflowAlert, and CorrelationAlert. Each subclass is dedicated for a specific purpose. The *ToolAlert* class signifies the attack tool used by the attackers. The *OverflowAlert* class stores the information regarding buffer overflow attacks. Finally, *CorrelationAlert* class merges the alerts together to form groups. IDMEF messages represented in the XML DTD [11, 21] and an IDMEF alert are as follows:

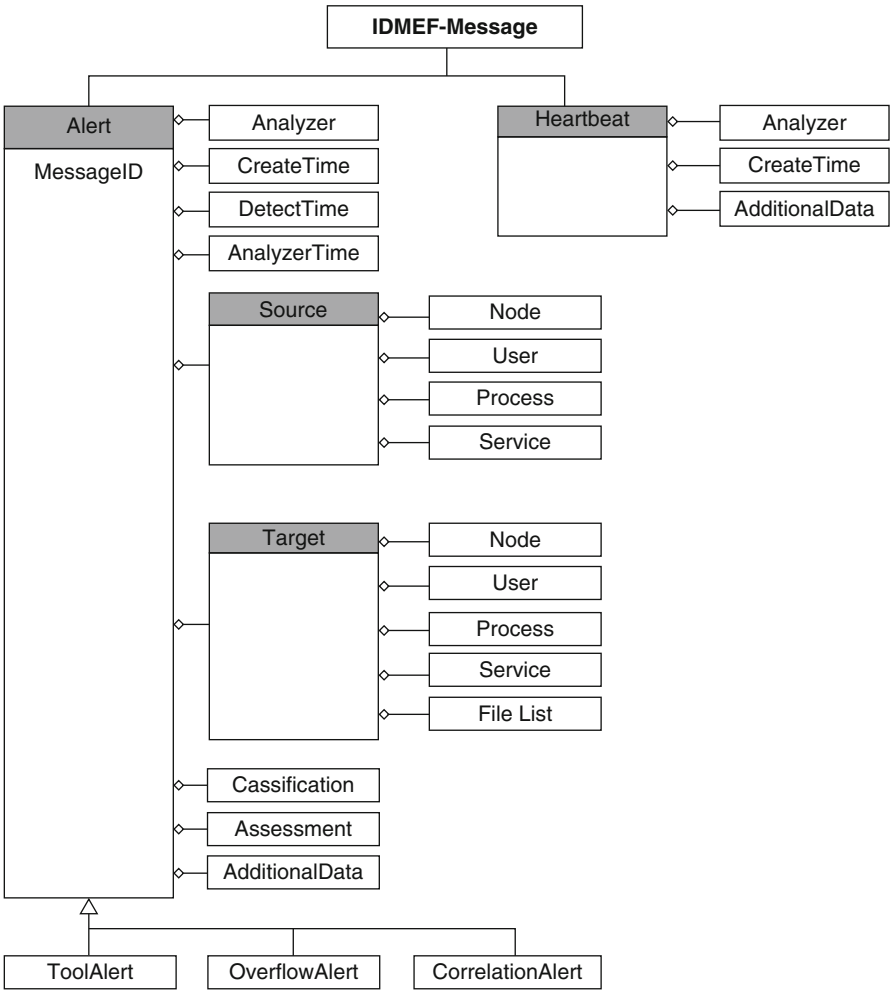


Fig. 5.2 A simple version of IDMEF message format [11]

1. IDMEF message in XML DTD format

```
<!ENTITY % attlist.idmef                    ''
    version              CDATA              #FIXED    '1.0'
  ''>
<!ELEMENT IDMEF-Message                     (
    (Alert | Heartbeat)*
  )>
<!ATTLIST IDMEF-Message
    %attlist.idmef;
  >
```

2. IDMEF alert format

```
<!ELEMENT Alert                          (
    Analyzer, CreateTime, DetectTime?, AnalyzerTime?,
 Source*, Target*, Classification,  Assessment?, (ToolAlert
    | OverflowAlert | CorrelationAlert)?, AdditionalData*
  )>
<!ATTLIST Alert
    messageid           CDATA                   '0'
  >
```

There are several techniques to normalize the alerts. A sample pseudo-code for normalization process [46] is given below.

```
global normal_db, alertname_db
normalize(raw_alert)
  alert ← new alert
  alert.alertid ← get_unique_id()
  alert.name ← get alertname from alertname_db using
  (raw_alert.name, raw_alert.sensortype) as key
  mappings ← get all m: mapping from normalization_db
  where m.sensor = raw_alert.sensor
  for each m: mapping in mappings:
      alert_attri ← m.alert_attri
      raw_attri ← m.raw_attri
      alert.alert_attri ← raw_alert.raw_attri
      pass alert to next correlation component
```

Normally each raw alert is translated into a standardized alert format. Its name and attributes are copied to the appropriate fields of the alert as defined by the attribute mappings in the normalization database. The primary attributes considered for alert normalization are given in Table 5.2.

If different types of sensor are used in an IDS for alert generation, a single sensor format may not have been used. So, normalization is required, but in this case, there is no need to have a normalization component. Most datasets are composed of data from several sensors. For generalization, it may be necessary to normalize the alerts so that the number of alerts is the same as the number of input alerts.

5.1.1.2 Preprocessing

Alert preprocessing is applied to the raw alerts to convert them into a generic format. It also reduces the number of alerts to be correlated. False alerts also need to be handled at an early stage. This reduces negative impact on the correlation results.

Table 5.2 Alert attributes with description

Alert attribute	Description
Alertid	A unique ID identifying the alert
Analyzertime	The time when the IDS sent alert
Attackernodes	The set of nodes where the attack originated
Attackgraph	A graph showing the progress of complex attacks
Consequence	A set of systems that are affected by this attack
Createtime	The time when the IDS generated the alert
Detecttime	The time when the IDS detected the attack
End_time	The time when the attack ended
Name	The name of the attack
Priority	A value indicating how important the attack is
Receivedtime	The time the alert was received by the correlator
Reference	A set of references to other alerts
Sensornode	The node at which the IDS that generated the alert runs
Start_time	The time when the attack started
Type	The attack type (DoS, Reconnaissance, Breakin)
Verified	If the attack was successful (true, false, unknown)
Victimnodes	The set of nodes that were victims of the attack
Victimprocess	The full path of the process that was attacked
Victimservice	Port number and protocol of the service that was attacked

Several components may initiate an attack, but in alert processing four major components are used to describe the attack source. They are the node, the user, the process, and the network service that originated the attack. The attacker usually includes infected files attached to any of the components on the target. During the correlation process, it requires at least one nonempty field for processing. In host-based alerts, the node field provides the address of the attack source and target, where the sensor is located. In network-based alerts, the node information provides source and destination IP addresses. However, an alert can be amplified with additional information based on a standardized alert name, which is assigned by the normalization component. A sample pseudo-code [46] for preprocessing of alert based on the sensor data is given in Fig. 5.3.

Based on the information provided by the preprocessing module, Table 5.3 shows the alerts for the example attacks.

5.1.1.3 Correlation Techniques

Alerts are mostly generated by IDSs independently, but they may have some logical connections with each other. Alerts may exhibit an attack that involves multiple stages in compromising a large-scale network. The correlation techniques try to reconstruct the attack scenarios from alerts obtained from the IDSs. Alert correlation techniques can be classified as (a) similarity based, (b) statistical based, (c) knowledge based, and (d) hybrid.

```
global attack_type_db

preprocess(alert) {
 if alert.start_time is null:
  if alert.detecttime is not null:
   alert.start_time ← alert.detecttime
  else if alert.createtime is not null:
   alert.start_time ← alert.createtime
  else if alert.analyzertime is not null:
   alert.start_time ← alert.analyzertime
  else:
   alert.start_time ← alert.receivedtime

 if alert.end_time is null:
  alert.end_time ← alert.start_time

 if alert is produced by a host-based IDS:
  alert.victimnodes ← alert.sensornode
  alert.attackernodes ← alert.sensornode

 alert.type ←
  get alertype from attack_type_db using
   alert.name as key
 pass alert to next component
}
```

Fig. 5.3 A sample pseudo-code for preprocessing of alerts [46]

Table 5.3 Example attacks: effect of preprocessing of an alert

Alert ID	Name	Sensor	StartEnd	Source	Target	Tag
8	Local exploit	H	22.222.2	10.0.0.1	10.0.0.1, Apache	
9	Bad request	A	28.528.5	10.0.0.1	10.0.0.1, Apache	
9	Local exploit	A	22.222.2	10.0.0.1	10.0.0.1, linuxconf	

(a) *Similarity-based techniques*: Such technique works using the principle of similarity, where it selects a relevant feature set, such as source IP, destination IP, and port numbers for correlation. Alerts with a high degree of similarity are correlated.

An example of similarity-based alert correlation technique is reported by Valdes and Skinner [45]. They consider a few features from the standard

template proposed by IETF [11]. They define a similarity function to measure the feature correlation level between features of the alerts. A new alert is compared with the existing alerts based on the similarity value of each feature to decide either to correlate or to create a real new alert. The similarity score or value is computed based on a user-defined threshold. If dissimilar, the alert is added to the meta-alert list. The meta-alert list is composed of alerts from heterogeneous sensors.

Measuring feature similarity among alerts is the primary concern in this approach. They consider four different metrics for measuring the similarity between two alerts. They are feature overlap, feature similarity, expectation of similarity, and minimum similarity. Each metric is briefly described below.

- *Feature overlap*: Each alert has some features. A new alert and existing alerts may share some common features such as source address, destination address, flag, time information, and types of attack. Only common features are used in measuring the similarity of two alerts.
- *Feature similarity*: The value of similarity scores based on common feature of alerts depends on the type of information. For example, similarity of two source addresses can be estimated based on the higher bits of IP addresses, while similarity between attack classes is computed based on an incident class similarity matrix.
- *Expectation of similarity*: This measure is used to normalize overall similarity. Two features will match if they are related. For example, in a probing attack, an attacker is trying to connect different hosts, and therefore, the contribution of similarity of destination host is low.
- *Minimum similarity*: The minimum similarity is the degree of similarity that must be met by certain features of each alert. It is a minor requirement and is not a sufficient criterion for alert correlation. If the degree of similarity is lower than the minimum similarity, the alert will not be correlated regardless the value of the overall similarity.

The overall similarity between two alerts can be computed using the following formula.

$$Sim(X^i, Y) = \frac{\sum_j E_j Sim(X_j^i, Y_j)}{\sum_j E_j} \tag{5.1}$$

where, X^i is the candidate meta alert i for matching; Y is the new alert; j is the index over the alert features; E_j is the expectation of similarity for feature j; and X_j and Y_j are the values for feature j in alerts X and Y, respectively. The main aim is to find an i such that $Sim(X^i, Y)$ is maximum. If $Sim(X^i, Y)$ is greater than or equal to minimum similarity, X^i and Y will be correlated; otherwise, Y will become a new meta-alert. There are other techniques [8, 22], which are based on the mechanism of similarity.

The main advantages of similarity-based correlation techniques are:

- It is good to discover simple attacks with a small number of features.
- They work well for a known set of alerts with a known feature set.

The shortcomings of similarity-based correlation techniques are:

- They are suitable for known alerts only.
- Similarity-based techniques are not able to discover causality among alerts and other statistical relationships.
- It is limited in discovering complicated attacks.

(a) *Statistical-based techniques*: Statistical algorithms are dependent on the attributes for which statistical properties can be computed. Similar attacks have similar statistical attributes, and so, they can be categorized easily. To discover attack steps, these techniques store causal relationships between incidents and analyze them with frequencies of occurrence during a certain interval of time [3]. Results may be correlated to different attack stages. Statistical algorithms do not have any prior knowledge of the attack scenarios. Combining data of different sensors becomes impossible if the sensor is incomplete or abnormal [28]. Statistical computation can be of three different categories, viz., (i) detection of repeated and repetition patterns; (ii) estimation of causal relationships between alerts, predicting next alert occurrence, and detecting attacks; and (iii) combining reliability by mixing completely similar alerts. Maggi and Zanero [25] use different statistical tests for alert correlation. They use Granger causality test [42] for alert correlation and find better results without configuring complex parameters. The advantages of statistical techniques are:

- Without knowledge of attack scenarios, a statistical technique can identify the correlation between alerts based on the statistical attributes.

A few disadvantages are given below:

- Not able to discover dependencies
- Difficult to estimate correlation parameters
- Not able to discover structural cause relationships between alerts

(a) *Knowledge-based techniques*: Such technique is designed based on a knowledge base of attack definitions. The required knowledge can be classified as [28] (i) pre-requisites and consequences and (ii) attack scenarios. With the use of prerequisites and consequences, each incident is connected to each other by a network of conjunctions and disjunctions. Network attack scenarios are likely to be included in such a description. At a higher level, such a technique correlates alerts based on the feature similarities. It combines alerts based on predefined attack patterns at a lower level. In scenario-based algorithms, previous knowledge is necessary for determining prerequisites and incident results. Each attack must follow one by one the necessary steps to achieve success. Low-level alerts can be compared with predefined intrusion steps and

a sequence of alerts correlated to each attack. So, definitions of different attack scenarios must exist in a knowledge base. Once a new alert is created, it is compared to the available scenarios, and those whose probability of match is more than predefined threshold are attached to a scenario. Otherwise, if the alert is compatible with one of the possible scenario definitions inside the knowledge base, a new current scenario is generated using this alert. These techniques correlate based on known attack scenarios, where an attack scenario is specified either by an attack language such as STATL [14] or LAMBDA [10] or learned from the training datasets using a data mining approach [12].

Eckmann et al. [14] present a state transition-based attack language known as STATL. An analyst uses it to describe a sequence of actions that an attacker can perform to compromise the system. STATL has its own syntax and semantics that include lexical elements, data types, scenarios, front matter, states, transitions, EventSpecs, NamedSigActions, code blocks, timers, assertions, and annotations. States and transitions are the two main components. A state represents the status of the attack in a particular stage in case of multistage attack scenarios. Each state has a name and optional elements such as annotations, assertion, and a code block. A transition is a connection between two states. A transition may have annotations and a code block and must specify a transition type and an event type to match. The event type of a transition can be specified either directly by the EventSpecs, which define what events to match and under what conditions, or by the NamedSigAction, which is a reference to a named EventSpec. STATL uses its syntax and semantics to specify the attack scenarios and to correlate alerts that match these predefined scenarios. An example of ftp-write attack scenario can be described using STATL specification as given in Fig. 5.4 [14].

The objective of using the STATL language for alert correlation is to find a sequence of alerts that match the predefined scenarios. Based on the precondition and post-condition relationship between two alerts, the language statements are used to construct attack scenarios [7, 10, 14]. However, there are additional mechanisms such as the ones introduced in [1, 27] that a reader may want to explore further.

The main advantage of these techniques is:

– A knowledge base is usually built based on known attack scenarios.

The main disadvantages of these techniques are:

– One needs to manually define prerequisites.
– It is not able to deal with new patterns of alerts.
– It is difficult to update the correlation knowledge.
– It is not able to discover structure and statistical relationships
– It is impractical for use in large scale or real time due high computational expense.

(a) *Hybrid techniques*: These techniques combine features of all other techniques for alert correlation. Wang et al. [47] present an alert correlation method that

```
use ustat; scenario ftp_write {
    int user;
    int pid;
    int inode;

    initial state s0 { }

    transition create_file (s0 -> s1)
        nonconsuming
    {
        [WRITE w] : (w.euid != 0) && (w.owner != w.ruid)
        { inode = w.inode; }
    }

    state s1 { }
    transition login (s1 -> s2)
        nonconsuming
    { [EXEC e] :
        match_name(e.objname, "login")
        {
            user = e.ruid;
            pid = e.pid;
        }
    }

    state s2 { }
    transition read_rhosts (s2-> s3)
        consuming
        {
            [READ r] : (r.pid == pid) && (r.inode == inode)
        }

    state s3
    {
        {
            string username;
            userid2name(user, username);
            log("remote user %s \gained local access", username);
        }
    }
}
```

Fig. 5.4 An example of ftp-write attack scenario described using STATL specification [46]

hypothesizes the missed alerts and finds them. The approach uses three different types of information, viz., vulnerabilities, their dependencies, and network connectivity. First, it creates an attack graph *AG* based on the information considered. The authors suggest the building of a queue graph *QG* also. The method correlates alerts based on the attack graph *AG*. This method demonstrates the efficiency of the correlation process.

Each security condition is considered a variable and each exploit is stored in a queue of length one. Each alert generated by the NIDS is first compared with an exploit and placed in the concern queue. Correlations are collected as a directed graph known as the result graph. A breadth-first search is performed in the attack graph based on the directed edges because each edge represents

one exploit. For each edge, a forward pointer is created to connect the concern queue and the variable. This process is performed for each search. Later the backward edges are used for correlation purposes and forward edges were used for prediction purposes.

The missing alerts cause inconsistency between the knowledge encoded and AGs and facts represented by received alerts. The missing alerts can be plausibly hypothesized. A queue is kept in the memory for matching recent known exploits. Both new and in memory alerts are explicitly recorded. This method filters those alerts that do not match existing vulnerabilities. It hypothesized that these alerts are related to unknown and new vulnerabilities. If these fall into the wrong hands, they would be very valuable tools for an attacker. There are several hybrid alert correlation mechanisms available [3, 16, 35] that reflect in different levels of performance.

The advantages of hybrid alert correlation techniques are the following:

- They use the best features of both knowledge-based and similarity-based alert correlation techniques.
- Such techniques utilize the most common features of each alert in performing correlations.

The disadvantages of hybrid alert correlation techniques include the following:

- The use of such alerts may lead to complex architectures.
- Sometimes such a technique increases complexity due to use of a large number of alert features.

5.1.1.4 Post-Processing

A NIDS produces a large number of alerts each day. Several are generated based on failed attacks and false alarms triggered by legitimate traffic. So, it is indeed necessary to perform post-processing of intrusion alerts. Without post-processing, it may really be a difficult task to effectively recognize a particular attack. There are certain procedures to perform alert post-processing such as *alert prioritization* and *intention recognition*.

(a) *Alert prioritization*: Alert prioritization categorizes and assigns priorities to alerts based on importance. It is efficient. Porras et al. [32] present an alert correlation and incident ranking mechanism known as M-correlator. To better represent intrusion alerts for incident ranking, M-correlator creates an *incident handling fact base* that matches the alert content against mission specifications and dependencies of an incident with the configuration of the target host. The incident handling fact base defines entries including incident codes, COST codes, descriptions, incident classes, bound ports, vulnerable OSes and hardware, and applications. These entries help in computing incident ranking.

Fig. 5.5 Components of incident ranking

The M-correlator model assesses a security incident based on the incident ranking score. It is estimated with respect to three different components: alert outcome, relevance, and security incident priority (see in Fig. 5.5).

- **Alert outcome**: The alert outcome indicates that the incident has happened and also assumes that the INFOSEC device has correctly reported the incident. INFOSEC device provides the incident report along with a level of possibility.
- **Relevance**: Relevance indicates an appraisal of the likelihood of a successful intrusion. The M-correlator model computes the incident relevance score based on the attributes stored in the incident. It takes into account the topology of the target host including (a) the type and version of the operating system, (b) the type of hardware of the target host, (c) the service suite that is installed on the target host, (d) the network services that are enabled on the target host, and (e) the applications that are installed on the target host.

  The resultant incident relevance score values are between 0 and 255. 0 specifies that vulnerabilities required for the successful intrusion were not matched to the target host. It might reduce the overall score of incident ranking. A relevance score value close to 255 indicates the opposite situation.
- **Security incident priority**: The security incident priority is defined with respect to a security stream and a mission, where stream is the set of incident events $\{e_1, e_2, e_3, \cdots, e_n\}$. The main objectives of prioritization is to catch a set of events with high impact (HI), e.g., HI = $e_{h_1}, e_{h_2}, \cdots, e_{h_n} \subseteq Stream$ such that $\forall Threat_{\text{Rank}}(e_j, Mission) > T_{\text{acceptable}}$ with $e_j \in HI$. The mission has an objective to bring together and use the network assets. The mission specification has two major parts. The first part is the list of the most critical assets and services of the protected network, including critical computing assets, critical network services, sensitive data assets, and important user accounts. The second part is the type of incident, which is of major concern to the analyst. Examples of incident types are PRIVILEGE VIOLATION, PROBE, and DENIAL-OF-SERVICE. An analyst can specify low, medium-low, medium-high, and high interest for a particular incident.

  The incident ranking is defined as an assessment and ranking of events $\{e_1, e_2, \cdots, e_n\}$ with respect to mission profile (MP), $MP = \{CR_{\text{assets}}, CR_{\text{resources}}, CR_{\text{users}}, Incident_{\text{weight}}\}$ and the probability of success, which is estimated based on alert outcome and relevance.

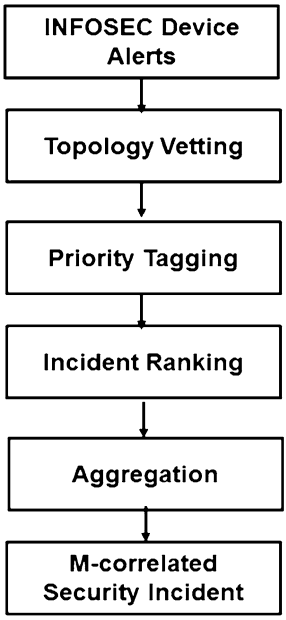


Fig. 5.6 Incident ranking estimation using M-correlator

The incident ranking is computed based on Bayes framework that is used to build belief propagation tree [44]. In the Bayes framework, each node maintains a table of conditional probabilities that represents the relationship between the child node and its parent node. The parent node contains the likelihood value of each node. Finally, the belief in hypothesis at the root node is estimated based on propagated belief in the hypothesis and the conditional probability tables at other nodes. The process of incident ranking estimation using the M-correlator is shown in Fig. 5.6 in terms of the incident rank tree containing outcome, priority, and relevance. Let H_i represent the ith hypothesis state of the root node and S_i, $i = 1, 2, 3$ represent the hypothesis state of its ith child nodes. The incident rank $P(H_i|S_1, S_2, S_3)$ is given below.

$$P(H_i|S_1, S_2, S_3) = \frac{P(H_i) \prod_{k=1}^{3} P(S_k|H_i)}{P(S_1, S_2, S_3)} \tag{5.2}$$

Each node's outcome, priority, and relevance can be estimated in a similar way with respect to the belief in a hypothesis.

An alert priority computation model similar to the M-correlator is proposed by Qin et al. [33], which is designed based on the Bayesian network framework. They use alert prioritization for scenario analysis instead of incident ranking. Lippman et al. [24] present an alert prioritization mechanism that uses vulnerability information to prioritize intrusion alerts. These

alert prioritization methods need additional information about the protected network to produce meaningful results when estimating alert ranking or priority.

5.1.1.5 Alert Prioritization Metrics

Alert prioritization is made based on alert priority score metrics. An alert score is likely to be better if it takes into consideration a large number of attributes as indication of alerts. A few alert score metrics [2] are described below.

1. **Applicability metric**: It identifies whether a raised alert for an attack is applicable to the current network environment. This requires information from different knowledge bases, such as a vulnerability knowledge base, and the set of running services, applications, and operating systems. Appropriate alert attributes help identify potential attacks. The vulnerability base indicates whether the attack is applicable to the current environment.
2. **Victim metric**: It specifies the properties of critical machines, services, applications, accounts, and directories in the current network environment. The main objective of this metric is to increase the alert score for anomalous activities to differentiate among them properly. Based on Eq. 5.2, it estimates the weighted projection of the alert metric to those alerts that are critical in nature:

$$I(M) = \prod_{x \epsilon \{serv, app, acct, dir\} running on M} \tag{5.3}$$

   where I represents the condition of the victim host running in the protected environment. The value of I is computed in the range between 0 and 1. 0 indicates that the reported alert does not have any importance with respect to the host, service, account, and directory. A value closer to 1 represents that an attack is targeting the system or a host in the network.
3. **Sensor status metric**: The sensor-based status metric is computed based on the Bayesian detection rate (BDR) formula. BDR estimates true positive probability $p(A|I)$ that an alert is raised when the attack I is detected:

$$P(A|I) = \frac{P(I)P(A|I)}{P(I)P(A|I) + P(\neg I)P(A|\neg I)} \tag{5.4}$$

   where A indicates that the alert has been raised, $\neg A$ indicates that no alert has been raised. I represents that an attack has occurred, and $\neg I$ indicates that no attack has occurred. The sensor states will be high if the location of the sensor is critical, its accuracy is high, and it is well configured and up-to-date. In such case, the alert confidence becomes high when an alert is generated from such sensors.

4. **Attack security metric**: This metric measures the risk level posed by a particular vulnerability. There are several attack security score metrics for known attacks such as MITRE, CVE, and Securia. It varies from organization to organization.
5. **Service vulnerability metric**: This metric is designed based on the services that an attacker is targeting. A unified score is computed representing both the strengths and weaknesses of the target host service.
6. **Social activity metric**: This metric is designed by exploiting features of a social network. Each node of the social network has an address, a target address, an attack ID, and the sites the users visited. Based on these alert attributes, it can be used to find hidden participants in a communication session.

5.1.1.6 Intention Recognition

Intention recognition of an attacker infers by collecting an intruder's behavior by observing regular activities and actions. It can be used to give an early warning and prevent an intrusion before it can damage the system or host. It is a very difficult task to recognize the behavior of attackers because attack behavior is unpredictable. Attack behavior changes with time and environment. For intention recognition, it is necessary to know the intruder's strategic plan. Intruder's strategy can be extracted using alert correlation.

Each intruder must perform several actions to achieve its goal. Modeling of multistage attack scenarios is a challenge. Alert correlation provides a facility to reconstruct typical attack scenarios. The next step should be alert correlation analysis that results in the intruder's behavior or strategy with respect to the target host.

Cuppen et al. [9] introduce an objective model to recognize the intruder's anomalous intention. This model has two fields: objective name and set of conditions, also known as state-condition. A state-condition represents the set of conditions that must be met to achieve the attacker's objective or goal set. This method still works based on the correlation of alerts with objectives. Geib and Goodman [18] present an intention recognition method for detecting and preventing attackers from attacking a target host or network at an early stage through plan recognition. The authors define a plan recognition system that can handle the following cases:

- **Observation of state change**: A state change in a protected system must indicate that there may have been some malicious actions during a certain period.
- **Unobserved action**: It is always possible that an attack might enter a host or network without being discovered by an IDS. A plan recognition system may be able to report the occurrence of unknown actions that are not detected by an IDS.
- **Partially ordered plan**: An attacker may be able to use several alternative sequences of actions to achieve its goals. A plan recognition system has to be designed in such a way that it is able to recognize different possible attack sequences created by the attacker.

- **Multiple concurrent goals**: An attacker may have multiple goals to achieve after getting certain level of access to the victim host or network. The plan recognition system may be able to discover multiple such goals.
- **Actions used for multiple effects**: An attacker may perform a single action that may have multiple effects. So, it is a critical requirement for a plan recognition system to be able to handle such a situation without affecting the target system.
- **Failure to observe**: When an abnormal activity happens, there are normally other activities that follow. Sometimes such activities that follow may not be recognized easily. A plan recognition system may be able to discover such activities that are part of a complex attack plan.
- **Impact of world state on adopted plans**: Important resources of the networks are targeted by an attacker by gaining knowledge of the network structure either directly or indirectly. A plan recognition system may also be able to get help from knowledge of network structure when protecting the network.
- **Consideration of multiple possible hypotheses**: An attacker's plan may be flexible in nature. For example, a single action may have multiple effects, and alternately multiple actions may fulfill a single goal. It is important for the plan recognition system to rank each action plan of the attacker.

5.1.2 Alert Validation (Verification)

It is a technique to verify a NIDS generated alert, i.e., to recognize if any changes have been taken place in the network or not. It can be determined based on an alert verification mechanism. Two major verification mechanisms are available. They are active verification and passive verification. Active verification mechanisms work in online mode and identify the location of an alarm and its effect in the network. Zhou et al. [51] analyze a protocol to verify the success of an attack on its target. They assume that an attack changes the behavior of victim programs that violate the protocol. The main idea to recognize the changes wrought by successful attack.

5.1.3 Alert Merging (Aggregation)

The main task of alert merging is data reduction by combining multiple alerts. It is necessary because a single attack generates multiple alerts as well as a large number of duplicate alerts. Alerts are merged based on alert attributes such as time stamp, source IP, destination IP, port(s), username, process name, attack class, and server ID. These attributes are defined in IDMEF (Intrusion Detection Message Exchange Format) attributes and can be fused together [32, 33, 36]. A similar method is proposed by Debar and Wespi [13], where they introduce an aggregation and correlation component (ACC) to discover duplicate relationships. A duplicate definition file is generated to store duplicate alerts that use the same reference alerts

when alerts are merged into a single alert. Sadoddin et al. [37, 38] consider three generic patterns to model every single-step attack.

- A one-to-many single step attack is represented as $A^{*<}$ to indicate a set of alerts with same source IP but multiple destination IPs.
- A many-to-one attack is an abstraction of similar types of alerts with a single destination IP and multiple source IPs. It is represented as $A^{*>}$.
- A multi-one-to-one attack indicates alerts with the same source IP and the same destination IP and is represented by $A^{*=}$.

5.1.4 Alert Clustering

An alert clustering technique uses a set of unlabeled alerts and generates clusters of similar type. A cluster may represent true or false alerts, which needs to be assigned later based on alert cluster properties. Julisch [22] uses clustering algorithms to create clusters of NIDS alerts. According to Julisch, majority of alerts are triggered due to misconfiguration of software. A way to identify this cause is known as root cause discovery. Learning the root cause of false alerts allows future alerts to be classified as true or false.

5.1.5 Alert Correlation Architectures

An alert correlation architecture may be classified as centralized or hierarchical. The alert correlation architecture is agent based. In a centralized architecture, all alerts are usually sent to the correlation center for processing. The center analyzes the alerts and presents the output to the next level. Alternatively, alerts may be locally correlated first and then sent to the centralized correlator for global correlation.

(a) **Centralized correlation**: Carey et al. [5] present a prototype for IDS interoperability. Each correlation framework has three major components.

- *Alert agent*: It is available within a NIDS and performs translation of alert messages to IDMEF format. Finally it passes the messages to the control unit for further processing.
- *Control unit*: It takes IDMEF messages, preprocesses, and stores them in a database.
- *Administrative control*: The alerts stored in a database are processed centrally in offline mode and are correlated.

(b) **Hierarchical correlation**: This correlation architecture is designed hierarchically and separated into local analysis, regional analysis, and global analysis [34]. The detection agents are installed locally to identify anomalous behavior in a network. The correlator engine is responsible for a region and sends its

report to the intrusion detection manager, which combines the outputs of local agents deployed in different locations. The following are the major components of an intrusion detection agent:

- *Detection module*: This module uses three different levels of detection engines, viz., signature, profile, and MIB engines. The signature engine generates alarms based on known signatures, whereas the profile engine generates alarms based on anomalousness with respect to the legitimate user and network behaviors. The MIB engine checks the value of MIB objects and compares with the MIB profiles.
- *Knowledge base*: It stores attack signatures and user and network MIB profiles.
- *Response module*: It decides the type of countermeasures that need to be taken after detection.
- *Control module*: It updates the knowledge-base and detection engine based on alert information serviced by the correlator agents.
- *Alarm engine*: It sends alarms to the detection engine.
- *SNMP interface*: It uses SNMP packets to transport information between the intrusion detection agent and the correlator. The intrusion detection correlator is designed to correlate alarms received from intrusion detection agents. It communicates with the intrusion detection correlator manager for global correlation among alarms. The major components of an intrusion detection correlator engine are the following:
  - Alarm correlation module: It receives alarm information from different agents.
  - Security correlation engine: It identifies and predicts intrusions and their trends based on the correlated alarms received from intrusion detection agents.
  - Knowledge database: It keeps information necessary for the correlation algorithm to work.
  - Intrusion response module: For a specific domain, it generates responses based on the defined security policies.
  - Knowledge controller: It manages the knowledge database as updates or changes take place.

5.1.6 Validation of Alert Correlation Systems

Validation is an important part of an alert correlator system. An experimental approach to validation of a correlation system is introduced by Haines et al. [19]. The authors define three major dimensions to recognize attacks and identify the target.

- Prioritization: It assigns weights to each alert based on the probability that it may indicate an attack and dispensation of the target.
- Multistep correlation: The alert correlator can reconstruct a multistep attack scenario by correlating different individual attack steps. This step is important to infer attack intention and their effective response.
- Multi-sensor correlation: It combines multiple alerts received from different sensors to create an overall picture of the system.

The authors demonstrate that most alert correlation models are able to reduce alert volume significantly as well as obtain high-level reasoning results such as multistage attack recognition.

5.2 Network Intrusion Prevention Techniques

Network intrusion prevention systems (NIPS) have gained in popularity because security professionals are increasingly focusing on stopping potential attacks before they become a threat to a host or network. A NIPS can be thought of as a refined firewall that is able to deny entrance to hostile traffic, while legitimate traffic is allowed to enter the network.

5.2.1 Understanding NIPS

To avoid complete failure of networks due to attacks, a NIPS may be deployed to achieve the following objectives:

- A high level of performance,
- Multi-method event detection,
- Up-to-the second intelligence,
- Continuous monitoring.

5.2.1.1 Types of NIPS

Like IDSs, IPSs can be classified as host-based IPS and network-based IPS based on deployment location.

1. *Host-based IPS (HIPS)*: An HIPS is usually deployed as an agent in each host of the network to prevent entry of attack into the individual host. The agent binds closely to the operating system kernel and services, monitoring and intercepting system calls to the kernel or APIs in order to prevent attacks as well as log them. Since a host-based IPS obstructs all requests to the host it protects, it must satisfy certain strict perquisites. For example, it must be reliable, should not affect overall performance of the network, and must not block legitimate traffic.

The main disadvantage of a host-based IPS is its integration with the operating system. On the flip side, operating system upgrades can cause problems.

2. *Network-based IPS (NIPS)*: A Network-based IPS is deployed in a network to monitor the in-out traffic of the network to identify possible attacks and prevent them at the entry point. The network has to be configured in such a way that all traffic must go through the NIPS. A NIPS has two interfaces, internal and external. Packets at any interface are forwarded to the detection engine to check as malicious or legitimate.

   A NIPS is placed inside the network and all packets pass through it. Once a packet has been detected as malicious before it is passed to the internal interface and on to the protected network, it can be dropped and flagged as suspicious. So, all subsequent such packets that are part of that session can be dropped at the entry point with small additional processing.

5.2.1.2 NIPS vs. Firewall

Both NIPS and firewall are crucial for protection of enterprise networks from large-scale attacks. It is because of the following reasons:

- A NIPS analyzes a stream of traffic (i.e., packet payload) to identify the attacks and prevent them at entry point. Figure 5.7 shows how a NIPS inspects the packet payload and how a firewall inspects the packet header.
- A firewall is designed to inspect the packet header information irrespective of packet payload to identify attacks. The packet header information usually includes source IP, destination IP, port numbers, protocol type, and so on.

An IPS sensor may be connected to the network in one of the following ways:

- *Inline*: The IPS is placed behind a firewall, router, or switch, so that all traffic passes through it.
- *Network tap*: It is a hardware device that allows access to the traffic flowing through the network.
- *Switch span port*: This port is monitored continuously because all traffic that passes through it flows through the switch.

Common places to place IPS sensors are the following:

- Perimeter or DMZ,
- Core or data center network,
- Extranets,
- Wireless access points,
- Virtualization platforms,
- Critical network segments.

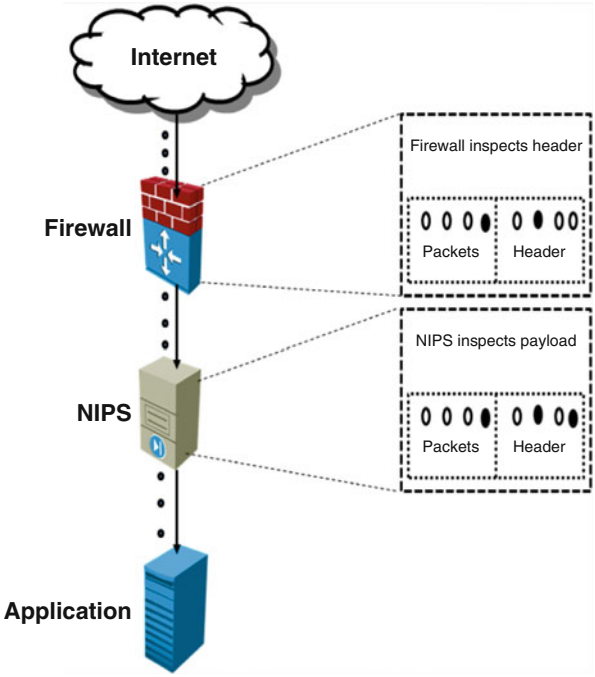


Fig. 5.7 How NIPS and firewall inspect network traffic

5.2.2 Criteria for NIPS Selection

A NIPS can be chosen based on the characteristics that are important in a certain environment.

- *Detection*: Whether an NIPS can detect malicious traffic based on signature based, anomaly based, or both.
- *Scalability*: A NIPS must have the ability to expand the configuration to support additional bandwidth and technologies.
- *Compliance*: A NIPS must satisfy any relevant governmental or industry compliance regulations that affect an organization.
- *Performance*: A NIPS should perform well after updating of any software components. An updated version must work without replacing any hardware.
- *Viability*: A company that developed the NIPS can continue to provide excellent long-term support to the customer.
- *Manageability*: A NIPS should be easily manageable so that system administrator can configure it as per requirement of the organization.
- *Support*: A vendor must be chosen based on the support provided, because the organization using NIPS may need support in good or bad times.
- *Cost*: It is based on what an individual organization can afford.

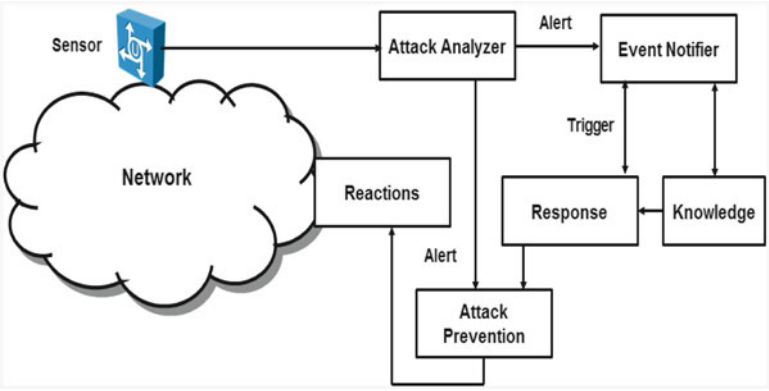


Fig. 5.8 NIPS: A generic architecture

5.2.3 Prevention Techniques

Both intrusion detection systems (IDS) and intrusion prevention systems (IPS) play crucial roles in the protection of enterprise networks from different types of attacks. An IPS follows a proactive technique, so that when an attack is detected, the IPS blocks immediately and logs the offending data. In some instances, an IPS imitates techniques used by an IDS such as the detection mechanism, the nature of the monitoring sensors, and the alert mechanism [41]. A generic architecture of a network-based IPS is given in Fig. 5.8.

The data collection sensor is one of the parts that may act fussy in all classes of IPSs. It is because the capacity and performance of sensor are usually limited in the amount of traffic that it can be processed. The use of data collection sensor makes it easy to monitor the network and generate alert incident responses, notify administrators, or block traffic quickly. There is a challenge in evaluating the log files because a data sensor generates huge number of log files by logging data transactions, logging attacker traffic, logging victim traffic, logging summary reports, and logging failure reports [40].

One major problem faced by detection using an IPS is that it is difficult to analyze traffic in real time, especially in high-speed networks. Network intrusion prevention techniques are classified into three major classes including signature or pattern-based prevention techniques, anomaly-based prevention techniques, and behavior-based techniques [43].

1. *Pattern-based prevention techniques*: Pattern-based techniques, also known as signature-based techniques, are used to identify a specific pattern, which is not behaving legitimately. It is again classified into two mechanisms, viz., pattern detection and deobfuscation techniques [6, 30, 48]. A pattern detection technique uses a pattern matching language that enables one to define a flexible string matching patterns. On the other hand, deobfuscation focuses on extracting the concrete syntax of the program.

2. *Anomaly-based prevention techniques*: This is also known as profile-based prevention, because we must build profiles that define normal or legitimate activity. So, anything that deviates from the legitimate profiles immediately generates alerts [23, 50].
3. *Behavior-based prevention techniques*: This is similar to pattern prevention techniques, where the behavior defines the amount of suspiciousness [17].

5.3 Chapter Summary

This chapter has examined various alert-related measures and network intrusion prevention techniques. There are several alert management techniques available in the literature. We cover most common mechanisms where alerts are managed efficiently in terms of storage, processing, and analysis. Topics one needs to know to understand alert management well include alert normalization, alert representation, alert correlation, alert validation and alert merging, alert clustering, and alert correlation architectures. In the second part of the chapter, we cover the basic concept of NIPS, types of NIPS, comparison of NIPS with firewall, and network intrusion prevention techniques. We make the following observations:

- Normalizing alerts based on alert attributes is challenging. It is because there is a possibility of loss of data during normalization. So, it may impact on analysis.
- Validation of an alert after all operation is an issue yet to be addressed properly.
- Deployment of a NIPS in a real high-speed network is also a challenging task.

We hope that the reader is able to gain a good understanding of alert management and network intrusion prevention techniques with their applicability by the end of this chapter.

References

1. AlienVault: Alienvault unified security management, data-sheet (2013). USM Appiliance
2. Alsubhi, K., Al-Shaer, E., Boutaba, R.: Alert prioritization in intrusion detection systems. In: NOMS 2008 – 2008 IEEE Network Operations and Management Symposium, pp. 33–40 (2008). doi:10.1109/NOMS.2008.4575114
3. Beng, L.Y., Ramadass, S., Manickam, S., Fun, T.S.: A survey of intrusion alert correlation and its design considerations. IETE Tech. Rev. **31**(3), 233–240 (2014). doi:10.1080/02564602.2014.906864
4. Carey, N.: Correlations of heterogeneous IDS alerts for attack detection. Master's thesis, Information Security Research Centre, Faculty of Information Technology, Queensland University of Technology, Australia (2004)
5. Carey, N., Clark, A., Mohay, G.: IDS Interoperability and Correlation Using IDMEF and Commodity Systems, pp. 252–264. Springer, Berlin/Heidelberg (2002). doi:10.1007/3-540-36159-6_22

6. Carter, E.: Intrusion prevention fundamentals: an introduction to network attack mitigation with IPS. Technical Report, Cisco Press (2006)
7. Cheung, S., Lindqvist, U., W Fong, M.: Modeling multistep cyber attacks for scenario recognition. In: DARPA Information Survivability Conference and Exposition (DISCEX III), pp. 284–292. Washington, DC (2003). http://www.sdl.sri.com/papers/cheung-lindqvist-fong-discex3-cr/
8. Cuppens, F.: Managing alerts in a multi-intrusion detection environment. In: Proceedings of the 17th Annual Computer Security Applications Conference, ACSAC'01, pp. 22–31. IEEE Computer Society, Washington, DC (2001)
9. Cuppens, F., Miege, A.: Alert correlation in a cooperative intrusion detection framework. In: Proceedings 2002 IEEE Symposium on Security and Privacy, pp. 202–215 (2002). doi:10.1109/SECPRI.2002.1004372
10. Cuppens, F., Ortalo, R.: LAMBDA: A Language to Model a Database for Detection of Attacks, pp. 197–216. Springer, Berlin/New York (2000)
11. Curry, D., Debar, H.: Intrusion detection message exchange format (2003). Http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-10.txt
12. Dain, O., Cunningham, R.K.: Fusing a Heterogeneous Alert Stream into Scenarios, pp. 103–122. Springer, Boston, MA (2002)
13. Debar, H., Wespi, A.: Aggregation and correlation of intrusion-detection alerts. In: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection, RAID'00, pp. 85–103. Springer, London (2001)
14. Eckmann, S.T., Vigna, G., Kemmerer, R.A.: STATL: an attack language for state-based intrusion detection. J. Comput. Secur. **10**(1–2), 71–103 (2002)
15. Elshoush, H.T., Osman, I.M.: An improved framework for intrusion alert correlation. In: Proceedings of the World Congress on Engineering, London, vol. I, pp. 1–6 (2012)
16. Feng, C., Peng, J., Qiao, H., Rozenblit, J.: Alert fusion for a computer host based intrusion detection system. In: Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, pp. 433–440. IEEE Computer Society (2007)
17. Frias-Martinez, V., Stolfo, S.J., Keromytis, A.D.: Behavior-profile clustering for false alert reduction in anomaly detection sensors. In: Proceedings of the 2008 Annual Computer Security Applications Conference, ACSAC'08, pp. 367–376. IEEE Computer Society, Washington, DC (2008). doi:10.1109/ACSAC.2008.30
18. Geib, C.W., Goldman, R.P.: Plan recognition in intrusion detection systems. In: Proceedings of the DARPA Information Survivability Conference amp; Exposition II, 2001. DISCEX'01, vol. 1, pp. 46–55 (2001). doi:10.1109/DISCEX.2001.932191
19. Haines, J., Ryder, D.K., Tinnel, L., Taylor, S.: Validation of sensor alert correlators. IEEE Secur. Priv. **1**(1), 46–56 (2003). doi:10.1109/MSECP.2003.1176995
20. Hubballi, N., Suryanarayanan, V.: Review: false alarm minimization techniques in signature-based intrusion detection systems: a survey. Comput. Commun. **49**, 1–17 (2014)
21. IETF IDWG: IETF intrusion detection working group, intrusion detection message exchange format (2004). http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-12.txt
22. Julisch, K.: Clustering intrusion detection alarms to support root cause analysis. ACM Trans. Inf. Syst. Secur. **6**(4), 443–471 (2003). doi:10.1145/950191.950192
23. Le, A., Al-Shaer, E., Boutaba, R.: On optimizing load balancing of intrusion detection and prevention systems. In: IEEE INFOCOM Workshops 2008, pp. 1–6 (2008). doi:10.1109/INFOCOM.2008.4544576
24. Lippmann, R., Webster, S., Stetson, D.: The Effect of Identifying Vulnerabilities and Patching Software on the Utility of Network Intrusion Detection, pp. 307–326. Springer, Berlin/Heidelberg (2002). doi:10.1007/3-540-36084-0_17
25. Maggi, F., Zanero, S.: On the use of different statistical tests for alert correlation - short paper. In: Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection (RAID'07), Gold Goast, 05-07 Sept, pp. 167–177. Springer, Berlin/Heidelberg (2007)

26. Maggi, F., Zanero, S.: On the use of different statistical tests for alert correlation: short paper. In: Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection, RAID'07, pp. 167–177. Springer, Berlin/Heidelberg (2007)
27. McAfee: Advanced Correlation Engine (2017). http://www.mcafee.com/in/products/advanced-correlation-engine.aspx
28. Mirheidari, S.A., Arshad, S., Jalili, R.: Alert Correlation Algorithms: A Survey and Taxonomy, pp. 183–197. Springer, Berlin/New York (2013)
29. Ning, P., Cui, Y., Reeves, D.S.: Analyzing Intensive Intrusion Alerts Via Correlation, pp. 74–94. Springer, Berlin/Heidelberg/Zurich (2002)
30. Ollmann, G.: Intrusion prevention systems (IPS) destined to replace legacy routers. Netw. Secur. **03**(11), 18–19 (2003)
31. Pietraszek, T.: Alert classification to reduce false positives in intrusion detection. Ph.D. thesis, Institut fur Informatik, Albert-Ludwigs-Universit, Germany (2006)
32. Porras, P.A., Fong, M.W., Valdes, A.: A mission-impact-based approach to INFOSEC alarm correlation. In: Proceedings of the 5th International Conference on Recent Advances in Intrusion Detection, RAID'02, pp. 95–114. Springer (2002)
33. Qin, X., Lee, W.: Statistical Causality Analysis of INFOSEC Alert Data, pp. 73–93. Springer, Berlin/Heidelberg (2003)
34. Qin, X., Lee, W., Lewis, L., Cabrera, J.B.D.: Integrating intrusion detection and network management. In: Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP, pp. 329–344 (2002). doi:10.1109/NOMS.2002.1015591
35. Ren, H., Stakhanova, N., Ghorbani, A.: An online adaptive approach to alert correlation. In: DIMVA'10: Proceedings of the 7th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Lecture Notes in Artificial Intelligence, pp. 153–172. Springer (2007)
36. Sadoddin, R., Ghorbani, A.: Alert correlation survey: framework and techniques. In: Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services, pp. 37:1–37:10. ACM, New York (2006). doi:10.1145/1501434.1501479
37. Sadoddin, R., Ghorbani, A.A.: Real-time alert correlation using stream data mining techniques. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, 13–17 July 2008, pp. 1731–1737 (2008)
38. Sadoddin, R., Ghorbani, A.A.: An incremental frequent structure mining framework for real-time alert correlation. Comput. Secur. **28**(3–4), 153–173 (2009). doi:10.1016/j.cose.2008.11.010
39. Siraj, A.: A unified alert fusion model for intelligent analysis of sensor data in an intrusion detection environment. Ph.D. thesis, Mississippi State University, Mississippi (2006)
40. Stiawan, D., Abdullah, A.H., Idris, M.Y.: The trends of intrusion prevention system network. In: 2010 2nd International Conference on Education Technology and Computer, vol. 4, pp. 217–221 (2010). doi:10.1109/ICETC.2010.5529697
41. Stiawan, D., Abdullah, A.H., Idris, M.Y.: Characterizing network intrusion prevention system. Int. J. Comput. Appl. **14**(1), 11–18 (2011)
42. Thurman, W., Fisher, M.: Chickens, eggs, and causality, or which came first? Am. J. Agric. Econ. **70**(2), 237–244 (1998)
43. Tzur-David, S.: Network intrusion prevention systems: signature-based and anomaly detection. Ph.D. thesis, The Hebrew University of Jerusalem (2011)
44. Valdes, A., Skinner, K.: Adaptive, Model-Based Monitoring for Cyber Attack Detection, pp. 80–93. Springer, Berlin/Heidelberg (2000)
45. Valdes, A., Skinner, K.: Probabilistic Alert Correlation, pp. 54–68. Springer, Berlin/Heidelberg (2001). doi:10.1007/3-540-45474-8_4
46. Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R.A.: A comprehensive approach to intrusion detection alert correlation. IEEE Trans. Dependable Secur. Comput. **1**(3), 146–169 (2004). doi:10.1109/TDSC.2004.21

47. Wang, L., Liu, A., Jajodia, S.: Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. Comput. Commun. **29**(15), 2917–2933 (2006). doi:10.1016/j.comcom.2006.04.001
48. Weinsberg, Y., Tzur-David, S., Dolev, D., Anker, T.: High performance string matching algorithm for a network intrusion prevention system (nips). In: 2006 Workshop on High Performance Switching and Routing, pp. 147–153 (2006). doi:10.1109/HPSR.2006.1709697
49. Yu, J.: TRINETR: an intrusion detection alert management and analysis system. Ph.D. thesis, West Virginia University, Morgantown, West Virginia (2004)
50. Zhang, X., Li, C., Zheng, W.: Intrusion prevention system design. In: The Fourth International Conference on Computer and Information Technology, CIT'04, pp. 386–390 (2004). doi:10.1109/CIT.2004.1357226
51. Zhou, J., Carlson, A., Bishop, M.: Verify results of network intrusion alerts using lightweight protocol analysis. In: 21st Annual Computer Security Applications Conference, pp. 1063–9527 (2005). doi:10.1109/CSAC.2005.62

Chapter 6
Practical Tools for Attackers and Defenders

A tool is usually developed for a specific purpose with respect to a specific task. For example, *nmap* is a security scanning tool to discover open host or network services. Network security tools provide methods to network attackers as well as network defenders to identify vulnerabilities and open network services. This chapter is composed of three major parts, discussing practical tools for both network attackers and defenders. In the first part, we discuss tools an attacker may use to launch an attack in real-time environment. In the second part, tools for network defenders to protect enterprise networks are covered. Such tools are used by network defenders to minimize occurrences of precursors of attacks. In the last part, we discuss an approach to develop a real-time network traffic monitoring and analysis tool. We include code for launching of attack, sniffing of traffic, and visualization them to distinguish attacks. The developed tool can detect attacks and mitigate the same in real time within a short time interval. Network attackers intentionally try to identify loopholes and open services and also gain related information for launching a successful attack.

6.1 Steps to Launch an Attack

Attackers attempt to discover vulnerabilities and loopholes of target hosts or target networks before launching an attack. Attackers scan the network to discover open services and major loopholes of hosts or systems. This information is exploited to launch an attack using malicious code available on the Internet. It may first compromise a single host of the network and then exploit more loopholes to disrupt the entire network within a certain time interval, which may be short or long. Attackers usually use five major steps to launch an attack on a target host or target network. Each step is discussed below briefly.

M.H. Bhuyan et al., *Network Traffic Anomaly Detection and Prevention*,
Computer Communications and Networks, DOI 10.1007/978-3-319-65188-0_6

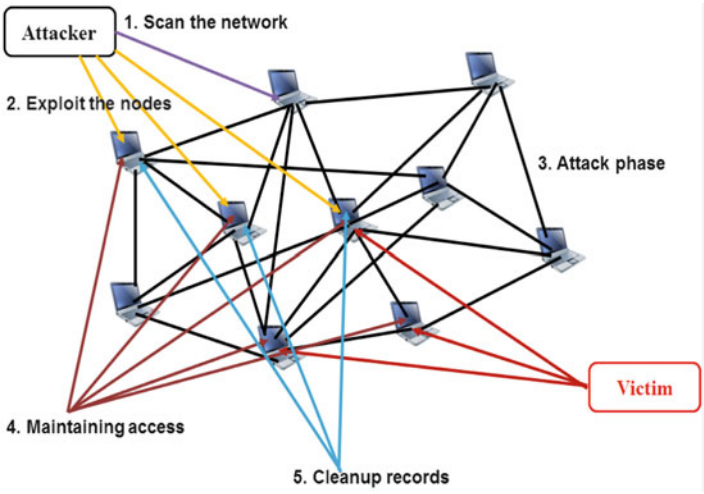


Fig. 6.1 Steps to launch an attack

1. *Information Gathering*: The attacker attempts to gather vulnerability information from the network with the hope that some of the information can be used to aid in the ensuing attack.
2. *Accessing Vulnerability*: Based on the vulnerabilities learned in the previous step, the attacker attempts to compromise some nodes in the network by exploiting malicious code, as a precursor to the launching of attack(s).
3. *Launching Attack*: The attacker launches the attack on the victim machine(s) using the compromised nodes.
4. *Maintaining Access*: After gaining access, the attacker must maintain the access for a long time to attain objectives. Attacker's vulnerability for detection increases in this phase.
5. *Clean up Footprints*: Finally, the attacker attempts to eliminate the attack history by cleaning up all the registry or log files from the victim machine(s).

An illustration of these steps is shown in Fig. 6.1.

One can also use attack launching tools such as Dsniff [8], IRPAS [30], Ettercap [20], and Libnet [23] to generate MAC attacks, ARP attacks, or VLAN attacks. The main purpose of the attacker is to disrupt services provided by the network either by consuming resources or consuming bandwidth. These types of attacks can be launched by flooding legitimate requests such as TCP SYN Flooding, ICMP flooding, and UDP flooding.

6.2 Impact of Network Security Tools

To detect an attack, one must know the characteristics of an attack and its behavior in a network. The network administrator needs a visualization or monitoring system to observe differences between the characteristics of abnormal traffic and the normal [11]. An attack can be detected from the traffic volume by looking up the packet header or network flow information. However, such detection usually requires processing huge volumes of data in near real time. Obviously, designing a real-time defense mechanism that can identify all attacks is a challenging and quite likely impossible task. Most detection methods need some prior information about attack characteristics to use during the detection process. The evaluation of these intrusion detection mechanisms or systems is performed using misclassification rate or false alarm rate. To obtain satisfactory results, an IDS designer needs to be careful in choosing an approach, matching mechanism or heuristic, or making assumptions. Approaches that have been able to obtain acceptable results include statistical [29], soft computing [7], probabilistic [13], knowledge-based [9], and hybrid [1]. A detailed discussion of these approaches is available in [3, 4].

To launch an attack, the attacker must know weaknesses or vulnerabilities in the network. Actual attacks exploit these vulnerabilities. The vulnerabilities can be obtained by scanning the network in an information gathering step. After gathering vulnerability information, the attacker tries to exploit the weakness(es) of the security system to launch an attack. There are different methods for launching an attack. For example, one may use Trojans or worms to generate an attack in a system or a network. Also, scanning or information gathering may be coordinated with an attack and performed simultaneously. One can also use attack launching tools to generate attacks in a network. The main purpose of the attacker is to disrupt services provided by the network either by consuming resources or consuming bandwidth. These types of attacks can be launched using flooding of legitimate requests such as TCP SYN Flooding, ICMP flooding, and UDP flooding.

At the current time, there are many attack launching tools and systems to generate network attacks in the public domain. These tools can be used on any one of the network layers in the TCP/IP network model to launch an attack, but most are used in network and transport layers. Common application layer attacks include HTTP-, FTP-, SMTP-, and DNS-related attacks. Most application layer attacks are generated by malware (e.g., Trojans, viruses, worms, and backdoors). HTTP-related attacks such as DDoS using HTTP GET request, Cross-Site Scripting (XSS), and SQL Injection are very common attacks on this layer. Modern tools such as slowhttptest and AppDDoS are now available to generate such attacks. In addition to these, some popularly used data link layer tools such as Dsniff [8], IRPAS [30], Ettercap [20], Libnet [23], and Gobbler are used to generate MAC attacks, ARP attacks, and VLAN attacks.

Identification of known as well as unknown anomalies with high detection accuracy and with minimum false alarms is the major concern in today's security environment. Hackers have a number of offensive tools to launch attacks on a

network. Similarly, there are modern security tools and systems to detect intrusions occurring in a network. To detect an attack, most systems capture network traffic, extract or select relevant features, and then analyze the traffic information over a relevant subspace of features, with or without the knowledge of previous references. Snort is a well-known lightweight packet sniffer as well as a detection system. It can watch for activities such as Queso TCP/IP fingerprinting scans, Nmap scans, or various types of probes. With the help of these sniffing or attack tools, it is very easy for hackers to steal network information. So, it is the responsibility of network defenders to build protection mechanisms incorporating attack launching and detection software tools that are available. Usually, antivirus software tools and spam detectors are recommended for network security, at a minimum. A network intrusion detection system is deployed to protect a network from attackers or intruders.

6.3 Taxonomy of Network Security Tools

People use different attack tools to disrupt a network for different purposes. As mentioned earlier, attackers generally target Websites or databases as well as enterprise networks by gathering information based on their weaknesses. In general, attackers use relevant tools for the class of attack they desire to launch. A large number of defense tools have also been made available by various network security research groups as well as security professionals. These tools have different purposes, capabilities, and interfaces.

We categorize existing tools into two major categories: tools for attackers and tools for network defenders. A taxonomy of the tools used in network security is shown in Fig. 6.2. For each basic category, we show subcategories considering their general characteristics.

6.4 Tools for Attackers

There are many tools available in the areas of network security to discover vulnerability, launching attacks, monitoring networks traffic, analysis, and visualization. We classify them into two major classes, tools for attackers and tools for defenders. Several tools are commonly used in both classes. For example, information gathering tools are commonly used by both attackers and defenders.

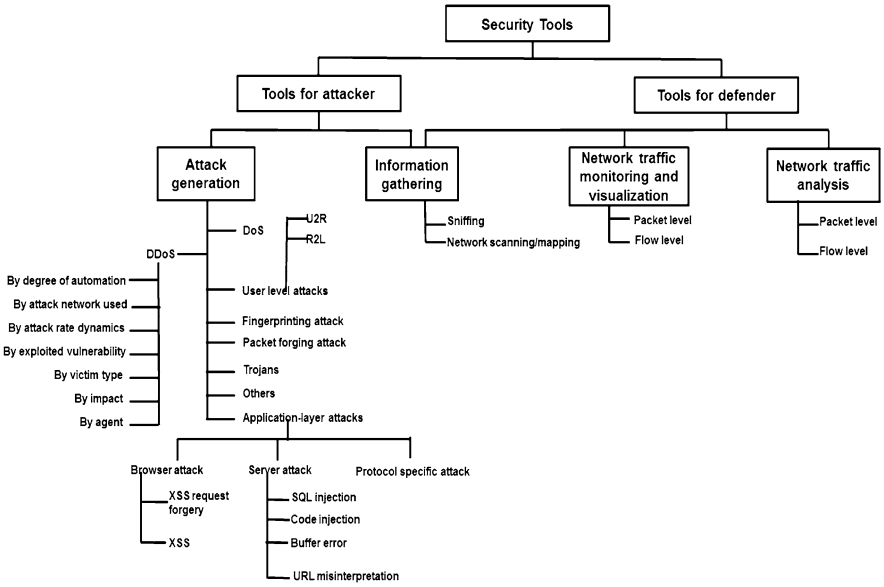


Fig. 6.2 Taxonomy of network security tools

6.4.1 Information Gathering Tools

Before launching an attack, attackers need to understand the environment where the attack is to be launched. To do so, attackers first gather information about the network such as open port numbers of machines, operating systems, and so forth. After gathering information, attackers find weaknesses in the network using various tools. Information gathering tools are further classified as sniffing tools and network mapping/scanning tools.

6.4.1.1 Sniffing Tools

A sniffing tool aims to capture, examine, analyze, and visualize packets or frames traversing across the network. To support extraction of additional packet features and for subsequent analysis, it also requires the protocol information during visualization. Some packet sniffing tools are discussed below (Table 6.1):

(a) *Angst*: Angst runs on Linux and OpenBSD and is an active packet sniffer that can capture data on switched networks by injecting data into the network. Angst is able to flood a network using random MAC addresses, by causing switches to transmit packets toward all ports.

Table 6.1 Comparison of sniffing tools

Tool's name	Input	Protocol	Purpose	Effectiveness	Sources
Ethereal	I	C/HT/S	Capturing	Powerful, user friendly	www.ethereal.com.
Tcpdump	I	C/U/IC	Capturing	Less intrusive than Ethereal	www.tcpdump.org
Net2pcap	I	C/U/IC	Capturing	Linux based, auditable	www.secdev.org
Snoop	N	C/U/IC /Te/F	Capturing	No packet loss, supports more than 12 options	www.softpanorama.org
Angst	H/p	C	Sniffing	Easy to use, aggressive	www.angst.sourceforge.net
Ngrep	I	C/U/IC	Capturing	Multi-platform, handle large data	www.ngrep.sourceforge.net
Ettercap	I	C/U	Man-in-middle attack	Efficient, support more that 35 options	www.ettercap.sourceforge.net
Dsniff	I	F/Te/S/HT/P	Password sniff	Unix based	www.naughty.monkey.org
Cain & Abel	I		Password recovery	Easily accessible	www.oxid.it
Aimsniff	H	C/U/HT	Easy accessible	Linux based	www.sourceforge.net
Tcptrace	F	C	Analysis of traffic	High applicability	www.tcptrace.org
Tcptrack	I/p	C	TCP connection analysis	Linux based	www.rhythm.cx
Hexinject	I/p	C	Packet inspector and sniffer	Linux based	www.hexinject.sourceforge.net
Argus	F	C/U	Analysis of audit data	Muti-platform, real-time processing	www.qosient.com/argus
Karpski	I	C/U	Packet analyzer	Easy sniffer, limited applicability	www.softlist.net
Junkie	I	C/U/HT /D/IC	Packet sniffer and analyzer	Easy sniffer, limited applicability	www.performancevision.com
IPgrab	I/p		Display packet header	Displays packet details	www.ipgrab.sourceforge.net
Nast	I	C/U	traffic analysis	supports more than 12 options	www.nast.berlios.de

(continued)

Table 6.1 (continued)

Tool's name	Input	Protocol	Purpose	Effectiveness	Sources
GULP	I	C/U/IC	Packet capturing/visualization	Very efficient, easily accessible	www.staff.washington.edu/corey
Libpcap	I	C/U/IC	Packet capturing	High performance	www.tcpdump.org
NFSEN	I	C/U	Flow capturing/visualization	Easy navigation of NetFlow data	www.nfsen.sourceforge.net
NFDUMP	I	C/U	Flow capturing/visualization	Powerful packet analyzer	www.nfdump.sourceforge.net

I Interface ID, *N* Network IP, *H* Host IP, *p* port, *F* Traffic captured file, *C* TCP, *U* UDP, *IC* ICMP, *IG* IGMP, *HT* HTTP, *S* SMTP, *F* FTP, *P* POP, *Te* Telnet, *D* DNS

(b) *Aimsniff*: It is a simple tool to capture IP address of an AOL Instant Messenger user while a direct connection is established between the user with the others. Once the connection is established, one is able to simply click on the sniff button to capture the IP address.

(c) *Argus*: Argus runs on several operating systems, such as Linux, Solaris, MAC OS X, FreeBSD, OpenBSD, NetBSD, AIX, IRIX, Windows, and OpenWrt. It can process either live traffic or captured traffic files and can output status reports on flows detected in the stream of packets. Its reports reflect flow semantics. This tool provides information on almost all packet parameters, such as reachability, availability, connectivity, duration, rate, load, loss, jitter, retransmission, and delay metrics for all network flows.

(d) *Aldebaran*: It is an advanced libpcap-based sniffing and filtering tool for the TCP protocol. It provides basic information about the source and destination addresses and ports but no information regarding flags. One can use it to monitor data sent by connections as well as to sniff passwords. Based on libpcap rules, one can use it to sniff packet headers as well as payload contents and can transmit captured traffic to another host via UDP. Aldebaran also allows (i) to encrypt the content saved in dump files, (ii) to analyze interface traffic, and (iii) to report packet statistics, viz., *packet count, size*, and *average speed* in HTML or as a plain text file.

(e) *Cain & Abel*: It is a multipurpose sniffer tool that runs on Windows NT, 2000, and XP and allows for password recovery for a number of protocols, including MSN messenger and RADIUS shared keys. It can also launch man-in-the-middle attacks for SSHv1 traffic.

(f) *Dsniff*: Dsniff is a collection of tools that enable active sniffing on a network. It can perform man-in-the-middle attacks against SSHv1 and HTTPS sessions. It can also sniff switched networks by actively injecting data into the network and redirecting traffic.

(g) *Ettercap*: Ettercap is a very good sniffer that runs on almost all platforms. More of an active hacking tool, Ettercap uses an ncurses interface and is able to decode several protocols. Ettercap operates in multipurpose mode: sniffer and

interceptor or logger mode for switched LANs. Ettercap can collect passwords for multiple applications, kill connections, inject packets, inject commands into active connections, and has additional plugins.

(h) *Ethereal*: Ethereal is a sniffing and traffic analyzing software tool for Windows, Unix, and Unix-like OSs, released under the GNU license scheme. It includes two primary library utilities, (i) *GTK+*, a GUI based library, and (ii) *libpcap*, a packet capture and filtering library. Ethereal is also capable of reading the output of *tcpdump* and can apply tcpdump filters to select and display records satisfying certain parameters. Ethereal offers decoding options for a large number ($\geq$400) of protocols and is useful in network forensics. It supports preliminary inspection of attacks in the network.

(i) *GULP*: It is used to capture very high volume of network traffic efficiently from the network firehose. It overcomes the packet loss problem of tcpdump and records a large amount of data that are stored in secondary storage for further processing. It can capture packets from multiple CPUs for better performance and writes the captured data in pcap files.

(j) *Hexinject*: Hexinject is a powerful, versatile packet injector and sniffer. It provides command-line access to raw networks and facilitates the creation of powerful shell scripts capable of reading, intercepting, and modifying network traffic in a transparent manner.

(k) *IPgrab*: This packet sniffing tool provides facility for network debugging at multiple layers, such as data link, network, and transport layers. It outputs detailed header field information for all layers.

(l) *Junkie*: Junkie is designed for real-time packet sniffing and analyzing. So, it stands somewhere between Wireshark and tcpdump. Features of junkie include IP reassembly, TCP reordering, and parsing of protocols like HTTP, DNS, or SIP and are extendable to C.

(m) *Karpski*: This is a user-friendly tool with limited sniffing and scanning capabilities. It provides flexibility to include protocol definitions dynamically and also can serve as an attack launching tool against addresses on a local network.

(n) *Nfsen*: Nfsen is used to visualize NetFlow flow data and to display the captured data such as flows, packets, and bytes in graphical interface. Also, we can visualize protocol-specific flows in a graphical format using nfsen.

(o) *Nfdump*: It is used to collect and process the NetFlow data. It reads NetFlow data from the files created by nfcapd. It organizes captured data in a time-based fashion, typically for 5 min, and stores them for further processing. Analyzing data can be done for a single file or by concatenating several files for a single run. The output is either ASCII text or binary data, when saved into a file, ready to be processed again with the same tool.

(p) *Nast*: Nast uses *libnet* and *libpcap* to sniff packets in normal mode or in promiscuous mode to analyze them. It captures packet header parameters, payload information, and saves them in a file in ASCII or ASCII-hex format.

(q) *Nstreams*: It is a tool to display and analyze network streams generated by users between several networks and between networks and the outside. Nstreams

also can output optionally the *ipchains* or *ipfw rules* matching these streams. It parses outputs generated by tcpdump or files generated using tcpdump with *-w* option.

(r) *Net2pcap*: It is a simple tool to read packet traffic from an interface and to transform into a pcap file. Net2pcap is a Linux tool which does not use any library during the transformation. However, it is partially dependent on *libc*, a Linux library utility. The command *%tcpdump -w capfile* almost does the same task as Net2pcap. However, Net2pcap is usually used to capture and represent network traffic in a hostile environment to support subsequent analysis.

(s) *Ngrep*: Ngrep provides filtering facility on packets payloads. It also supports sniffing with the help of tcpdump and libpcap.

(t) *Snoop*: It is a Linux-based tool that works like *tcpdump*. However, the format of a snoop file is different from the *pcap* format and is defined in RFC 1761. An important feature of this tool is that when writing to an intermediate file, it reduces the possibility of packet loss under busy trace conditions. Snoop allows one to filter, read, as well as to interpret packet data. To observe the traffic between two systems, say *X* and *Y*, the following command is used to execute the tool.
% snoop *X*, *Y*

(u) *ScoopLM*: This is a Windows 2000-based sniffing tool for capturing LM/NTLM authentication information on the network. Such information can later be used by a tool such as *BeatLM* to crack authentication data.

(v) *Tcpdump*: Tcpdump is a premier packet analyzer for information security professionals. It enables one to capture, save, and view packet data. This tool works on most flavors of the Unix operating system. One can also use third-party open-source software, e.g., *wireshark* to open and visualize tcpdump captured traffic.

(w) *Tcptrace*: It is a powerful tool to analyze tcpdump files and to generate various types of outputs including connection-specific information, such as the number of bytes and segments sent and received, elapsed time, retransmissions, round trip times, window advertisements, and throughput. It accepts a wide range of input files generated by several capture tools such as tcpdump, snoop, etherpeek, HP Net Metrix, and WinDump. It also provides a graphical presentation of traffic characteristics for further analysis.

(x) *Tcptrack*: It can sniff and display TCP connection information, as seen in the network interface. Tcptrack has the following functions: (i) watch passively for connections on the network interface, (ii) keep track of their state, and (iii) display a list of connections. It displays source IP, destination IP, source port, destination port, connection state, idle time, and bandwidth usage.

We have seen that most discussed sniffing tools are not equally important for one's purposes all the time. Their usefulness and importance are measured based on user's requirements and purposes. One cannot use Cain & Abel for capturing live network traffic since it performs only password cracking. Most people use tcpdump and libpcap as network sniffing tools that can capture all the information of a packet

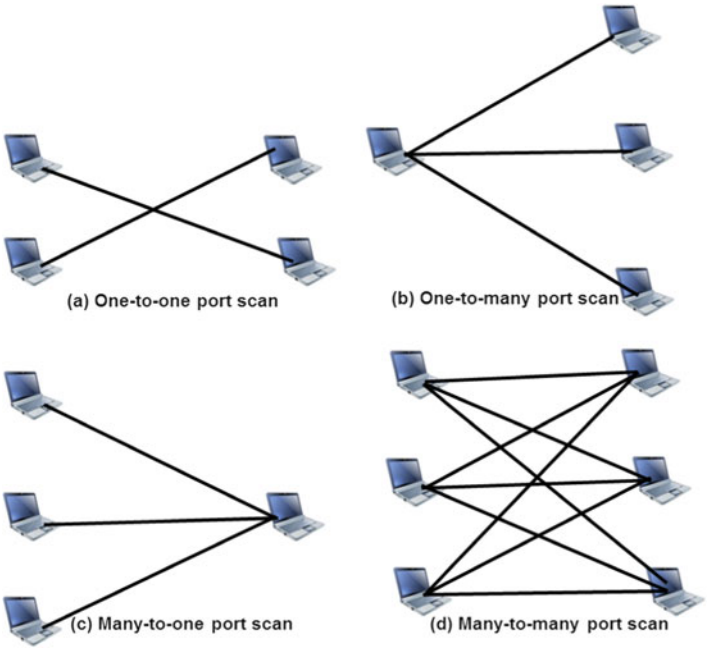


Fig. 6.3 Different types of port scans

and store them in a file. One can also use NFSEN and NFDUMP tools for NetFlow traffic capturing, whereas GULP is used for packet level traffic capturing. One can also use tcpdump as an implicit tool for packet as well as NetFlow capturing.

6.4.1.2 Network Scanning and Mapping Tools

A network scanning tool aims to identify active hosts on a network, either (i) to attack them or (ii) to assess vulnerabilities in the network. It provides an overall status report regarding network hosts, ports, IPs, etc. The four possible types of port scans are (i) one-to-one, (ii) one-to-many, (iii) many-to-one, and (iv) many-to-many as shown in Fig. 6.3. Below, we present a few network vulnerability scanning tools.

(a) *Amap*: Amap detects an application protocol without depending on the TCP/UDP ports it is bound to. It identifies applications running on a specific port by sending trigger packets, which are typically used for application protocol handshakes. Most network daemons only respond to the correct handshake (e.g., SSL). Amap considers the responses and looks for matches. This tool supports TCP and UDP protocols, regular and SSL-enabled ASCII, and binary protocols and has a wide list of options to control its behavior. It accepts an nmap machine readable output file and logs to a file and screen.

(b) *DMitry*: DMitry (Deepmagic Information Gathering Tool) is a command-line network scanning tool hard coded in C and works on UNIX or Linux platform. It performs well in scanning simple networks.
(c) *Ike-scan*: This tool assists in discovery, fingerprinting, and testing of IPSec VPN servers based on the IKE protocol. Ike-scan works on Linux, Unix, Mac OS, and Windows environment under the GPL license.
(d) *Nmap*: This network mapping tool facilitates network exploration and security auditing. It can scan large networks fast, especially against single hosts. It is effective in using raw IP packets to identify a large number of useful parameters, such as available hosts, services offered by the hosts, OSs running, and the use of packet filters or firewalls. In addition to its use in security audits, network administrators can use it for routine tasks such as maintaining network inventory, manage service upgrade schedules, and monitoring host or service uptime.
(e) *Paketto*: It is a set of tools to assist in manipulating TCP/IP networks based on nontraditional strategies. These tools provide tapping functionality within the existing infrastructure and also extend protocols beyond their original intention. Example tools include (i) *scanrand*, which facilitates fast discovery of network services and topologies; (ii) *minewt*, which serves as a user space NAT/MAT router; (iii) *linkcat*, which offers a Ethernet link to stdio; (iv) *paratrace*, which helps trace network paths without spawning new connections; and (v) *phentropy*, which uses *OpenQVIS* to render arbitrary amounts of entropy from data sources in 3-D phase space.
(f) *rnmap*: rnmap (Remote Network Mapper) is a scanning tool that works in client-server mode. Each client is usually connected to the centralized server and performs the scanning operations on a particular host or a system. It uses nmap for actual scanning operations.
(g) *Ttlscan*: This tool uses *libnet* and *libpcap* utilities to identify a host by sending TCP SYN packets to each port of the host. It sniffs the response from the host and uses it to identify hosts with services by forwarding packets to another host behind a firewall. It can detect the OS and its version running on a host behind the firewall by reading specific header parameters such as TTL, window size, and IP ID.
(h) *Unicornscan*: This is an asynchronous scanner as well as a payload sender. This scalable and flexible tool gathers and collects information quickly. For fast response, it uses a distributed TCP/IP stack and provides a user-friendly interface to introduce a stimulus into a TCP/IP-enabled device or network and measuring the response. The main features of this tool include asynchronous protocol-specific UDP scanning, asynchronous stateless TCP scanning with wide variations in TCP flags, and asynchronous stateless TCP banner grabbing.
(i) *Vmap*: This version mapper tool allows one to identify the version of a daemon by fingerprinting its characteristics, based on its responses to bogus commands.

Table 6.2 Comparison of scanning tools

Tool's name	Input	Protocol	Purpose	Effectiveness	Sources
Nmap	IP/p	C/U	Scanning	Very powerful, easily accessible	www.insecure.org
Rnmap	IP/p	C/U	Scanning	Very powerful, easily accessible	www.rnmap.sourceforge.net/
Amap	IP/p	C/U	Scanning	Powerful application mapper	www.freeworld.thc.org
Vmap	T	C/U	Version mapping	Few options, easy to use	www.tools.l0t3k.net
Unicornscan	T/p	C/U	Scanning	Supports more than 15 options	www.unicornscan.org
Ttlscan	T/p	C	Scanning	Linux based	www.freebsd.org
Ike-scan	T	C/U	Host discovery	Supports more than 50 options	www.stearns.org
Paketto	IN	C	Scanning	Very fast scanner	www.packages.com
DMitry	CL	C	Scanning	Simple scanner	www.mor-pah.net
ZMap	IN	C/UDP	Scanning	Fast scanner	www.github.com/zmap

IP IP address(es), *T* Target IP, *p* port, *IN* Interface ID, *C* TCP, *U* UDP, *CL* Command-line

(j) *ZMap*: ZMap is a fast network scanning tool designed for Internet-wide network surveys. It perform scanning of the entire public IPv4 address space within 45 min. If the connection is 10gigE speed and PF_RING, then ZMap performs the scan operations of IPv4 address space in under 5 min. It works on GNU/Linux, Mac OS, and BSD.

Table 6.2 shows the platforms on which these tools work and the sources from where they can be obtained. Almost all these tools are Linux based.

For scanning a large network, one can use nmap as the most effective tool. Nmap has the ability to scan a large network to determine multiple parameters such as active hosts and ports, operating systems, protocols, timing and performance, firewall/IDS evaluation and spoofing, and IPv6 scanning. Because of its multiple functionalities, network administrators find it very useful when monitoring a large network. Amap and Vmap do not support most of the functionalities performed by nmap. Attackers use nmap to find the vulnerabilities of a host to compromise it for constructing botnet during DDoS attack generation using the agent-handler architecture.

6.4.2 Attack Generation Tools

A large number of network security tools that use cryptographic mechanisms for attack launching are available on the Web. People can freely download these tools and can use them for malicious activities such as Trojan propagation, network

mapping, probe attacks, buffer overflow attacks, DoS/DDoS attacks, and application layer attacks. Such tools can be used to launch layer-specific and protocol-specific attacks, such as HTTP-, SMTP-, FTP-, or SNMP-related attacks. Other tools can be used to launch DoS/DDoS attacks, which can disrupt the services of a network or a Website very quickly. Some tools are used in wired networks to capture and exploit valuable information, while others are used in wireless networks.

6.4.2.1 Trojans

Trojans are malicious executable programs developed to break the security system of a computer or a network. A Trojan resides in a system as a benign program. Once the user attempts to open the file, the Trojan is executed, and some dangerous action is performed. Victims generally unknowingly download the Trojan from multiple sources such as from (i) the Internet, (ii) an FTP archive, (iii) via peer-to-peer file exchange using IRC, and (iv) Internet messaging. Typically, Trojans are of seven distinct types. We introduce each type with example.

T_1. *Remote access Trojans*: These are malware programs that use backdoors to control the target machine with administrative privilege. Remote access Trojans are downloaded invisibly with a user request for a program such as a game or an email attachment. Once the attacker compromises a machine, the Trojan uses this machine to compromise more machines to construct a botnet for launching a DoS or DDoS attack. An example of remote access Trojan is *danger*.

T_2. *Sending Trojans*: This Trojan type aims to capture and provide sensitive information such as passwords, credit card information, log files, email addresses, and IM contact lists to the attacker. In order to collect such information, such Trojans attempt to install a keylogger to capture and transmit all recorded keystrokes to the attacker. Examples of this type of Trojans are, namely, *Badtrans.B email virus* and *Eblast*.

T_3. *Destructive Trojans*: This Trojan type is very destructive for a computer and is often programmed to delete automatically some essential executable programs, configuration, and dynamic link library (DLL) files. Such a Trojan acts either (i) as per the instructions of a back-end server or (ii) based on pre-installed or programmed instructions, to strike on a specific day, at a specific time. Two common examples of this type are *Bugbear virus* and *Goner worm*.

T_4. *Proxy Trojans*: This Trojan type attempts to use a victim's computer as a proxy server. A Trojan of this kind compromises a computer and attempts to perform malicious activities such as fraudulent credit card transactions and launching of malicious attacks against other networks. Examples of proxy Trojans are *TrojanProxy:Win32* and *Paramo.F*.

T_5. *FTP Trojans*: This type of Trojan attempts to open port 21 and establish a connection from the victim computer to the attacker using the File Transfer Protocol (FTP). An example of FTP Trojan is *FTP99cmp*.

T_6. *Security software disable Trojans*: Such Trojans attempt to destroy or to thwart defense mechanisms or protection programs such as antivirus programs or firewalls. Often such a Trojan is combined with another type of Trojan as a payload. Some examples are *Trojan.Win32.KillAV.ctp* and *Trojan.Win32.Disable.b*.

T_7. *DoS Trojans*: Such Trojans attempt to flood a network instantly with useless traffic, so that it cannot provide any service. Some examples of this category of Trojan are *Ping of Death* and *teardrop*.

6.4.2.2 DoS/DDoS Attacks

Denial of service (DoS) is a commonly found yet serious class of attacks caused by an explicit attempt by an attacker to prevent or block legitimate users of a service from using desired resources. Such an attack occurs in both distributed as well as in a centralized setting. Some common examples of this class of attacks are: *SYN flooding, smurf, fraggle, jolt, land, and Ping of Death*.

A *distributed denial of service* (DDoS) attack is a coordinated attempt on the availability of services of a victim system or a group of systems or on network resources, launched indirectly from a large number of compromised machines on the Internet. Typically, a DDoS attacker adopts an *m*:1 scheme, i.e., many compromised machines are used to attack a single victim machine or an *m*:*n* approach that makes it very difficult to detect or prevent. A DDoS attacker normally initiates such a coordinated attack using either an architecture based on agent handlers or Internet relay chat (IRC). The attacking hosts are usually personal computers with broadband connections to the Internet. These computers are compromised by viruses or Trojan programs called *bots*. The compromised computers are usually referred to as *zombies*. The actions of these zombies are controlled by remote perpetrators often through (i) *Botnet* commands and (ii) a control channel such as IRC. Generally, a DDoS attack can be launched using any one of the following ways.

(i) *By degree of automation*: The attack generation steps such as recruit, exploit, infect, and use phase can be performed in three possible ways: manual, automatic, and semiautomatic.

(ii) *By exploited vulnerability*: The attacker exploits the vulnerability of a security system to deny the services provided by the system. In semantic attacks, it exploits a specific feature or implementation bug of some protocols or applications installed on the victim machine to overload the resources used by that machine. Example of such attack is TCP SYN attack.

(iii) *By attack network used*: To launch a DDoS attack, an attacker may use either an agent-handler network or an IRC network.

(iv) *By attack rate dynamics*: Depending on the number of agents used to generate a DDoS attack, the attack rate may be either constant rate or variable rate. Besides these, increasing rate attack and fluctuating rate attack can also be found based on rate change mechanism.

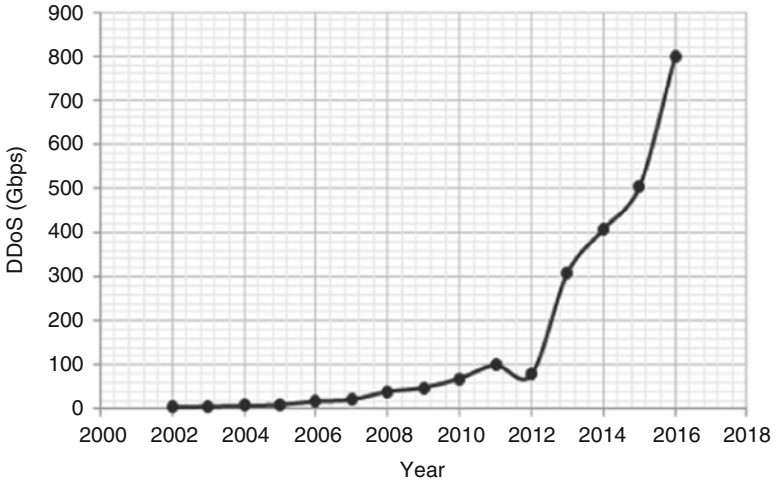


Fig. 6.4 DDoS attack statistics [26]

(v) *By victim type*: DDoS attacks can be generated based on four different victims, such as application attack, host attack, network attack, and infrastructure attack.
(vi) *By impact*: The impact of a DDoS attack may be either disruptive or degrading.
(vii) *By agent*: A DDoS attack can be generated by a constant agent set or a variable agent set.

Some statistics on DDoS attacks are shown in Fig. 6.4. Out of several DoS and DDoS attack generation tools, a few tools are explained below (Table 6.3).

(a) *Burbonic*: This DoS exploit attempts to victimize a Windows 2000 machine by sending randomly a large number of TCP packets with random settings with the purpose of increasing the load on the machines so that it leads to a crash.
(b) *Blast20*: This TCP service stress tool is able to identify potential weaknesses in the network servers quickly. An example use of this tool is shown below.
% blast targetIP port start_ size end_size /b (i.e. begin text) "GET/SOME TEXT" /e (i.e. end text) "URL"
The command is used to send some attack packets of size minimum *start_size* bytes to maximum *end_size* bytes to a server address "http://" in the specified target IP.
(c) *BlackEnergy*: This Web-based DDoS attack tool, an HTTP-based botnet, uses an IRC-based command and control method.
(d) *Crazy Pinger*: It attempts to launch an attack by sending a large number of ICMP packets to a victim machine or to a large remote network.
(e) *FSMax*: It is a server stress testing tool. To test a server in evaluating buffer overflows that may be exploited during an attack, it accepts a text file as input and takes a server through a sequence of tests based on the input.

Table 6.3 Comparison of attacking tools

Tool's name	Input	Protocol	Purpose	Effectiveness	Sources
Jolt	T	IC	DoS	Uses 100% cpu time	www.flylib.com
Burbonic	T	C	DoS	Multi-platform, easily accessible	www.packetstormsecurity.org/
Targa	T	C/U/IC	DoS	Very efficient	www.packetstormsecurity.org/
Blas20	T	C	DoS	Multi-platform, perform quick damage of a system	
Crazy Pinger	V	IC	DoS	Multi-platform, easily accessible	www.softwaretopic.informer.com
UDPFlood	V	U	DoS	Windows based, less powerful	www.foundstone.com
FSMax	F		DoS	Windows based, efficient for server testing	www.brothersoft.com
Nemsey	T/p	C	DoS	Windows based	packetstormsecurity.org/
Panther	T/p	U	DoS	Easily accessible	www.bestspywarescanner.net
Land & LaTierra	T	C	DoS	Powerful for land attack	
Slowloris	T	HT	DoS	Powerful for HTTP attack	www.ha.ckers.org/slowloris/
Blackenergy	S	C/U/IC	DDoS	Simple and powerful for DDoS	www.airdemon.net
HOIC	T	HT	DDoS	Very effective for DDoS	www.rapidshare.com
Shaft	V	U/C/IC	DDoS	Multi-platform, high applicability	
Knight	V	C/U	DDoS	Less powerful	www.cert.org
Kaiten	V	U/C	DDoS	Windows based	www.mcafee.com
RefRef	T		DDoS	Effective for DDoS	www.hackingalert.net
Hgod	T/p	C/U/IC	DDoS	Easily accessible	www.flylib.com
LOIC	T/p	C/U/IC	DDoS	Very effective, powerful for flooding attack	www.sourceforge.net
Trinoo	T/p	U	DDoS	Multi-platform, easy to use	www.nanog.org
TFN	T	U/C/IC	DDoS	Multi-platform, effective for flooding attacks	www.codeforge.com
TFN2K	T	U/C/IC	DDoS	Simple and easy to execute	www.goitworld.com
Stachaldraht	T	C	DDoS	Multi-platform, supports more features	www.packetstormsecurity.org
Mstream	T	C	DDoS	Multi-platform and more primitive	www.ks.uiuc.edu
Trinity	T	C/U	DDoS	Very effective to compromise hosts	www.garykessler.net

T Target IP, *V* Victim IP, *C* TCP, *U* UDP, *IC* ICMP, *F* Input text file, *p* Port, *HT* HTTP, *S* Server IP

(f) *HOIC*: It is an HTTP-based DDoS tool that focuses on creation of high-speed multi-threads to generate HTTP flood traffic. It is able to flood simultaneously up to 256 Websites. The built-in scripting system in this tool allows the attacker to deploy boosters, which are scripts designed to thwart DDoS counter measures.

(g) *Hgod*: This is a Windows XP-based tool that spoofs source IPs and can specify protocol and port numbers for an attack. By default, it is used for TCP SYN flooding. An example of TCP SYN flooding attack command against 192.168.10.10 on port 80 with a spoofed address of 192.168.10.9 is shown below.
%hgod 192.168.10.10 80 -s 192.168.10

(h) *Jolt*: This DoS attack tool sends a large number of fragmented ICMP packets to a target machine running Windows 95 or NT in such a manner that the target machine fails to reassemble them for use, and as a result, it freezes up and cannot accept any input from the keyboard or mouse. However, this attack does not cause any significant damage to the victim system, and the machine can be recovered with a simple reboot.

(i) *Knight*: This IRC-based tool can launch multiple DDoS attacks for SYN flood, UDP flood, and urgent pointer flood on Windows machines.

(j) *Kaiten*: This is an IRC-based attack tool and is able to launch multiple attacks, viz., UDP flood, TCP flood, SYN flood, and PUSH+SYN flood attacks. It uses random source addresses.

(k) *LOIC*: It is an anonymous attacking tool via IRC. It operates in three modes based on the protocols TCP, UDP, and HTTP. It exists in two versions: binary and Web-based. It uses multiple threads to launch an attack.

(l) *Nemsey*: The presence of this tool implies that a computer is insecure and infected with malicious software. It attempts to launch an attack with attacker specified number of packets and sizes including information such as protocol and port.

(m) *Panther*: This UDP-based DoS attack tool can flood a specified IP address at a specified port instantly.

(n) *RefRef*: RefRef is used to exploit existing SQL injection vulnerabilities using features included in MySql such as SELECT permissions to create a DoS attacks on the associated SQL server. It sends malformed SQL queries carrying payloads which force servers to exhaust their own resources. It works with a Perl compiler to launch an attack.

(o) *Stacheldraht*: This DDoS attacking tool is a hybridization of TFN and Trinoo, with some additional features, such as encrypted transmission between the components and automatic updation of the daemons.

(p) *Slowloris*: It creates a large number of connections to a target victim Web server by sending partial requests and attempts to hold them open for a long duration. As a consequence, the victim servers maintain these connections as open, consuming their maximum concurrent connection pool, which eventually compels them to deny additional legitimate connection attempts from clients.

(q) *Shaft*: It is variant of Trinoo, which provides statistics on TCP, UDP, and ICMP flooding attacks. These help the attackers identify the victim machine's status (i.e., completely down or alive) or to decide termination of zombies in addition to the attack.
(r) *Targa*: Targa is a collection of 16 different DoS attack programs. One can launch these attacks individually as well as in a group and can damage a network instantly.
(s) *Trinoo*: This is an effective DDoS attack tool that uses a master host and several broadcast hosts. It issues commands using TCP connection to the master host, and the master instructs the broadcast hosts via UDP to flood at specific target IP at random ports with UDP packets. To launch an attack using this tool, an attacker should possess prior access to the host to install a TRINOO master or broadcast, either by passing or by compromising the existing security system.
(t) *TFN*: Like Trinoo, TFN is comprised of a client host and several daemon hosts. It is very effective in launching DDoS attacks, viz., ICMP flood, UDP flood, SYN flood, and smurf attacks. TFN2K, a variant of TFN, also includes some special features, such as encryption and decryption, stealth attacks, DoS attacks to crash a specified target host, and to communicate shell commands to the daemons.
(u) *UDPFlood*: This tool can flood a specific IP at a specific port instantly with UDP packets. The flooding rate, maximum duration, and maximum number of packets can be specified in this tool. It can also be used for testing the performance of a server.

A large number of attacking tools are available in the Internet, and most of them are powerful enough to crash networks and Websites. Among them, we found LOIC and HOIC are able to launch DDoS attacks within a very short time interval. LOIC supports TCP, UDP, and HTTP protocols to construct attack packets, whereas HOIC supports only the HTTP protocol. Though TFN, Trinoo, and Stacheldraht are used to launch DDoS attacks, they are not as powerful as LOIC. It should be noted that the use of LOIC to launch an attack in a public network is a crime.

6.4.2.3 Packet Forging Attack Tools

Packet forging tools are useful in forging or manipulating packet information. An attacker can generate traffic with manipulated IP addresses based on this category of tools. We describe some commonly found packet forging tools (Table 6.4).

(a) *Aicmpsend*: This ICMP packet sending tool supports several features including ICMP flooding and spoofing. It allows one to implement all the ICMP flags and codes.
(b) *Arp-sk*: ARP Swiss Knife (Arp-sk) allows one (i) to create totally arbitrary ARP requests, (ii) to manipulate ARP packets, and (iii) to test network security and connectivity.

Table 6.4 Comparison of packet forging attack tools

Tool's name	Input	Protocol	Purpose	Effectiveness	Sources
Packeth	S/D	U/C/IC	Packet generator	Supports many features	www.sourceforge.net
Packit	I	C/U/IC	Packet analysis and injection	Supports more than 60 options	www.packetfactory.openwall.ne
Packet excalibur				Multi-platform	www.freecode.com
Nemesis	H	IC/IG/C/U	Packet crafting/injection	multi-platform, powerful	www.sourceforge.net
Tcpinject	H	C	Packet generator	Linux based, easy accessible	www.packetstormsecurity.org
Libnet	H	C	Packet injection	Portable, efficient, and easy to use	www.packetfactory.net/libnet
SendIP	H	C/U	Packet generator	Supports so many options	www.softpedia.com
IPsocery	H	C/U/IC/IG	Packet generator	Easy to use	www.tools.l0t3k.net
Pacgen	H	C/U	Packet generator	Simple packet generator	www.sourceforge.net
Arp-sk	H	A	ARP packet generator	Sensitive for ARP attack	www.arp-sk.org
ARP-SPOOF	T	A	Packet intercept	Efficient for ARP cache poisoning	www.sourceforge.net
Libpal	H	C/U/IC	Packet intercept	User friendly	www.sourceforge.net
Aicmspend	T	IC	ICMP packet flooding	Supports many features	www.packetstormsecurity.nl

S Source IP, *D* Destination IP, *I* Interface ID, *H* Host IP, *T* Target IP, *C* TCP, *U* UDP, *IC* ICMP, *IG* IGMP, *A* ARP

(c) *Arpspoof*: This tool is also known as ARP cache poisoning. It allows one to spoof the contents of an ARP table on a remote computer on the LAN. Two addresses are used to establish connection between two computers on an IP/Ethernet network: (i) MAC address, which is used on a local area network before packets go out of the gateway, and (ii) IP address, which is used to surf the Internet through a gateway.

(d) *IPsorcery*: This TCP/IP packet generating tool has the ability to send TCP, UDP, and ICMP packets using the GTK+ interface.

(e) *Libpal*: This user-friendly packet assembly library provides utilities to build and send forged Ethernet, IP, ICMP, TCP, and UDP packets. It uses a *struct* to represent a packet.

(f) *Libpal*: This user-friendly packet assembly library provides utilities to build and send forged Ethernet, IP, ICMP, TCP, and UDP packets. It uses a *struct* to represent a packet.

(g) *Libnet*: This tool provides facilities to the application programmer including construction and injection of network packets through a portable and high-level API. To support underlying packet creation and injection functionality, it uses the *libnet* utility.

(h) *Nemesis*: This Unix-like and Windows-based network packet crafting and injection tool is useful for testing any NIDS, firewall, IP stack, and a variety of other tasks. This command-line driven tool also provides an option for scripting. Nemesis allows an attacker to craft and inject a large variety of packets. Especially in IP and the Ethernet injection modes, it allows one to craft and inject almost any custom packets.
(i) *Pacgen*: This Linux-based Ethernet IP TCP/UDP packet generating tool allows an attacker to generate custom packets with configurable Ethernet, IP, TCP, and UDP layers as well as custom payloads. It also includes additional features such as *packet count* and *programmable time interval* between packets sent.
(j) *Packit*: This network auditing tool allows one to customize, inject, monitor, and manipulate IP traffic. It is useful in various ways such as (i) to test a NIDS, (ii) to evaluate the performance of a firewall, (iii) to scan the network, and (iv) to simulate the network traffic and (v) in TCP/IP auditing.
(k) *Packeth*: Packeth is a Linux-based tool with graphical user interface. It can send any packet or sequence of packets using raw sockets on the Ethernet. It provides a large number of options such as incorrect checksum and wrong header length.
(l) *Packet excalibur*: This forging tool allows one to (i) sniff packets, (ii) build and receive custom packets, and (iii) to spoof packets. It has a graphical interface to build scripts as a text file and to define additional protocols.
(m) *Tcpinject*: This forging tool allows one to transmit a wide variety of TCP/IP packets by specifying multiple parameters, such as source IP, destination IP, source port, destination port, packet size, payload, TCP control flags, and TCP window size.
(n) *SendIP*: This command line forging tool allows one to send arbitrary IP packets with a large number of options to specify the content of every header of a specific packet. Any data can be added to the packet during transmission.

Based on the study of packet forging tools, we note that Nemesis is widely used to generate custom packets using different protocols. It supports most protocols such as ARP, DNS, ICMP, IGMP, IP, OSPF, RIP, TCP, and UDP, making it much more effective than other tools. Other advantages of this tool include the facts that (i) anyone can generate custom packets from the command prompt or shell script of a system and (ii) attackers find it very useful to generate attack packets.

6.4.2.4 Application Layer Attack Tools

In an application layer attack, the attacker uses legitimate application layer HTTP requests from legitimately connected network machines to overwhelm a Web server [27]. The application layer attack may be of different types such as session flooding attack, request flooding attack, or an asymmetric attack [21, 32]. Application layer DDoS attacks are more subtle than network layer attacks, and the detection of

application layer attacks is challenging since they use legitimate protocols and legitimate connections. Application layer attacks are of four types.

(a) *HTTP-related attacks*: In this attack, a massive amount of HTTP request is sent to overwhelm the target site in a very short time frame. Some commonly used tools of this category are Code Red Worm and its mutations, Nimda Worm and its mutations, Cross-site scripting attacks, Malicious URLs, and AppDDoS.
(b) *SMTP-related attacks*: The SMTP protocol is used to transmit emails over the Internet. The attacker tries to attack a mail server using this Internet mail transfer protocol. Some example attack tools of this category are SMTP mail flooding, SMTP worms and their mutations, extended relay attacks, and firewall traversal attacks.
(c) *FTP-related attacks*: The first step of this attack is to initiate a legitimate FTP connection and then send some attack packets to the victim. Examples include FTP bounce attacks, FTP port injection attacks, passive FTP attacks, and TCP segmentation attacks.
(d) *SNMP-related attacks*: The main goal of an SNMP attack is to change the configuration of a system and then monitor the state or availability of the system. Examples of this category of attacks include SNMP flooding attacks, default community attacks, and SNMP put attacks.

6.4.2.5 Fingerprinting Attack Tools

Fingerprinting tools are used to identify specific features of a network protocol implementation by analyzing its input and output behavior. The identified features include protocol version, vendor information, and configurable parameters. Fingerprinting tools are used to identify the operating system running on a remote machine and can also be used for other purposes. Existing fingerprinting tools are implemented for the key Internet protocols such as ICMP, TCP, TELNET, and HTTP [2, 24, 28]. Network administrators can use remote fingerprinting to collect information to facilitate management, and an intrusion detection system can capture the abnormal behavior of attackers or worms by analyzing their fingerprints [25].

(a) *AmapV4.8*: The Amap fingerprinting tool identifies applications and services by creating bogus communication without listening on default ports. It maintains a database of all the known applications, including non-ASCI-based applications and enterprise services.
(b) *CronOS*: This fingerprinting tool is used to determine the operating system of a target machine. This tool is embedded in Namp-CronOS, and it has three options to perform different operations. The *S* option guessed the time-out of SYN_RCVD states, the *I* option determines the last ACK state time-out, and the *f* option uses FIN_WAIT_1 state time-out for fingerprinting.
(c) *Disco*: The Disco fingerprinting tool is used to discover unique IP addresses on a network. In addition to IP discovery, it also fingerprints TCP SYN packets.

(d) *Nmap*: Nmap is one of the best fingerprinting tools for both Unix and Windows operating systems. It is very useful in network mapping as well as for information gathering from a remote machine on a network as described in Sect. 6.4.1.1.
(e) *P0f*: P0f is a passive OS fingerprinting tool that uses active fingerprinting performed by Nmap. The only difference is that passive fingerprinting simply sniffs the network and classifies the host based on the observed traffic. This is more difficult than active fingerprinting, since one has to accept whatever communication happens rather than designing custom probes.
(f) *Queso*: This utility runs on Linux and Solaris operating systems. It is used to remotely determine the operating system's version and manufacture's information by analyzing network packets. It provides precise information about a network or a system by scanning the network.
(g) *Sprint*: This fingerprinting tool is used to identify the operating system running on a machine. In addition, sprint also has the ability to calculate up times and contains advanced banner grepping functionality. Sprint, when run with -n switch, simulates netcraft.
(h) *Xprobe*: This OS fingerprinting tool is used to find the operating system run by a remote machine. Xprobe is as simple as Nmap, and it exploits the ICMP protocol in its fingerprinting approach.

Among the above discussed fingerprinting tools, Nmap is the best due to its multiple functionalities and greater effectiveness compared to others. P0f supports passive scanning, whereas Xprobe and Queso[1] are used for remote operations. Nmap provides detailed information on a network or a host with the maximum amount of vulnerability information. Table 6.5 shows the platforms used as sources for these tools.

6.4.2.6 User Attack Tools

In user attacks [17], the attacker either (i) attempts as a normal legitimate user to gain the privileges of a root or superuser or (ii) attempts to access a local machine by exploiting its vulnerabilities without having an account on that machine. Both types of attempts are very difficult to detect because their behavior resembles normal characteristics. We discuss these attacks by category along with launching tools for these attacks.

(a) *U2R attack*: In this attack, as shown in Fig. 6.5, the attacker initially attempts to gain access to the local victim machine as a legitimate user by some means. The means may be a password sniffing attempt, dictionary attack, or any social engineering approach. The attacker then explores the possible vulnerabilities or bugs associated with the operating system running in the victim machine

[1] http://spot-act.heck.in/queso-scanner-v-0-5.xhtml

Table 6.5 Comparison of fingerprinting tools

Tool's name	Input	Protocol	Purpose	Effectiveness	Sources
Nmap	H/N	C/U	Network scanning	Easily accessible, powerful, widely used	www.nmap.org
P0f	H	C	Remote network identification	Passive fingerprinting, difficult to use	www.lcamtuf.coredump.cx
Xprobe	T	C/U	Port scanning	Simple as nmap, powerful	www.sourceforge.net
CronOS	T	C	OS identification	Linux based	pen-testing.sans.org
Queso	H	C/U	OS fingerprinting	Multi-platform, widely used	www.tools.l0t3k.net
AmapV4.8	T/p	C/U	Host and port scanning	Powerful application mapper	www.linux.softpedia.com
Disco		C	IP discovery and Fingerprinting	Support large number of features	www.tools.l0t3k.net
Sprint	H	C	OS identification	Simple and efficient	www.safemode.org

H Host IP, *N* Network IP, *T* Target IP, *C* TCP, *U* UDP, *p* Port

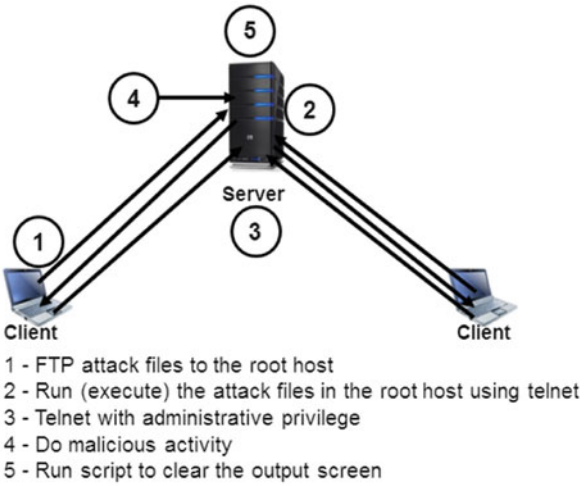


Fig. 6.5 Steps in U2R attack

to perform the transition from user to superuser or root level. Once the root privileges are acquired, the attacker possesses full control of the victim machine to install backdoor entries for future exploits, manipulate system files to gather information, and other damaging actions. Two well-known U2R attack tools are described next.

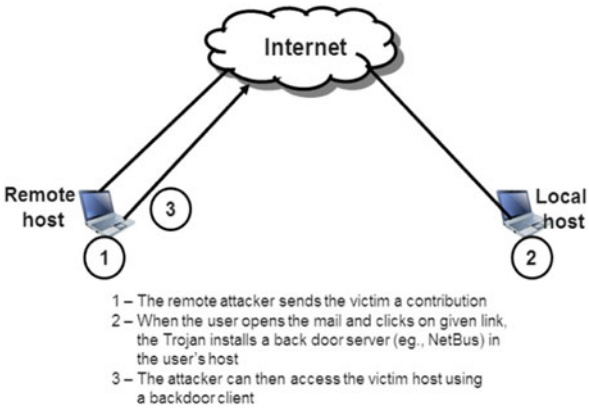


Fig. 6.6 Steps in R2L attack

- *Sqlattack*: Here, the attacker creates a TCP connection with the SQL database server on a Unix machine. The database shell exits when a special escape sequence is issued and the root shell of the machine is started by running the Perlmagic[2] script.
- *Yaga*: This tool is used to create a new administrator account by compromising registry files. The attacker edits the registry file to crash some system services on the victim machine and create a new administrator account.

(b) *R2L attack*: In this attack, a remote attacker, without having an account on a local machine, attempts to send packets to that machine by gaining local access, based on the vulnerabilities of that machine. To gain access to the local machine, the attacker attempts various ways as shown in Fig. 6.6. Two such ways are (i) using online and offline dictionary attacks to acquire the password to access the machine and (ii) making repeated guesses at possible usernames and passwords. The attacker also attempts to take the advantage of those legitimate users who are often casual in choosing their passwords. Below are two R2L attack tools:

- *Netcat*: This R2L attack tool uses a Trojan program to install and run Netcat on the victim machine on port number 53. The Netcat program works as a backdoor to access the machine using Netcat port without any username and password.
- *ntfsdos*: Here, the attacker gains the console of a WinNT machine by running ntfsdos. The program mounts the machine's disk drives. Thus the attacker can copy secret files on the secondary media.

[2] http://www.perlmagic.org/

Table 6.6 Comparison of other tools

Tool's name	Input	Protocol	Purpose	Effectiveness	Sources
Ping	H	IC	Host discovery	Commonly used, easily accessible	www.download.cnet.com
Hping2	T	IC/C/U	Port scanning	Supports so many features	www.hping.org.
Hping3	T	IC/C/U	Port scanning	Powerful for network testing	www.hping.org.
Traceroute	H	IC/C/U	Route discovery	Multi-platform, easily accessible	www.brothersoft.com
Tctrace	H	C	Route discovery	Multi-platform, easily accessible	www.tcptrace.org/
Tcptrace route	I	C	DNS lookup	Powerful for route discovery	www.michael.toren.net/
Traceproto	H	C/U/IC	Route discovery	Effective for firewall testing	www.traceproto.sourceforge.net
Fping	T	IC	Target host discovery	More powerful than ping	www.softpedia.com
Arping	S	A	Send ARP request	Linux based, efficient	www.linux.softpedia.com

H Host IP, *T* Target IP, *I* Interface ID, *S* Source IP, *C* TCP, *U* UDP, *IC* ICMP, *A* ARP

6.4.2.7 Other Attack Tools

In addition to the tools reported in the preceding sections, there are other tools that have direct or indirect use in the attack launching process. We discuss some of them to increase the awareness of learners and security researchers (Table 6.6).

(a) *Arping*: The arping tool is used in the Linux platform to send ARP request messages to a destination host in a LAN. It is used to test whether an IP address is in use or not.
(b) *Fping*: Fping uses ICMP protocol to determine whether a host is active or not. Fping is more powerful than ping because it can scan any number of hosts or a file containing the list of hosts. Instead of trying one host until it times out or replies, fping sends out a ping packet and moves to the next host in a round-robin fashion. If a host replies, it is noted and removed from the list of hosts to check. If a host does not respond within a certain time limit and/or retry limit, it is considered unreachable. Unlike ping, fping is meant to be used in scripts, and its output is easy to parse.
(c) *Hping2*: It is used to send custom TCP/IP packets and display reply messages received from the target. It handles fragmentation and arbitrary packet size and can be used to transfer files. It performs testing of firewall rules, port scanning, protocol-based network performance testing, and path MTU discovery.
(d) *Hping3*: This tool works almost like Hping2 and can handle fragmentation with arbitrary packet sizes. It finds the sequence number for reply packets from the

source port. It starts with a base source port number and increases this number when each packet is sent. The default base source port is random. The source port number may be kept constant for each sent packet.

(e) *Ping*: This tool is used to test network connectivity or reachability of a host on an IP network. Ping is a pioneering tool developed to check the connectivity of a computer, router, or the Internet. The ping request is sent to a particular host or to a network using command prompt. As a reply, it displays the response of the destination host and how long it takes to receive a reply. It uses the ICMP protocol, which has low priority and slower speed than the regular network traffic.

(f) *Traceroute*: Traceroute is used to show the route between two systems in a network. It also lists all intermediate routers from the source end to destination end. Using this tool, one determines how systems are connected to each other or how an Internet service provider connects to the Internet to provide services. The traceroute program is available on most computers including most Unix systems, Mac OS, and Windows OS.

(g) *Tctrace*: Though Tctrace is similar to traceroute, it uses TCP SYN packets to trace. This makes it possible for one to trace through firewalls if one knows a TCP service that is allowed to pass from the outside.

(h) *Tcptraceroute*: Tcptraceroute sends either UDP or ICMP ECHO request packets using a TTL field, which is incremented by one with each hop until the destination is reached. It shows the path that a packet has traversed to reach the destination. However, the widespread use of firewalls filters tcptraceroute packets as a result of which, it may not be able to complete the path to the destination.

(i) *Traceproto*: Traceproto is almost similar to traceroute, but this tool allows the user to choose protocols to be traced. It currently allows TCP, UDP, and ICMP protocol trace. It can be used to test and bypass firewalls, packet filters, and check if ports are open. Traceproto is actually a traceroute replacement tool written in C.

6.5 Tools for Defenders

Defenders use both categories of tools because it is usually required to evaluate any network traffic anomaly detection technique or system in a realistic environment. Most commonly found and useful defender tools are listed below.

6.5.1 Network Traffic Monitoring and Visualization Tools

Monitoring of network traffic is an essential activity for network defenders to observe, analyze, and finally to identify any anomalies occurring in the network. To support such activities of the network defenders as well as to assist in the

meaningful interpretation of the outcomes of their analysis, network monitoring and analysis tools have been playing an important role [15, 16]. Rapid incidences of malicious attempts to compromise the confidentiality, integrity, and access control mechanisms of a system or to prevent legitimate users of a service from accessing the requested resources have led to an increased demand for developing useful tools to visualize network traffic in a meaningful manner to support subsequent analysis. We present some of these tools under two distinct categories, *visualization* [12, 31] and *analysis*.

An effective network traffic (both packet and NetFlow traffic) visualization tool can be of significant help for the network defenders in monitoring and analysis tasks. Appropriate visualization not only supports meaningful interpretation of the analysis results but also assists the security managers in identifying anomalous patterns. It also helps in taking appropriate action to mitigate attacks before they propagate and infect other parts of the network. Some visualization tools are discussed below.

(a) *EtherApe*: EtherApe allows to sniff live packets and to monitor captured data in the Unix environment.
(b) *NetViewer*: This tool not only supports observing the captured live traffic in an aggregate way but also helps in identifying network anomalies. In addition, NetViewer supports visualization of useful traffic characteristics in tuning of defense mechanisms.
(c) *NetGrok*: It is a real-time network visualization tool that aids in presenting a graphical layout and a tree map to support visual organization of the network data. It can visualize live packets, capture trace, and helps in filtering activities.
(d) *Network traffic monitor*: This tool can present and scan detailed traffic scenarios since the inception of an application process and also allows analyzing traffic details.
(e) *Rumint*: This tool enables out to visualize live captured traffic as well as save it as pcap file in the Windows environment.
(f) *Tnv*: This time-based traffic visualization tool presents packet details and links among local and remote hosts. It assists in learning the normal patterns in a network, investigating packet details, and network troubleshooting. Tnv provides multiple services to support inspection and analysis activities: (i) opening and reading libpcap files, (ii) capturing live packets, and (iii) saving captured data in a MYSQL database.
(g) *VizNet*: It helps visualize the performance of a network based on bandwidth utilization.

Most of the visualization tools discussed above support both visualization and analysis of network traffic (Table 6.7).

To visualize a network, one can use EtherApe in Unix or NetViewer in Windows platform. For real-time visualization of live traffic for intrusion detection, NetViewer is the best due to its ability to detect anomalous network traffic. Network defenders need real-time visualization tools that can detect abnormal behaviors of network traffic and immediately generate alert messages to inform the administrator.

Table 6.7 Comparison of visualization tools

Tool's name	Input	Protocol	Purpose	Effectiveness	Sources
Tnv	F	C/U/ IC	Traffic visualization	Supported by all OSs	www.tnv.sourceforge.net
Network Traffic Monitor 1.02	H	C/U/IC	Live traffic monitoring	Easy accessible, efficient for visualization	www.monitor-network-traffic.winsite.com
Rumint	H	C/U/IC/IG	Visualize life traffic	Extremely flexible	www.rumint.org
EtherApe	H	C	Traffic flow visualization	Very simple and powerful	www.brothersoft.com
Netgrok	H	C/U/IC	Real-time traffic visualization	Multi-platform and easy accessible	www.softpedia.com
Netviewer	H	C/U	Traffic analysis	Powerful defense tool	www.brothersoft.com
VizNet	H	C/U	Traffic analysis and visualization	Efficient for visualization	www.viznet.ac.uk

H Host IP, *F* Captured data file, *C* TCP, *U* UDP, *IC* ICMP, *IG* IGMP

6.5.2 Network Traffic Analysis Tools

Network traffic analysis tools are used to capture, preprocess, and analyze traffic to detect anomalies. To capture live network traffic from a high-speed network, a packet or flow snipping tool is needed. Some network traffic analysis tools are discussed below (Table 6.8).

(a) *w3af*: It is a powerful and flexible tool for finding and exploiting web application vulnerabilities. w3af is like metasploit, but it is for web-based applications.
(b) *prtg*: PRTG is a network traffic monitoring and analysis tool to ensure services, device accessibility, and applications. It can perform scanning of all IT infrastructures.
(c) *Nagios*: Nagios is a powerful, scalable, and flexible tool for monitoring and analysis of network traffic to identify network outages and protocol failures.
(d) *OpenNMS*: OpenNMS is a highly integrated open-source tool for monitoring and analyzing network traffic to report any network performance and service problems. It ensures appropriate services and availability and accessibility of resources in time.
(e) *Capsa*: Capsa is an extremely powerful, comprehensive, and portable packet capturing and analysis tool to assess live network traffic. It generates reports for further action. It has a number of features including (i) real-time packet capture, (ii) multiple network behavior monitoring, (iii) advanced protocol analysis, and (iv) extensive statistics for each host. It provides a high-level window view of the entire network.

Table 6.8 Comparison of network traffic analysis tools

Tool's name	Input	Protocol	Purpose	Effectiveness	Sources
w3af	F/IL	HT	Finding web application vulnerabilities	Powerful, flexible, and easy to use	www.w3af.org
prtg	F/IL	HT/D/S	Monitor and analyze traffic	Flexible and easy to use	www.paessler.com/prtg
Nagios	F/IL	C/S	Monitor and analyze traffic	Flexible and scalable	www.nagios.com
OpenNMS	F/IL	HT/S	Network traffic monitoring and analysis	Flexible and extensible architecture	www.opennms.org
Capsa	F/IL	C/U/IC /HT/D/S	Monitor and analyze traffic	Portable and comprehensive	www.colasoft.com/capsa
Splunk	F/IL	C/U/S	Secure monitoring and analysis of traffic	Can identify external and internal threats	www.splunk.com
NetXMS	F/IL	S/HTTP	Monitor and analyze traffic	Multi-platform and easy to use	www.netxms.org

H Host IP, *F* Captured data file, *C* TCP, *U* UDP, *IC* ICMP, *IG* IGMP, *HT* HTTP, HTTPS, *D* DNS, *IL* Interface, *CL* Command-line, *S* SNMP

(f) *NetXMS*: NetXMS is an open-source multi-platform network management and monitoring system. It provides event management, alert management, performance management, report generation, and graph generation for all layers.
(g) *Splunk*: Splunk is a leading tool for operational intelligence. It facilitates traffic monitoring, visualization, and analysis to identify the threats to availability. It works in both Unix and Linux platforms.

6.6 Approach to Develop a Real-Time Network Traffic Monitoring and Analysis Tool

We believe that an interested technical reader's understanding will be enhanced if he/she is able to build a practical system himself/herself.

A denial of dervice (DoS) attack attempts to make machines or network's resources unavailable to its intended users either temporarily or indefinitely, interrupting or suspending services of a host connected to the Internet. Normally, DoS attacks are generated by a single host or a small number of hosts at the same location. On the other hand, a DDoS attack is a combination of DoS attacks where attacks are generated by a large number of hosts. These hosts may be amplifiers or reflectors or even may be zombies. They usually send the traffic to the target or victim host through the reflectors. For examples, DDoS attacks in 2000 to well-known Websites such as CNN, Amazon, and Yahoo stopped normal services of these victims for hours [5, 10].

Most existing network defense techniques and tools still heavily rely on actual security analysts (SA). These techniques involve SAs to analyze and detect network attacks manually. To enhance the human perception and understanding of all kinds of network attacks, network traffic visualization has become important in recent years to speed up the attack detection process.

We design a tool for information gathering, generation of attack traffic, capturing live traffic, and monitoring and analysis, known as KUD-Vis to generate and detect DDoS attacks. Our tool has the following characteristics: (a) information gathering using rnmap, (b) generation of attack traffic using Targa in a distributed mode, (c) packet capture using jpcap library, (d) examination of all packets at the monitoring host (in promiscuous mode), (e) use of memory efficient data structures, (f) generation of statistical summaries that can be retained for further analysis, and (g) generation of an undirected graph to visualize the network traffic (Table 6.9).

6.6.1 KUD-Vis: Information Gathering

To discover vulnerabilities in the network, we use the KUD-vis tool. It executes rnmap (remote nmap) [22], for both client and server. Any client can connect to the server for initiating scan operation to the target host or network. A remote nmap client is shown in Fig. 6.7.

The output of remote nmap is captured and used in the next steps.

6.6.2 KUD-Vis: Attack Traffic Generation

Based on the traffic vulnerabilities found in a host or network, KUD-vis initiates an attack to the target. KUD-vis analyzes the vulnerabilities gained in the previous step and decides immediately to initiate an attack to the target. KUD-vis exploits features of the system vulnerabilities when it generates attack traffic.

6.6.3 KUD-Vis: Capturing Traffic

We configure our network to redirect traffic in such a way that all traffic is forwarded to a particular port. As a result, KUD-Vis can capture each traffic record and visualize records for detection of DDoS attacks. KUD-Vis uses the jNetPcap [14] library for capturing and processing traffic. After capturing network traffic, it filters out the IP packets for subsequent consideration. It uses routers to extract various types of features from the IP packets and finally constructs a 5 min traffic feature sample. Each such sample is further divided into subsamples of 5 s. These samples are formatted for visualization. Due to lightweight nature of the system, KUD-Vis can visualize the traffic quickly to support attack detection.

Table 6.9 Category-wise tools information

Category	Tool name	Source
Trojans	NukeNabbler	http://community.norton.com
	AIMSpy	http://www.securitystronghold.com
	NetSpy	http://www.netspy-trojan-horse.downloads
Information gathering tools	ASS	http://www.manpages.ubuntu.com
	NMap	http://www.nmap.org
	p0f	http://www.lcamtuf.coredump.cx/p0f.shtml
	MingSweeper	http://www.hoobie.net/mingsweeper
	THC Amap	http://www.freeworld.thc.org/thc-amap
	Angry IP Scanner	http://www.angryziber.com/w/Download
DoS attack tools	Targa	http://www.security-science.com/
	Burbonic	http://www.softpedia.com
	Blast20	http://seomagz.com/2010/03/dos-denial-of-service-attack-tools-ethical-hacking-session-3/
Spoofing attack tools	Engage Packet Builder	http://www.engage-packet-builder.software.informer.com/
	Hping	http://www.hping.org
	Nemesis	http://www.nemesis.sourceforge.net
	PacketExcalibur	http://www.linux.softpedia.com
	Scapy	http://www.softpedia.com
TCP session hijacking tools	Firesheep	http://www.codebutler.github.com/firesheep/
	Hunt	http://www.packetstormsecurity.org/sniffers/hunt
	Juggernaut	http://www.tools.l0t3k.net/Hijacking/1.2.tar.gz
	TTY Watcher	http://www.security-science.com
	IP Watcher	http://www.download.cnet.com
	Hjksuit-v0.1.99	http://www.tools.l0t3k.net/Hijacking/hjksuite-0.1.99.tar.gz
Probe attack tools	Solarwind	http://www.solarwinds.com
	Network Probe	http://www.softpedia.com
	NMap	http://nmap.org
Spoofing attack tools in wireless	Kismet	http://www.linux.die.ne
	libpcap	http://www.sourceforge.net/projects/libpcap
	libnet	http://www.libnet.sourceforge.net
	libdnet	http://www.libdnet.sourceforge.net/
	libradiate	http://www.packetfactory.net/projects/libradiate
Application layer attack tools	HOIC	https://www.rapidshare.com
	LOIC	http://www.softpedia.com
	RefRef	http://www.softpedia.com

(continued)

Table 6.9 (continued)

Category	Tool name	Source
Network monitoring tools	w3af	http://w3af.org/
	PRTG	https://www.paessler.com/prtg
	Nagios	https://www.nagios.org/
	OpenNMS	https://www.opennms.org/en
	Capsa	http://www.colasoft.com/capsa/
	The Dude	https://mikrotik.com/thedude
	Splunk	https://www.splunk.com/
	NetXMS	https://www.netxms.org/

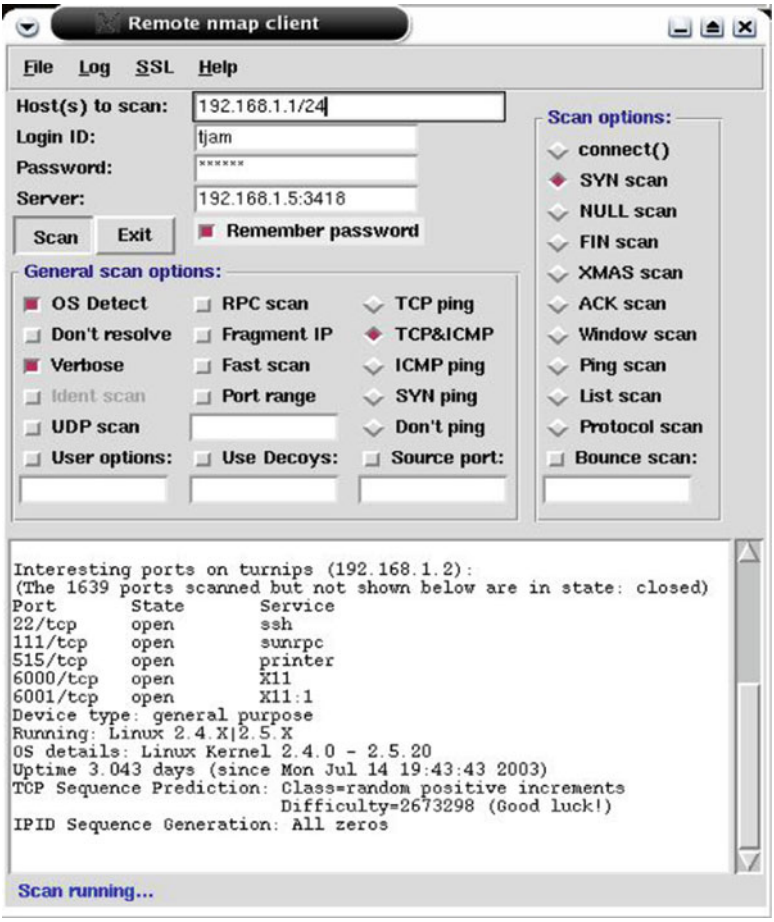


Fig. 6.7 Remote nmap [22]

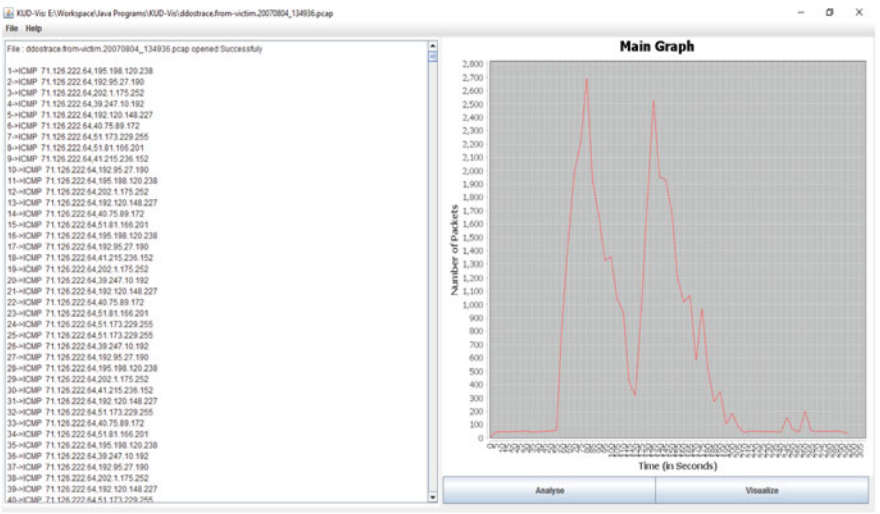


Fig. 6.8 KUD-Vis: working in offline mode using CAIDA DDoS 2007 dataset

6.6.4 KUD-Vis: Monitoring and Analysis

For analysis of captured traffic, we collect subsamples of interval 5 s for early visualization of the traffic details. Algorithm 1 shows the major steps used by the KUD-Vis system for network traffic visualization and analysis. It works in two modes, online() and offline(). In the online() mode, it captures, preprocesses, and splits the traffic instances and visualizes them for attack detection. But in offline() mode, it just visualizes the already stored preprocessed traffic instances for attack detection. So, cost of computation in the online() mode is more than in the offline mode to provide near real-time performance.

To test the performance of the KUD-Vis tool, we have tested it in two modes: offline and online. In offline mode, we used two benchmark datasets: (i) MIT Lincoln Laboratory [18] and (ii) CAIDA DDoS 2007 [6]. The MIT Lincoln Laboratory dataset is real time and contains pure normal data. This dataset was acquired as a tcpdump trace over a period of several weeks. It does not contain any attack traffic. The CAIDA DDoS 2007 dataset contains 1 h of anonymized traffic traces from a DDoS attack launched on August 4, 2007. This dataset includes mainly two types of attacks: consumption of computing resources and consumption of network bandwidth. While the data was collected, the servers were connected to the Internet. Figure 6.8 shows the main window of KUD-Vis in offline mode, and Fig. 6.9 shows the visualized network traffic in offline mode, when it uses the CAIDA DDoS 2017 datasets.

We also evaluate the KUD-Vis tool using the MIT Lincoln Laboratory dataset to differentiate between normal and attack traffic. Figure 6.10 shows the main window

Algorithm 1 KUD-Vis (online, offline)

Require: mode ▷ defines the mode of capturing packet
Ensure: The visualized Graph
if *mode* ≠ *online* **then**
  call online( )
else
  call offline( )
end if
function ONLINE( )
  Initialize *storePacket*[60] ▷ It is an array of linked list to store packets as they arrive, and 60 arrays each storing 5 sec data, total 5mins, *device*
  Find all the network devices connected to the machine
  device = get the choice of device from the list
  Open the *device* for capturing in promiscuous mode
  for $i \leftarrow 0, 59$ **do**
    storePacket[*i*] = CAPTURE(device)
  end for
  ANALYSE(storePacket)
  NODEVISUALGRAPH(storePacket)
  exit
end function
function NODEVISUALGRAPH(storePacket)
  Initialize *graph* ▷ an undirected graph where vertices are devices on the network and edges represent communication between them, *vertex* ▷ linked list of devices i.e. IP addresses, *edges* ▷ a linked list each having value (v_i, v_j) where v_i, $v_j \in$ vertex
  for $i \leftarrow 0$, to size of *storePacket* **do**
    for all *packet* in *storePacket*[*i*] **do**
      if *packet* is an IP packet **then**
        if *packet.sourceIP* not in *vertex* **then**
          add *packet.sourceIP* to *vertex*
        end if
        if *packet.destinationIP* not in *vertex* **then**
          add *packet.destinationIP* to *vertex*
        end if
        if (*packet.sourceIP*, *packet.destinatio* − *nIP*) not in *edges* **then**
          add (*packet.sourceIP*, *packet.desti* − *nationIP*)
        end if
      end if
    end for
  end for
  graph.addVertex(*vertex*)
  graph.addEdges(*edges*)
  return *graph*
end function
function OFFLINE( )
  Initialize *storePacket*[60] ▷ It is an array of linked list to store packets as they arrive, *time* = 5000 (It is time in milliseconds, 5000 represents 5 sec), $i = 0$ (for accessing the array)
  get the pcap file from user, i.e. *pcapFile*

(continued)

```
42:     open pcapFile to read packets
43:     for all packets in pcapFile do
44:         if packet.timestamp > time then
45:             time = time + 5000
46:             i + +
47:         end if
48:         add packet to storePacket[0]
49:     end for
50:     ANALYSE(storePacket)
51:     NODEVISUALGRAPH(storePacket)
52:     exit
53: end function
```

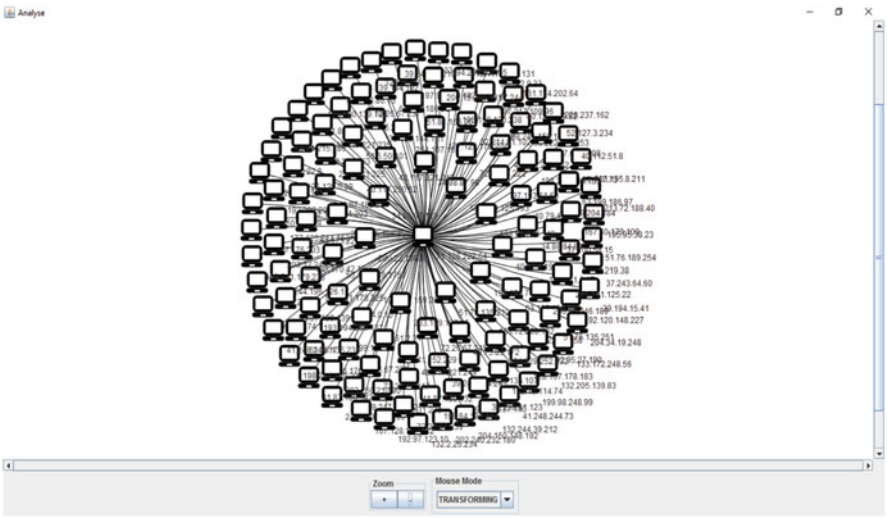


Fig. 6.9 KUD-Vis: visualization of network traffic available at CAIDA DDoS 2007 dataset

of KUD-Vis in offline mode, and Fig. 6.11 shows the visualized network traffic in offline mode using the MIT Lincoln Laboratory dataset.

In the online mode, we use the live network of Kaziranga University campus while executing attacks. Figure 6.12 shows the main window of our system in online mode using the KUD dataset. Simply, we capture traffic for only 40 s. Figures 6.13, 6.14, 6.15, 6.16, 6.17, and 6.18 show the window in analysis mode and represent graphs for *UniqueIP*, *SourceIP*, *DestinationIP*, *UniqueTCPPorts*, *UniqueUDPPorts*, and *Protocols*, respectively. Figure 6.19 shows the traffic visualization when executing attacks in our networks. We see that the visualization, all hosts in the network are not shown. It shows only those hosts that are active and either sending or receiving packets including the target.

As per Moore et al. [19], we generate low-rate DDoS attack traffic for evaluation of the KUD-Vis system. The attack traffic is generated at the rate of more than 3000 and less than 5000 packets per 5 s. This number varies depending on the dataset.

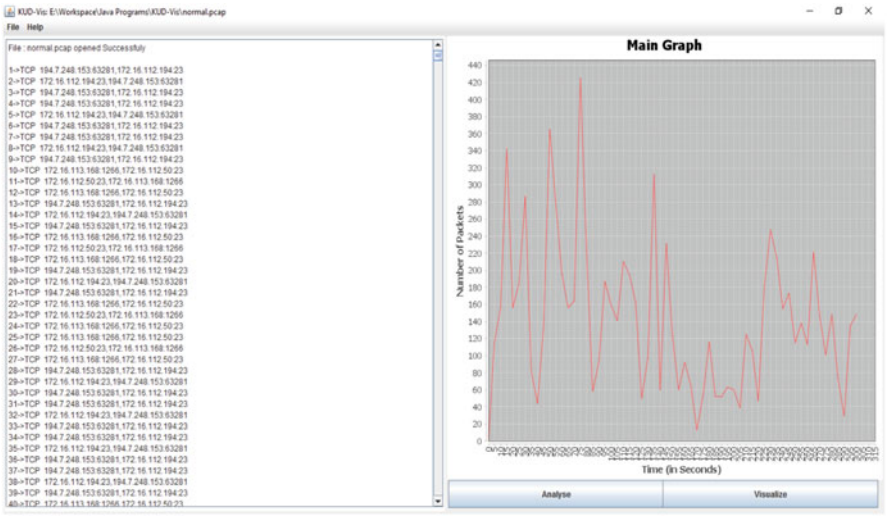


Fig. 6.10 KUD-Vis: working in offline mode using MIT Lincoln Laboratory dataset

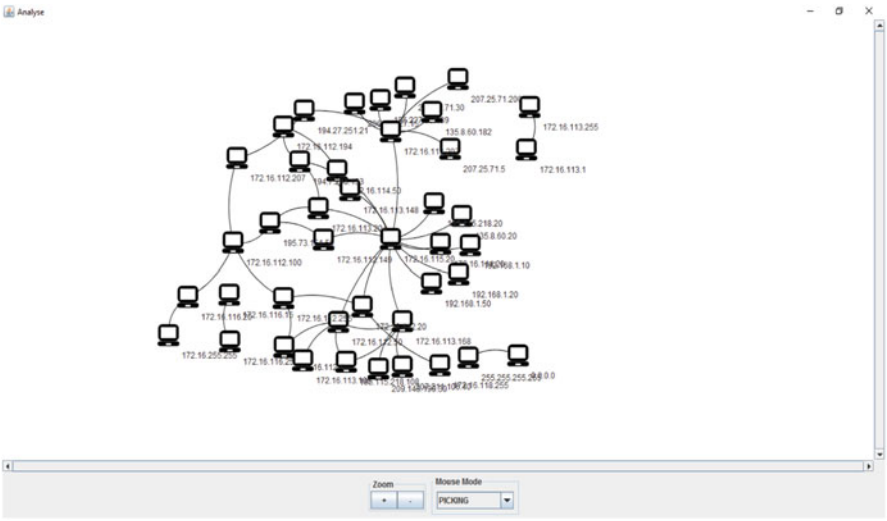


Fig. 6.11 KUD-Vis: visualization of network traffic using MIT Lincoln Laboratory dataset

Based on our experiments, we observe that KUD-Vis indicates an alarm when threshold $\tau_1 \geq 3000$ packets per seconds within a 2 second interval transmission over the network. The KUD-Vis system represents a significant development in view of the following points.

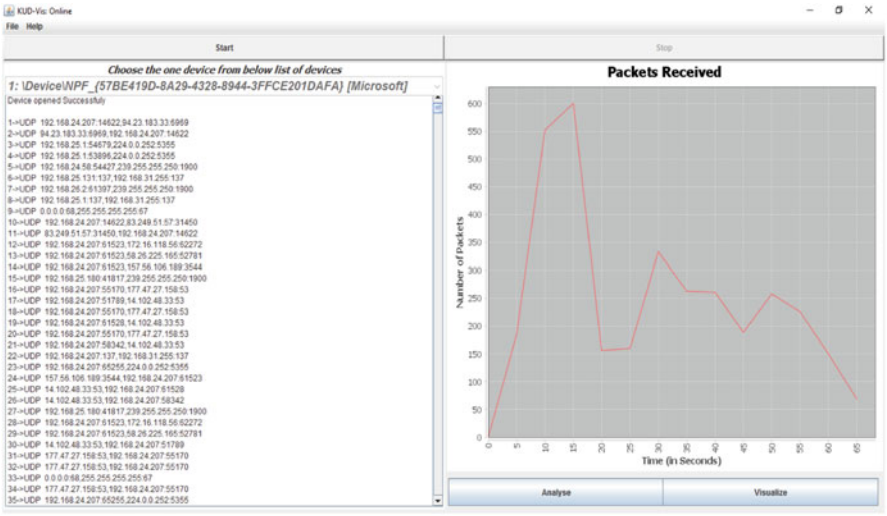


Fig. 6.12 KUD-Vis: working in online mode

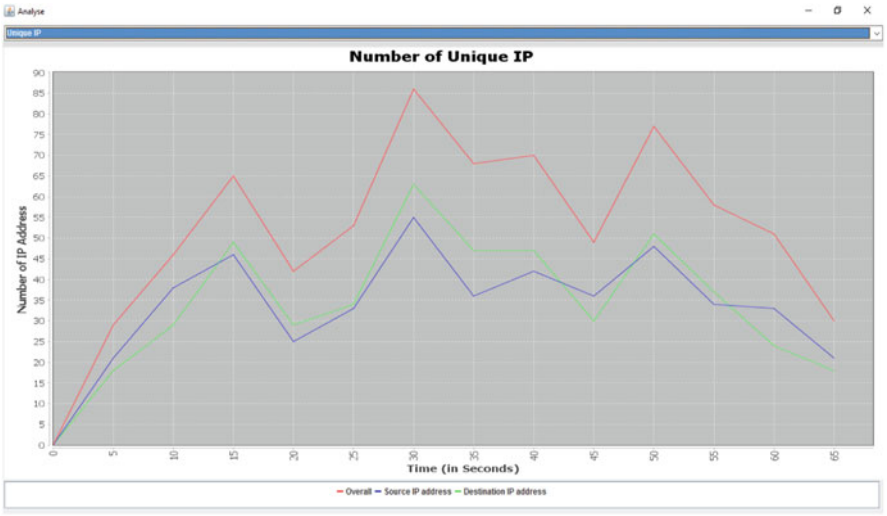


Fig. 6.13 Number of unique IP addresses

- KUD-Vis is cost-effective and can operate in both online and offline modes.
- It is fast and scalable.
- Accurate identification of low-rate DDoS attack is a difficult task. However, KUD-Vis is able to detect low-rate DDoS attacks effectively.

Fig. 6.14 Packets sent to different destination from particular sources

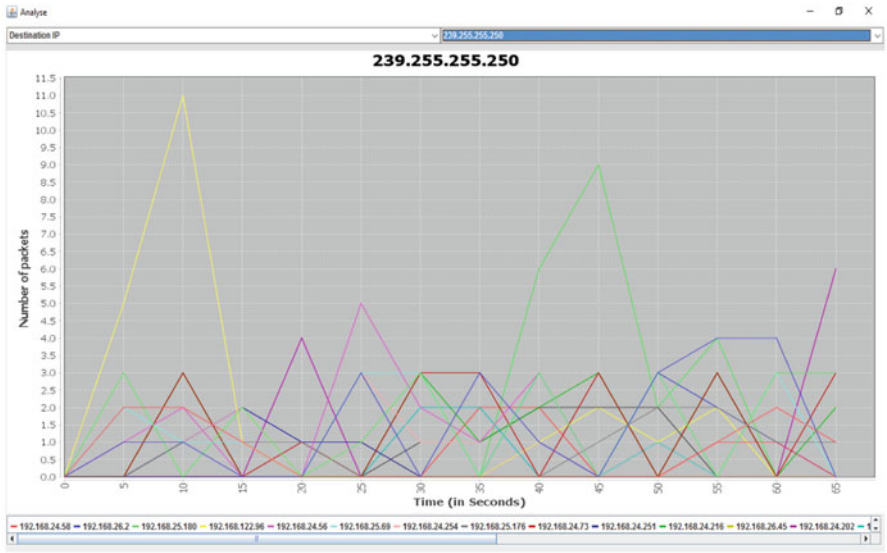


Fig. 6.15 Packets sent from different sources to a particular destination

6.7 Chapter Summary

This chapter has presented major tools that can be used by attackers as well as defenders, as initially given in Fig. 6.2. Even though several tools are available for the research community, the appropriate use of such tools is a major concern in real-time network security infrastructure. This chapter started with a brief description of

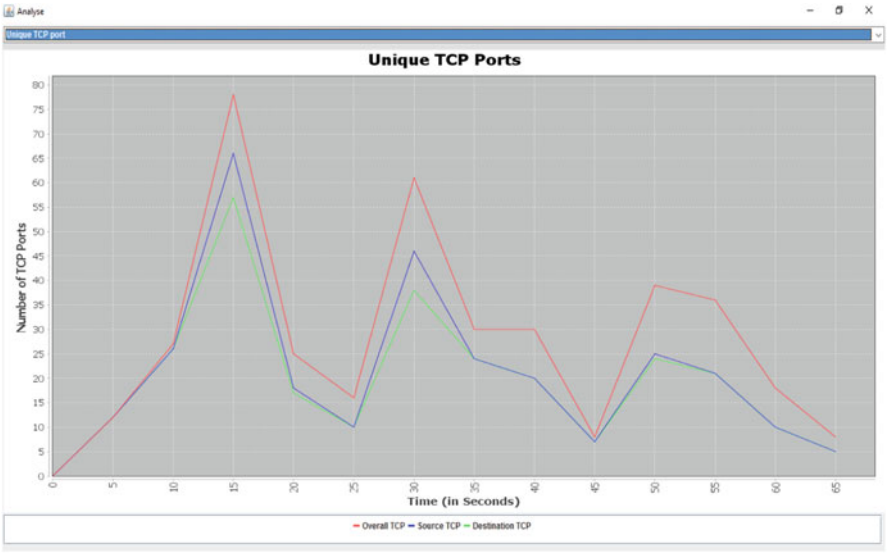


Fig. 6.16 Number of unique TCP ports

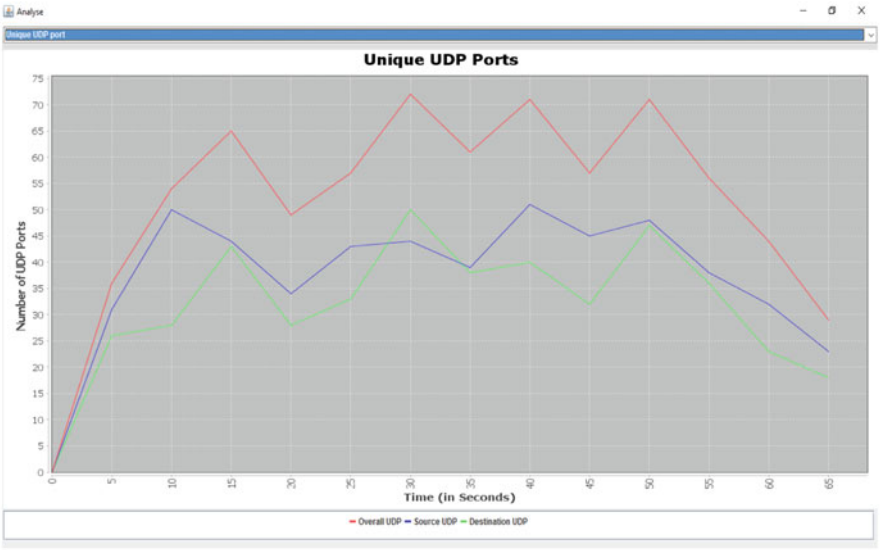


Fig. 6.17 Number of unique UDP ports

network attacks, security tools, their characteristics, and steps to launch an attack in a real-time environment. We covered the popular tools for launching of network attacks as well as for defending a network from attacks. Based on our detailed study of relevant issues, we make the following observations.

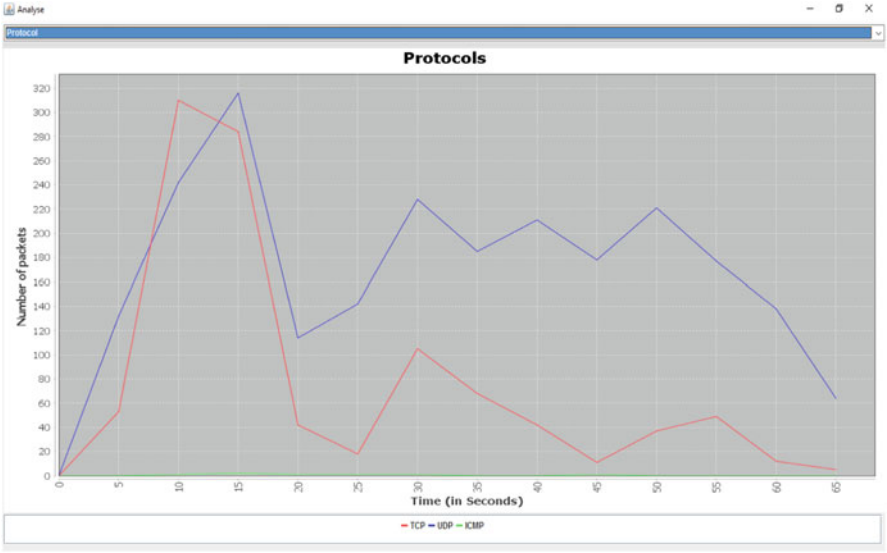


Fig. 6.18 Number of packets per protocol

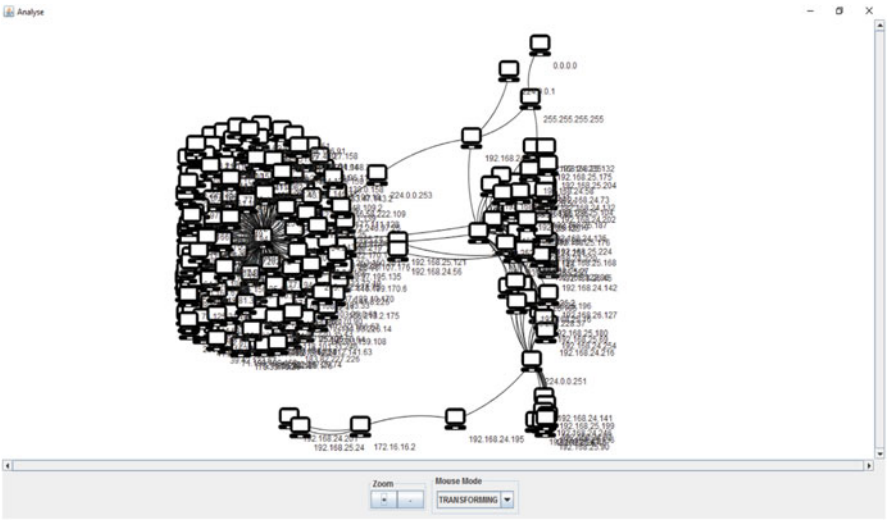


Fig. 6.19 KUD-Vis: visualization of traffic for our network

- Most currently available tools are limited in their abilities usually with a small number of features. So, development of an integrated tool to support capture, preprocessing, analysis, and visualization of both packet and NetFlow traffic remains a crucial need.

- Successful functioning of most attack detection tools is dependent on the setting of suitable values to necessary parameters. These parameters usually need to be adjusted for the environment where the tools are installed.
- It seems that existing tools cannot perform coordinated scan operations efficiently when time is of essence, especially in enterprise networks.
- Existing DDoS attacks generation tools are usually restricted to a limited number of attacks and their variations. It is difficult to customize such tools for multiple scenarios.
- Most existing attack generation tools do not support multilayer attack scenarios.
- Existing tools are unable to correlate between packet level traffic and NetFlow level traffic well.

Finally, we discussed an approach to developing a real-time network traffic monitoring and analysis tool. This tool is illustrated in a step by step manner in terms of design and accompanying code. We believe that this discussion may be of value to a researcher or practitioner who may try to develop a tool on his/her own.

References

1. Aydın, M., Zaim, A., Ceylan, K.: A hybrid intrusion detection system design for computer network security. Comput. Electr. Eng. **35**(3), 517–526 (2009)
2. Beverly, R.: A robust classifier for passive TCP/IP fingerprinting. In: Passive and Active Network Measurement, Antibes Juan-les-Pins, pp. 158–167 (2004)
3. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: NADO: network anomaly detection using outlier approach. In: Proceedings of the International Conference on Communication, Computing & Security, Odisha, pp. 531–536. ACM (2011)
4. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Survey on incremental approaches for network anomaly detection. Int. J. Commun. Netw. Inf. Secur. **3**(3), 226–239 (2011)
5. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Network anomaly detection: methods, systems and tools. IEEE Commun. Surv. Tutorials **16**(1), 303–336 (2014). doi:10.1109/SURV.2013.052213.00046
6. CAIDA: The cooperative analysis for Internet data analysis (2011). http://www.caida.org
7. Chen, W.H., Hsu, S.H., Shen, H.P.: Application of SVM and ANN for intrusion detection. Comput. Oper. Res. **32**(10), 2617–2634 (2005)
8. Danielle, L.: Introduction to Dsniff. In: Global Information Assurance Certification Paper. SANS Institute (2002)
9. Debar, H., Dacier, M., Wespi, A.: Towards a taxonomy of intrusion-detection systems. Comput. Netw. **31**(9), 805–822 (1999)
10. Garber, L.: Denial-of-service attacks RIP the Internet. Computer **33**(4), 12–17 (2000). doi:10.1109/MC.2000.839316
11. Girardin, L.: An eye on network intruder-administrator shootouts. In: Proceedings of the 1st Conference on Workshop on Intrusion Detection and Network Monitoring, ID'99, vol. 1, pp. 3–3. USENIX Association, Berkeley (1999)
12. Inselberg, A., Dimsdale, B.: Parallel coordinates: a tool for visualizing multi-dimensional geometry. In: Proceedings of the 1st Conference on Visualization '90, VIS 90, pp. 361–378. IEEE Computer Society Press, Los Alamitos (1990). http://dl.acm.org/citation.cfm?id=949531.949588

13. Jemili, F., Zaghdoud, M., Ben Ahmed, M.: A framework for an adaptive intrusion detection system using Bayesian network. In: Proceedings of the IEEE Intelligence and Security Informatics, pp. 66–70 (2007)
14. jNetPcap: jNetPcap – what is it?. http://jnetpcap.com/
15. Kallitsis, M.G., Stoev, S., Bhattacharya, S., Michailidis, G.: AMON: an open source architecture for online monitoring, statistical analysis and forensics of multi-gigabit streams. CoRR **abs/1509.00268** (2015)
16. Li, X., Bian, F., Crovella, M., Diot, C., Govindan, R., Iannaccone, G., Lakhina, A.: Detection and identification of network anomalies using sketch subspaces. In: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, IMC '06, pp. 147–152. ACM, New York (2006)
17. Lippmann, R.P., Cunningham, R.K.: Improving intrusion detection performance using keyword selection and neural networks. Comput. Netw. **34**(4), 597–603 (2000)
18. MIT Lincoln Laboratory Datasets: MIT LLS_DDOS_0.2.2 (2000). http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/2000data.html
19. Moore, D., Shannon, C., Brown, D.J., Voelker, G.M., Savage, S.: Inferring Internet Denial-of-service activity. ACM Trans. Comput. Syst. **24**(2), 115–139 (2006). doi:10.1145/1132026.1132027
20. Norton, D.: An Ettercap Primer. In: SANS Institute InfoSec Reading Room (2004)
21. Ranjan, S., Swaminathan, R., Uysal, M., Knightly, E.: DDoS-resilient scheduling to counter application layer attacks under imperfect detection. In: Proceedings of the 25th IEEE International Conference on Computer Communications, pp. 1–13 (2006)
22. Rnmap: Rnmap – remote nmap. http://rnmap.sourceforge.net/
23. Schiffman, M.D.: Libnet 101, Part 1: the primer. In: Guardent Security Digital Infrastructure, pp. 1–10 (2000)
24. Shah, S.: An Introduction to HTTP Fingerprinting. Net-Square Solutions (2004)
25. Singh, S., Estan, C., Varghese, G., Savage, S.: Automated worm fingerprinting. In: Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation, vol. 6, pp. 4–4. USENIX Association, Berkeley (2004)
26. Whalen, Kevin: DDoS Attacks: Beware Headline Risk. https://www.arbornetworks.com/blog/insight/ddos-attacks-beware-headline-risk/
27. Xie, Y., Yu, S.Z.: Monitoring the application-layer DDoS attacks for popular websites. IEEE/ACM Trans. Netw. **17**(1), 15–25 (2009)
28. Yarochkin, F.: Remote OS detection via TCP/IP stack fingerprinting. Phrack Mag. **17**(3) (1998)
29. Ye, N., Ehiabor, T., Zhang, Y.: First-order versus high-order stochastic models for computer intrusion detection. Qual. Reliab. Eng. Int. **18**(3), 243–250 (2002)
30. Yeung, K.H., Fung, D., Wong, K.Y.: Tools for attacking layer 2 network infrastructure. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, vol. 2, pp. 1–6 (2008)
31. Yin, X., Yurcik, W., Treaster, M., Li, Y., Lakkaraju, K.: VisFlowConnect: netflow visualizations of link relationships for security situational awareness. In: Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC 2004), Washington DC, 29 Oct 2004, pp. 26–34 (2004). doi:10.1145/1029208.1029214
32. Yu, J., Li, Z., Chen, H., Chen, X.: A detection and offense mechanism to defend against application layer DDoS attacks. In: Proceedings of the 3rd International Conference on Networking and Services, pp. 54–54. IEEE (2007)

Chapter 7
Evaluation Criteria

Performance evaluation is a major part of any network traffic anomaly detection technique or system. Without proper evaluation, it is difficult to make the case that a detection mechanism can be deployed in a real-time environment. An evaluation of a method or a system in terms of accuracy or quality provides a snapshot of its performance in time. As time passes, new vulnerabilities may evolve, and current evaluations may become irrelevant. The evaluation of an intrusion detection system (IDS) involves activities such as collection of attack traces, construction of a proper IDS evaluation environment and adoption of solid evaluation methodologies. In this chapter, we introduce commonly used performance evaluation measures for IDS evaluation. The main measures include accuracy, performance, completeness, timeliness, reliability, quality and AUC area. It is beneficial to identify the advantages and disadvantages of different detection methods or systems.

7.1 Accuracy

Accuracy is a metric that measures how correctly an IDS works, measuring the percentage of detection and failure as well as the number of false alarms that the system produces [1, 5]. If a system has 90% accuracy, it means that it correctly classifies 90 instances out of 100 as belonging to their actual classes. While there is a big diversity of attacks in intrusion detection, the main objective is that the system be able to detect an attack correctly. In general, the actual percentage of abnormal data is much smaller than that of the normal [2, 4, 7]. There are also a lot of different ways to intrude into a network or a system. Consequently, intrusions are harder to detect than normal traffic, resulting in excessive false alarms as the

M.H. Bhuyan et al., *Network Traffic Anomaly Detection and Prevention*,
Computer Communications and Networks, DOI 10.1007/978-3-319-65188-0_7

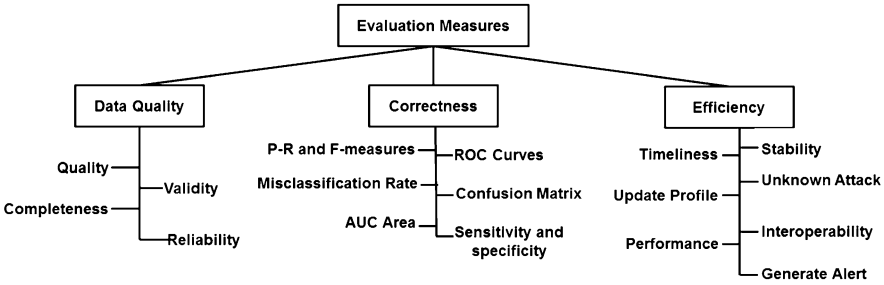


Fig. 7.1 Taxonomy of evaluation measures

biggest problem facing IDSs. Figure 7.1 shows the taxonomy of evaluation measures for network anomaly detection. This taxonomy represents mainly three different dimensions of evaluation, namely, correctness, efficiency, and characteristics of the data. The following are some accuracy measures.

7.2 Data Quality

Data quality is a major component to evaluate any network traffic anomaly detection techniques and systems. Because without good quality of data, usually it fails to test any detection and prevention techniques and systems. It is comprised of four parts including quality, validity, reliability, and completeness.

7.2.1 Quality

Evaluating the quality of data (used to evaluate an ANIDS) is another important task in ANIDS evaluation. The quality of the data is influenced by several factors, such as (i) source of data (should be from reliable and legitimate), (ii) selection of samples (should be unbiased), (iii) sample size (neither over nor under-sampling), (iv) time of data (should be frequently updated real-time data), (v) complexity of data (should be simple enough to be handled easily by the detection mechanism), and so on.

7.2.2 Reliability

Data reliability plays an important role in network traffic anomaly detection and prevention techniques and systems. Reliability refers to the data elements used

for evaluation of a system, and results should be accurate and consistent and must meet the expected purpose. Data reliability depends on quality, validity, and completeness.

7.2.3 Validity

Data quality and validity provide better information regarding the actual network traffic recorded for evaluation. Because parameters are crucial component for each step of designing to implementation of a detection and prevention technique and system. However, if data quality is not validated irrespective of techniques, then result obtained from such data would have more limitations in terms of quantifying any conclusion.

7.2.4 Completeness

The completeness criterion represents the space of the vulnerabilities and attacks that can be covered by an IDS. This criterion is very hard to assess because omniscience about attacks or abuses of privilege is impossible. The completeness of an IDS is judged against a complete set of known attacks. The ability of an IDS is considered complete, if it covers all the known vulnerabilities and attacks.

7.3 Correctness

For any network anomaly detection and prevention techniques and systems, it is always necessary to ensure the correctness of the technique or system in terms of several evaluation measures such as ROC curve, AUC area, precision, recall, F-measure, confusion matrix, misclassification rate, sensitivity, and specificity.

7.3.1 ROC Curve

The Receiver Operating Characteristics (ROC) analysis originates from signal processing theory. Its applicability is not limited only to intrusion detection but extends to a large number of practical fields such as medical diagnosis, radiology, bioinformatics as well as artificial intelligence and data mining. In intrusion detection, ROC curves are used on the one hand to visualize the relation between TP and FP rates of a classifier while tuning it and on the other hand to compare the accuracies of two or more classifiers. The ROC space [6, 8] uses the orthogonal

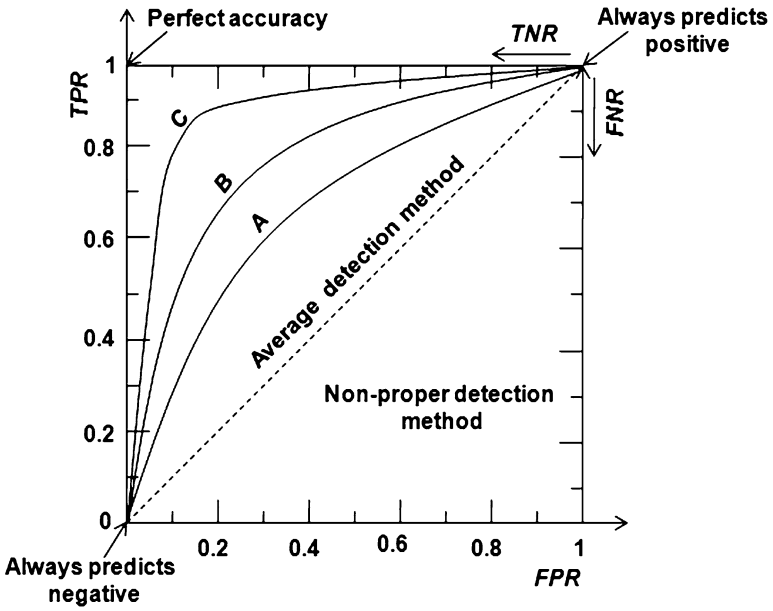


Fig. 7.2 Illustration of ROC measure where *A, B, C* represents the accuracy of a detection method or a system in ascending order

coordinate system to visualize the classifier accuracy. Figure 7.2 illustrates the ROC approach normally used for network anomaly detection methods and systems evaluation.

To understand the ROC curve, one should remember the following important points when evaluating a detection method or system [6, 8]:

- The lower-left point (0,0) characterizes an IDS that classifies all the data as normal all the time. Obviously in such a situation, the classifier has a zero false alarm rate but does not really detect anything.
- The upper-right point (1,1) characterizes an IDS that generates an alarm for each data point that is encountered. Consequently, it has a 100% detection rate and 100% false alarm rate as well.
- A line defined by connecting (0, 0) and (1,1), the two previous points, represents a classifier that uses a randomized decision engine for detecting the intrusions. Any point on this line can be obtained by a linear combination of the two previously mentioned strategies. Thus, the ROC curve of an IDS is always situated above this diagonal.
- The upper-left point (0,1) represents the ideal case when there is a 100% detection rate while having a 0% false alarm rate. Thus, the closer a point in the ROC space is to the ideal case, the more efficient the classifier is.

7.3.2 AUC Area

The most common measure of total detection accuracy is the magnitude of the area under the curve or AUC. It summarizes the total accuracy of the detector in a way that accounts for both the gains in the true positive (TP) rate and the losses in the false-positive (FP) rate. The value of AUC always ranges from 0.5 to 1.0, because the very worst ROC curve lies along the positive diagonal of ROC curve and has an area of 0.5. The very best ROC curve, passing through the "northwest" corner point (0,1), has an area of 1 or the unit square.

To find the AUC, simply calculate the trapezoidal area under each vertical slice of an empirical (unsmoothed) ROC curve having a straight-line segment as its top; then sum all the individual areas. For instance, if a binary detector produces only a single data point, then there will be two trapezoidal regions total for which to calculate area (one to the left of the point and one to the right of the point).

One drawback of the AUC measure is that given a single AUC number, one cannot reconstruct the shape of the curve which produced it. It is ambiguous precisely because it abstracts away the two-dimensional shape of the curve. If there is a specific use in mind for the signal detector, this may not be the best measure to use, because it gives only a very general picture of total accuracy (similar to an average accuracy over all possible modes of the detector). An ROC curve with a higher AUC may actually perform worse under some conditions than another ROC curve with a smaller AUC number.

However, when comparing together two or more ROC curves using the AUC measure, the curve with the greater AUC is truly more accurate (when averaged over all detection thresholds). However, we must keep in mind that sometimes it is more important for a curve to be above another (to dominate) over certain entire regions of the graph than for one curve to have a greater AUC than another.

7.3.3 Precision, Recall, and F-Measure

Precision is a measure of how well a system identifies attacks or normals. A flagging is accurate if the identified instance indeed comes from a malicious user; this is referred to as true positive. The final quantity of interest is recall, a measure of how many instances are identified correctly (see Fig. 7.3). Precision and recall are often inversely proportional to each other, and there is normally a trade-off between these two ratios. An algorithm that produces low precision and low recall is most likely defective with conceptual errors in the underlying theory. The types of attacks that are not identified can indicate which areas of the algorithm need more attention. Exposing these flaws and establishing the causes assist future improvement.

The F-measure combines these two measures as the harmonic mean of precision and recall [3, 11]. If we want to use only one accuracy metric as an evaluation criterion, F-measure is the most preferable. Note that when precision and recall both

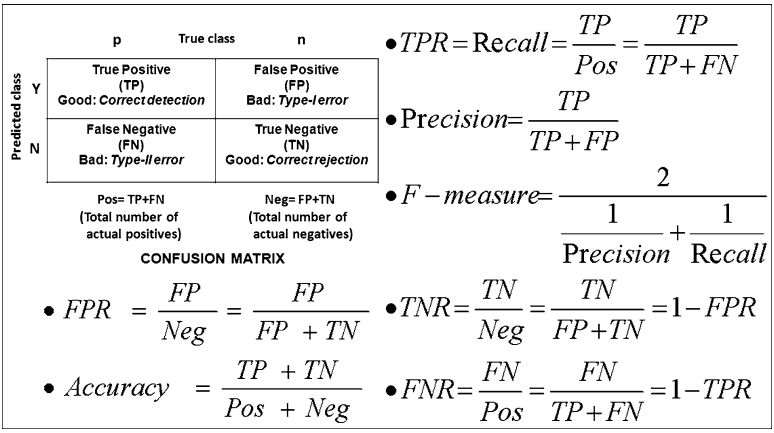


Fig. 7.3 Confusion matrix and related evaluation measures

reach 100%, the F-measure is the maximum, i.e., 1 meaning that the classifier has 0% false alarms and detects 100% of the attacks. Thus, a good classifier is expected to obtain F-measure as high as possible.

7.3.4 Confusion Matrix

The confusion matrix is an evaluation method that can be applied to any kind of classification problem. The size of this matrix depends on the number of distinct classes to be detected. The aim is to compare the actual class labels against the predicted ones as shown in Fig. 7.3. The diagonal represents correct classification. The confusion matrix for intrusion detection is defined as a 2-by-2 matrix, since there are only two classes known as intrusion and normal [2, 3, 11]. Thus, the TNs and TPs that represent the correctly predicted cases lie on the matrix diagonal, while the FNs and FPs are on the right and left sides. As a side effect of creating the confusion matrix, all four values are displayed in a way that the relation between them can be easily understood.

7.3.5 Misclassification Rate

This measure attempts to estimate the probability of disagreement between the true and predicted cases by dividing the sum of FN and FP by the total number of cases observed, i.e., (TP+FP+FN+TN). In other words, misclassification rate is defined as (FN+FP)/(TP+FP+FN+TN).

7.3.6 *Sensitivity and Specificity*

These two measures [10] attempt to measure the accuracy of classification for a two-class problem. When an IDS classifies data, its decision can be either right or wrong. It usually generates true for right and false for wrong, respectively.

If S is a detector and D_t is the set of test instances, there are four possible outcomes described using the confusion matrix given in Fig. 7.3. When an anomalous test instance (p) is predicted as anomalous (Y) by the detector S, it is counted as true positive (TP); if it is predicted as normal (N), it is counted as false negative (FN). On the other hand, if a normal (n) test instance is predicted as normal (N), it is known as true negative (TN), while it is a false positive (FP) if it is predicted as anomalous (Y) [3, 10, 11].

The true positive rate (TPR) is the proportion of anomalous instances classified correctly to the total number of anomalous instances present in the test data. TPR is also known as *sensitivity*. The false-positive rate (FPR) is the proportion of normal instances incorrectly classified as anomalous to the total number of normal instances contained in the test data. The true negative rate (TNR) is also called *specificity*. TPR, FPR, TNR, and the false-negative rate (FNR) can be defined for the normal class. We illustrate all measures related to the confusion matrix in Fig. 7.4.

Sensitivity is also known as the hit rate. Between sensitivity and specificity, sensitivity is set at high priority when the system is to be protected at all cost, and specificity gets more priority when efficiency is of major concern [10]. Consequently, the aim of an IDS is to produce as many TPs and TNs as possible while trying to reduce number of both FPs and FNs. The majority of evaluation criteria use these variables and the relations among them to model the accuracy of the IDSs.

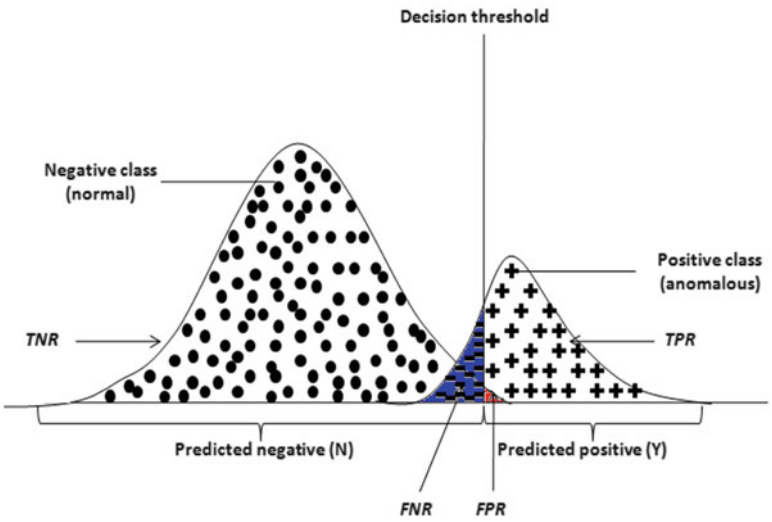


Fig. 7.4 Illustration of confusion matrix and related evaluation measures

7.4 Efficiency

Efficiency refers to the performance of a technique or a system on how they fulfill the expected outcome within a stipulated time period. Following components ensures the efficiency of any detection and prevention techniques or systems. It also ensures fastest result with consistency and within bounded time period.

7.4.1 Stability

An anomaly detection system should perform consistently in different network scenarios and in different circumstances. It should consistently report identical events in a similar manner. Allowing the users to configure different alerts to provide different messages in different network environments may lead to an unstable system.

7.4.2 Timeliness

An IDS that performs its analysis as quickly as possible enables the human analyst or the response engine to promptly react before much damage is done. This prevents the attacker from subverting the audit source or the IDS itself. The response generated by the system while combating an attack is very important. Since the data must be processed to discover intrusions, there is always a delay between the actual moment of the attack and the response of the system. This is called *total delay*, the time which the attack took place and the time of response, respectively. Thus, the total delay is the difference between t_{attack} and t_{response}. The smaller the total delay, the better an IDS is with respect to its response. No matter if an IDS is anomaly-based or signature-based, there is always a gap between the starting time of an attack and its detection.

7.4.3 Performance

The evaluation of an IDS's performance is an important task. It involves many issues that go beyond the IDS itself. Such issues include the hardware platform, the operating system, or even the deployment environment of the IDS. For a NIDS, the most important evaluation criterion for its performance is the system's ability to process traffic on a high-speed network with minimum packet loss when working real time. In real network traffic, the packets can be of various sizes; however, the effectiveness of a NIDS depends on its ability to handle packets of

any size. In addition to the processing speed, the CPU and memory usage can also serve as measurements of a NIDS's performance [9]. These are usually used as indirect measures that take into account the time and space complexities of intrusion detection algorithms. Finally, the performance of a NIDS is highly dependent upon (i) its individual configuration, (ii) the network it is monitoring, and (iii) its position in that network.

7.4.4 Update Profile

Once new vulnerabilities or exploits are discovered, signatures or profiles must be updated for future detection. However, writing new or modified profiles or signatures accurately and compactly without conflict is a challenge, considering the current high-speed network scenario.

7.4.5 Interoperability

An effective intrusion detection mechanism is supposed to be capable of correlating information from multiple sources, such as system logs, other HIDSs, NIDSs, firewall logs, and any other sources of relevant information available. This helps maintain interoperability, when an organization installs a range of HIDSs or NIDSs from various vendors.

7.4.6 Unknown Attack

New vulnerabilities are evolving almost every day. An anomaly-based network intrusion detection system should be capable of identifying unknown attacks, in addition to known attacks. The IDS should show consistent abilities in detecting unknown or even modified intrusion patterns.

7.5 Information Provided to Analyst

Alerts generated by an IDS should be meaningful enough to clearly identify the reasons behind the event to be raised and the reasons this event is of interest. It should also assist the analyst in determining the relevance and appropriate reaction to a particular alert. The alert should also specify the source of the alert and the target system.

7.6 Chapter Summary

This chapter presents evaluation measures for network traffic anomaly detection methods and systems. Evaluation measures include accuracy, performance, completeness, timeliness, reliability, and quality. Accuracy is further discussed in terms of sensitivity and specificity, ROC curves, AUC area, misclassification rate, confusion matrix, precision, recall, and F-measure. The use of these measures for ANIDS evaluation has advantages such as the following: (i) Accuracy is mainly effective in identifying false alarm rate of a detection method or system. (ii) Performance is evaluated based on the individual network configuration and position of the ANIDS in the network. Before deployment of a detection method or system, it is indeed necessary to evaluate it in realistic environments. So, evaluation measure needs to be chosen based on the nature of the network, the location of the ANIDS, the speed, the bandwidth capability of the network, and the size of the network in terms of devices, servers, appliances, etc.

References

1. Axelsson, S.: The base-rate fallacy and the difficulty of intrusion detection. ACM Trans. Inf. Syst. Secur. **3**(3), 186–205 (2000)
2. Dokas, P., Ertoz, L., Lazarevic, A., Srivastava, J., Tan, P.N.: Data mining for network intrusion detection. In: Proceedings of the NSF Workshop on Next Generation Data Mining (2002)
3. Ghorbani, A.A., Lu, W., Tavallaee, M.: Network Intrusion Detection and Prevention: Concepts and Techniques. Advances in Information Security. Springer, New York (2009)
4. Joshi, M.V., Agarwal, R.C., Kumar, V.: Predicting rare classes: can boosting make any weak learner strong? In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 297–306. ACM, New York (2002)
5. Lippmann, R.P., Fried, D.J., Graf, I., Haines, J., Kendall, K., McClung, D., Weber, D., Wyschogord, S.W.D., Cunningham, R.K., Zissman, M.A.: Evaluating intrusion detection systems: the 1998 DARPA offline intrusion detection evaluation. In: Proceedings of the DARPA Information Survivability Conference and Exposition, pp. 12–26 (2000)
6. Maxion, R.A., Roberts, R.R.: Proper use of ROC curves in intrusion/anomaly detection. Technical report CS-TR-871, School of Computing Science, University of Newcastle upon Tyne (2004)
7. Portnoy, L., Eskin, E., Stolfo, S.: Intrusion detection with unlabeled data using clustering. In: Proceedings of the ACM CSS Workshop on Data Mining Applied to Security, Philadelphia, pp. 5–8 (2001)
8. Provost, F.J., Fawcett, T.: Robust classification for imprecise environments. Mach. Learn. **42**(3), 203–231 (2001)
9. Sekar, R., Guang, Y., Verma, S., Shanbhag, T.: A high-performance network intrusion detection system. In: Proceedings of the 6th ACM Conference on Computer and Communications Security, pp. 8–17. ACM, New York (1999)
10. Wang, Y.: Statistical Techniques for Network Security: Modern Statistically-Based Intrusion Detection and Protection. Information Science Reference, IGI Publishing, Hershey (2008)
11. Weiss, S.M., Zhang, T.: Performance alanysis and evaluation. In: Ye, N. (ed.) The Handbook of Data Mining, pp. 426–439. Lawrence Erlbaum Associates, Mahwah (2003)

Chapter 8
Open Issues, Challenges, and Conclusion

It is hoped that this book increases awareness of the reader of threats that have come into existence recently and techniques, systems, and tools for detecting such threats. Any antivirus or defense software can only detect the threats if and only if the defender software understands how attackers get entry into a system and what tools they use to compromise a network or system. This chapter is focused on the open issues and challenges faced by the ANIDS research community.

8.1 Open Issues and Challenges

Even though many network traffic anomaly detection techniques, systems, and tools have been developed for protection of enterprise networks from different intelligent attacks, there are still a number of open issues and challenges. The most common bottleneck is defining boundaries for legitimate traffic while detecting network traffic anomalies, because they vary as a function of network size, architecture, environment, and organization. Tuning such parameters to detect network traffic anomalies is a difficult task. Another common issue is nonavailability of suitable performance metrics to evaluate detection techniques or systems. For evaluation of detection techniques or systems, common metrics are accuracy, performance, completeness, timeliness, and data quality.

Network traffic anomaly detection techniques and systems should be evaluated against valid, unbiased, and high-quality data. But getting such data with necessary variations is expensive and difficult. Without validating a detection technique or system, we cannot deploy the detection system in real-time environment of a network we are trying to protect. On the other hand, formally proving that a detection system behaves correctly is expensive, with respect to a particular environment [1, 2]. Generation of unbiased, realistic, and comprehensive datasets is still a challenge. Our study on network traffic anomaly detection techniques,

M.H. Bhuyan et al., *Network Traffic Anomaly Detection and Prevention*,
Computer Communications and Networks, DOI 10.1007/978-3-319-65188-0_8

systems, and tools has demonstrated that there is still a need to ensure scalability, high performance, and robustness when designing a detection technique or system or tool. We identify and enumerate some important issues and challenges based on our understanding of various dimensions of network traffic anomalies.

8.1.1 Reducing False Alarm Rate

An ANIDS or detection technique should ideally avoid a high rate of false alarms. However, it is not possible to escape totally from false alarms, even though it needs to be a good in any environment. The ANIDS should also facilitate adaptability at runtime. These are challenging tasks for the ANIDS development community.

8.1.2 Runtime Limitations

How to work properly at runtime in a real network environment remains a challenge from the perspective of data size as well. A real-time ANIDS should ideally be able to capture and inspect each packet without losing any packets in the process. For high-speed networks, it is difficult to capture all packets from live networks.

8.1.3 Reducing Environment Dependency

The performance of most ANIDSs and network intrusion detection techniques depends on the nature of the environment. Ideally, a system or technique should be independent of the environment. In other words, it should perform more or less at the same level in any environment in which it is installed.

8.1.4 Adaptability of ANIDS

The nature of anomalies keeps changing over time as intruders adapt their network attacks to evade existing intrusion detection solutions. So, adaptability of an ANIDS or detection method is necessary to update with the current anomalies encountered in the local network or the Internet.

8.1.5 Dynamic Updation of Profiles

Dynamic updation of profiles in an anomaly-based NIDS without introduction of conflicts among existing profiles and without compromising detection performance is an important task. The profile database needs to be updated maintaining consistency whenever a new kind of attack is detected and addressed by the system.

8.1.6 Generation of Unbiased and Realistic Intrusion Datasets

Preparing an unbiased network intrusion dataset with all normal variations in profiles is another challenging task. The number of normal instances is usually large, and their proportion with attack instances is very skewed in the existing publicly available intrusion datasets. Only a few intrusion datasets with sufficient amount of attack information are available publicly. Thus, there is an overarching need for benchmark intrusion datasets for evaluating ANIDSs and detection techniques.

8.1.7 Reducing Computational Complexity

Reducing computational complexity in preprocessing, training, and deployment is another task that needs to be addressed.

8.1.8 Detection and Handling Large-Scale Attacks

Due to continuous improvements in attack technology, state-of-the-arts attackers are able to compromise target systems or networks in a distributed manner within a short time interval. So, an ANIDS should not only be able to detect such attacks at an early stage but also limit the rate at which bad packets arrive without compromising access by legitimate users.

8.1.9 Reducing Dimensionality in Datasets

Network traffic datasets are high dimensional and large in volume. Some traffic features may have no variation in values, but the network may be functionally dependent on them. So, developing an effective technique to select optimal features to detect network traffic anomalies without compromising performance is another issue that needs to be addressed.

8.1.10 Multilayer DDoS Attack Detection

Attacker attempts to compromise the target host or network by considering network, transport, and application layers; possibility for detection of such attacks is low. Because cross-layer attacks are more intelligent than stand-alone attacks. So, detection of such attacks with low rate of false alarm is really difficult.

8.1.11 Traceback of Attacker Identity

Traceback and detect the source of attack(s) in real time with high accuracy is another important open challenge.

8.1.12 Dynamic and Adaptive Learning

Use of dynamic and adaptive learning that works with minimum knowledge for novel attack classification in real time is another challenge.

8.1.13 Protection Against IoT-Based DDoS Attacks

Protection of resources from IoT -enabled DDoS attacks is a challenge, which may be highly intensive as recently experienced of the order of 1.5 TB DDoS attacks.

8.2 Conclusion

In this book, we focus on network traffic anomaly detection and prevention techniques, systems, and tools with an attempt to provide a detailed experience to readers from novice to advanced level. We have discussed the characteristics of network traffic anomalies that result from the presence of malicious activities on the network. This book also provides a detailed discussion on vulnerabilities in a functioning real network at each layer that may be present due to weaknesses in protocol or design of the network. We have discussed a systematic approach to generate large and realistic network traffic datasets that contain both packet- and flow-level features. We have discussed network anomaly detection techniques and systems such as statistical, classification, clustering and outlier detection, soft computing, knowledge-based, and combination learners. We also provide advantages and disadvantages of each detection technique or system. Such analysis

might result in increasing readers' understanding of each class of techniques and systems. To help understand how attacks are related to one another, we have introduced an attack taxonomy hoping that readers will get clear ideas about the attacks with their characteristics. ANIDSs generate different types of alerts that need to be handled promptly, within short time interval. We have discussed the basic concept of alert and alert management. We hope that post-diagnosis of such alarms might increase the efficiency of detection techniques and systems. We also have presented different network intrusion prevention techniques and systems to help readers in developing a better understanding. We believe that a diligent reader will have acquired an adequate amount of knowledge regarding different attacks. We also have discussed several tools for both attackers and defenders. We also outlined how to develop a real-life network traffic monitoring and analysis tool for detection of network anomalies. Finally, we have briefly presented open issues and challenges in this chapter to help interested readers develop the ability to motivate by providing up-to-date as well as future solutions for the protection of enterprise networks.

References

1. Leite, A., Girardi, R.: A hybrid and learning agent architecture for network intrusion detection. J. Syst. Softw. (2017, in press). doi:10.1016/j.jss.2017.01.028
2. Patcha, A., Park, J.M.: An overview of anomaly detection techniques: existing solutions and latest technological trends. Comput. Netw. **51**(12), 3448–3470 (2007)

Index

M.H. Bhuyan et al., *Network Traffic Anomaly Detection and Prevention*,
Computer Communications and Networks, DOI 10.1007/978-3-319-65188-0