Rolf Oppliger

Internet Security: FIREWALLS *and* BEY

THE PARADIGM SHIFT FROM ALGORITHMS to interaction captures the technology shift from mainframes to workstations and networks, from number-crunching to embedded systems and graphical user interfaces, and from procedure-oriented to object-based and distributed programming. The radical notion that interactive systems are more powerful problem-solving engines than algorithms is the basis for a new paradigm for computing technology built around the unifying concept of interaction.

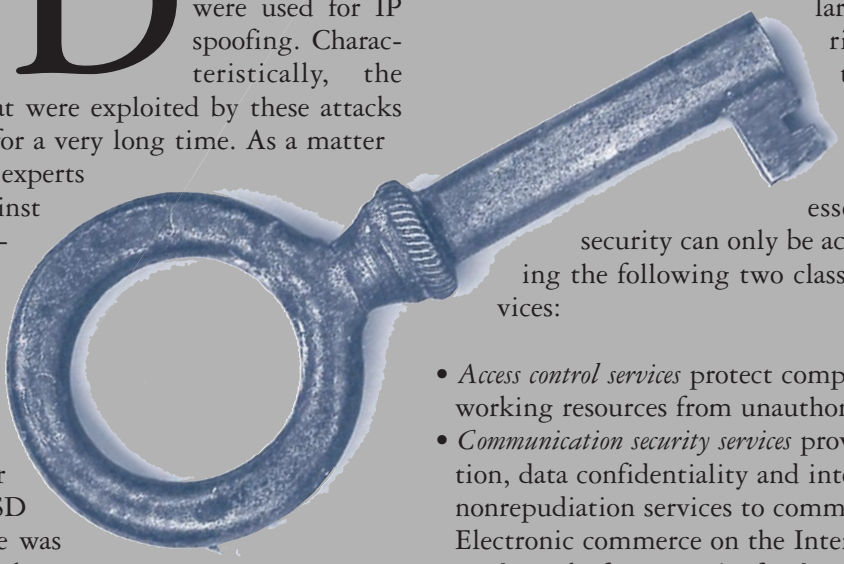
The emerging use of the TCP/IP communications protocol suite for internetworking has led to a global system of interconnected hosts and networks that is commonly referred to as the Internet. During the last decade, the Internet has experienced a triumphant advance. Projections based on its current rate of growth suggest there will be over one million computer networks and well over one billion users by the end of the century. Therefore, the Internet is seen as the first incarnation of a national information infrastructure (NII) as promoted by the U.S. government.

But the initial, research-oriented Internet and its communications protocol suite were designed for a more benign environment than now exists. It could, perhaps, best be described as a collegial environment, where the users and hosts were mutually trusting and interested in a free and open exchange of information. In this environment, the people on the Internet were the people who actually built the Internet. As time went on, the Internet became more useful and reliable, and these people were joined by others. With fewer goals in common and more people, the Internet steadily twisted away from its original intent.

Today, the Internet environment is much less collegial and trustworthy. It contains all the dangerous situations, nasty people, and risks that one can find in society as a whole. In this new environment, the openness of the Internet has turned out to be a dou-

Security problems may still hound the Internet, but existing tools and firewall technologies can strengthen our protective measures.

ble-edged sword. Since its very beginning, but especially since its opening and commercialization in the early 1990s, the Internet has become a popular target to attack. In November 1988, Robert T. Morris, Jr. launched the Internet Worm that flooded thousands of hosts [10]. Since then, reports of security incidents, such as attempted and successful system intrusions and other exploitations of various weaknesses on host systems on the Internet, have grown exponentially. More recently, thousands of passwords were sniffed on the Internet, and sequence number guessing attacks were used for IP spoofing. Characteristically, the vulnerabilities that were exploited by these attacks had been known for a very long time. As a matter of fact, security experts are warning against passwords transmitted in the clear since the very beginning of (inter)networking, and Morris described sequence number attacks for BSD Unix 4.2 when he was with AT&T Bell Laboratories in 1985 [5].



Today, virtually everyone on the Internet is vulnerable, and the Internet's security problems are the center of attention, generating much fear throughout the computer and telecommunications industry. Concerns about security problems have already begun to chill the overheated expectations about the Internet's readiness for full commercial activity, possibly delaying or preventing it from becoming a mass medium for the NII or the global information infrastructure. Several studies have independently shown that many individuals and companies are abstaining from joining the Internet simply because of security concerns. At the same time, analysts are warning companies about the dangers of not being connected to the Internet.

In this conflicting situation, most will agree the Internet needs more and better security. In a workshop held by the Internet Architecture Board (IAB) back in 1994, scaling and security were considered to be the two most important problem areas for the Internet as a whole.

But security, particularly Internet security, are vague terms that may mean various things to different people. In essence, Internet security can only be achieved by providing the following two classes of security services:

- *Access control services* protect computing and networking resources from unauthorized use.
- *Communication security services* provide authentication, data confidentiality and integrity, as well as nonrepudiation services to communicating peers. Electronic commerce on the Internet or World-Wide Web, for example, fundamentally depends on communication security services being deployed on a large scale.

In this article, we focus on the various security techniques available to provide Internet security in terms of both access control and communication security services. In particular, we discuss firewall technology and the security protocols that have been proposed for the Internet, transport, and application layer. We

conclude with some analogies that help us better understand the advantages and disadvantages of the various techniques.[1]

Firewall Technology

In days of old, brick walls were built between buildings in apartment complexes so that if a fire broke out, it would not spread from one building to another. Quite naturally, these walls were called "firewalls."

Today, when a network is connected to the Internet, its users are enabled to reach and communicate with the outside world. At the same time, however, the outside world can reach and interact with the network. In this dangerous situation, one (or several) intermediate system(s) can be plugged between the network and the Internet to establish a controlled link, and to erect an outer security wall or perimeter. The aim of this perimeter is to protect the network from network-based threats and attacks, and to provide a single choke point where security and audit can be imposed. Again, these intermediate systems are called firewalls, or firewall systems [1, 4].

In short, a firewall builds a blockade between an internal network that is assumed to be secure and trusted, and another network, usually an external (inter)network, such as the Internet, that is not assumed to be secure and trusted. The general reasoning behind firewall usage is that without a firewall, a network's systems are more exposed to inherently insecure Internet protocols and corresponding services, as well as probes and attacks from hosts elsewhere on the Internet. In a firewall-less environment, network security is solely a function of each host on the network, and all hosts must, in a sense, cooperate to achieve a uniformly high level of security. The larger the network, the less manageable it usually is to maintain all hosts at the same level of security. As mistakes and lapses in security become more common, break-ins can occur not only as the result of complex attacks, but also because of simple errors in configuration and inadequately chosen passwords. Given that situation, a firewall is to prevent unwanted and unauthorized communication into or out of the network, and to allow an organization to enforce a network security policy on traffic flowing between its network and the Internet.

A firewall system usually consists of screening routers and proxy servers. A screening router is a multiported IP router that applies a set of rules to each incoming IP packet, and decides whether it is to be forwarded or not. The screening router filters IP packets, based on information that is available in packet headers, such as protocol numbers, source and destination IP addresses and port numbers, connection flags, and eventually some other IP options.

A proxy server is a server process running on a firewall system to perform a specific TCP/IP function as a proxy on behalf of the network users. A proxy server is, in essence, an application-layer gateway; a gateway from one network to another for a specific network application. The user contacts a proxy using a TCP/IP application, such as telnet or ftp, and the proxy server asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user identification and authentication information, the proxy contacts the remote host, and relays IP packets between the two communication points. The whole process can be made transparent to the users. The identification and authentication information that a user provides may be used for user-level authentication. In the simplest case, this information consists of a user identification and password. However, if a firewall is accessible from the Internet, it is recommended to use strong authentication mechanisms, such as one-time password or challenge-response systems.

The advantages of screening routers are simplicity and low (hardware) costs. The disadvantages are related to the difficulties in setting up packet filter rules correctly, the costs of managing screening routers, and the lack of user-level authentication. Router vendors are working on solutions for these problems. In particular, they are working on graphical user interfaces for editing packet filter rules. They have also come up with a standardized user-level authentication protocol to provide a Remote Authentication Dial-In User Service (RADIUS).

The advantages of proxy servers are user-level authentication, logging, and accounting. The disadvantages are related to the fact that for full benefit, an application-layer gateway must be built specifically for each application. This fact may severely limit the deployment of new applications. More recently, an all-in-one proxy package called SOCKS has become available. SOCKS basically consists of a proxy to be run on a firewall system, as well as a package of library routines to be linked into network application programs.

Screening routers and proxy servers are usually combined in hybrid systems, where screening routers mainly protect against IP spoofing attacks. The most widely deployed configurations are dual-homed firewalls, screened host firewalls, and screened subnet firewalls.

[1] A more comprehensive overview is given in *Internet and Intranet Security*, R. Oppliger, Artech House, 1998.

In spite of their wide deployment within the Internet community, firewall technology is still an emotional topic. Firewall advocates consider firewalls as important additional safeguards because they aggregate security functions in a single point, simplifying installation, configuration, and management. Many companies are using firewalls as corporate ambassadors to the Internet, and thus as storage repositories for public information about their programs, products, and services. From a U.S. vendor's point of view, firewall technology is interesting because it doesn't use cryptography, and can be exported accordingly. However, most of the firewall systems currently offered do support some sort of IP layer encryption, and must deal with U.S. export controls accordingly. Another interesting feature of the firewall technology is related to the fact that its use is not restricted to TCP/IP protocols or the Internet. Indeed, a similar technology can, in principle, be used in any packet-switched network, such as an X.25 or ATM network.

Firewall detractors are usually concerned about the difficulty of using firewalls, requiring multiple logins and other out-of-band mechanisms, and their interference with the usability and vitality of the Internet itself. They claim that firewalls foster a false sense of security, leading to lax security within the firewall perimeter. They also observe that many attacks are perpetrated by insiders, immune to any perimeter defense strategy. Similarly, a firewall will not prevent all problems with incoming data. If users import programs and execute them, the programs may include malicious code that either discloses sensitive information, or modifies and deletes them (or both). This problem is expected to become even more serious as the proliferation of Java and JavaScript applets, ActiveX controls, and corresponding browsers grows continually. Another disadvantage of the firewall technology is that relatively few vendors have offered turnkey solutions in the past. Most firewalls were somewhat hand-built by site administrators. Nevertheless, this is about to change very rapidly.[2]

In spite of this controversial discussion, firewall advocates and detractors both agree that firewalls should not be regarded as a substitute for careful security management within a perimeter. Firewalls are a fact of life in the Internet today. They have been constructed for pragmatic reasons by organizations interested in a higher level of security than may be possible without them. Consequently, a firewall is

[2]A comparative overview of firewalls products is given in the Computer Security Institute's Spring '96 *Computer Security Journal*.

Glossary of Terms

AH	Authentication Header
BTP	Back-traffic Protection
CA	Certification Authority
CBC	Cipher Block Chaining
CLNP	Connectionless Network Protocol
DES	Data Encryption Standard
ESP	Encapsulating Security Payload
GSS-API	Generic Security Services API
IANA	Internet Assigned Numbers Authority
IESG	Internet Engineering Steering Group
IETF	Internet Engineering Task Force
IKMP	Internet Key Management Protocol
IPC	Interprocess Communications
IPRA	Internet Policy Registration Authority
IPSEC	IETF's Internet Protocol Security protocol
IPSP	IP Security Protocol
ISAKMP	Internet Security Association and Key Management Protocol
MKMP	Modular Key Management Protocol
NLSP	Network Layer Security Protocol (also Integrated or I-NLSP)
PCA	Policy Certification Authority
PCT	Private communications technology
PEM	Privacy Enhanced Mail
PFS	Perfect-forward secrecy
PKI	Public key infrastructure
RADIUS	Remote Authentication Dial-In User Service
SEPP	Secure Electronic Payment Protocol
SET	Secure Electronic Transactions
SDNS	Secure Data Network System
SHA	Secure Hash Algorithm
S-HTTP	Secure Hypertext Transfer Protocol
SKEME	Secure Key Exchange Mechanism
SKIP	Simple Key Management for Internet Protocols
SNP	Secure network programming
SP3	Security Protocol 3
SSH	Secure shell
SSL	Secure Socket Layer
STS	Station-to-Station protocol
STT	Secure Transaction Technology
S/WAN	Secure wide-area network
TLI	Transport Layer Interface
TLS	Transport Layer Security
TLSP	Transport Layer Security Protocol

not a magic bullet for all network security problems; rather, it is one of the many pieces in a network security policy or strategy.

Here, we focus on communication security services that may be provided by cryptographic protocols. Such protocols have been proposed for the Internet, transport, and application layer. The question of what layer is best suited to provide security services has a long tradition among network practitioners, and the OSI security architecture is not too specific about it either.

Internet Layer Security

The idea of having a standardized network or Internet layer security protocol is not new, and several proposals have been made during the last decade. For example, the Security Protocol 3 (SP3) is a network layer security protocol that has been proposed by the U.S. National Security Agency (NSA) and the National Institute of Science and Technology (NIST) as part of the Secure Data Network System (SDNS). The Network Layer Security Protocol (NLSP) is a security protocol standardized by the International Organization for Standardization (ISO) for the Connectionless Network Protocol (CLNP). The Integrated NLSP (I-NLSP) is a NIST proposal to provide security services for both IP and CLNP. SwIPe is yet another Internet layer security protocol proposed and prototyped by Ioannidis and Blaze. All these proposals are more alike than they are different. In fact, they all use IP encapsulation as their enabling technique. In essence, plaintext packets are encrypted and enclosed in outer IP headers that are used to route the encrypted packets through the Internet. At the peer systems, the outer IP headers are stripped off, and the packets are decrypted and forwarded to their final destinations.

The Internet Engineering Task Force (IETF) has chartered an Internet Protocol Security protocol (IPSEC) working group to standardize both an IP Security Protocol (IPSP) and a corresponding Internet Key Management Protocol (IKMP).

The primary objective of IPSP is to make available cryptographic security mechanisms to users who desire security. The mechanisms should work for both the current version of IP (IPv4) and the new version of IP (IPng or IPv6). The mechanisms should be algorithm-independent, in that the cryptographic algorithms can be altered without affecting the other parts of an implementation. In addition, the mechanisms should be useful in enforcing different security policies, but avoid adverse impacts on users who do not employ them. According to these requirements, the IPSEC working group has come up with a specification that is based on two cryptographic security mechanisms: the Authentication Header (AH) and the Encapsulating Security Payload (ESP). In short, the AH is to provide authenticity and integrity to IP packets, whereas the ESP is to provide confidentiality.

The IP AH refers to a message authentication code (MAC) that is computed prior to sending an IP packet. The sender uses a cryptographic key to compute the AH, and the receiver uses the same or another key to verify it. The keys are the same if the sender's and receiver's computations are based on secret key cryptography; keys are different if their computations are based on public key cryptography. In the second case, the AH mechanism can additionally provide nonrepudiation services. In fact, certain fields change in transit, such as the time-to-live field in IPv4 or the hop limit field in IPv6, must be omitted from the AH computation. RFC 1828 has originally specified the use of keyed MD5 in envelope mode for AH computation and verification [11]. Meanwhile, both MD5 and the envelope mode have been criticized as being too weak, and alternatives are being proposed.

The basic idea of the IP ESP is to encapsulate either an entire IP packet (tunnel mode) or only the upper-layer protocol data (transport mode) inside the ESP, and to encrypt most of the ESP. In the tunnel mode, a new IP header is appended in plaintext to the now encrypted ESP. The new IP header is used to route the IP packet through the Internet. The receiver removes and discards the plaintext IP header and options, decrypts the ESP, processes and removes the ESP header, and processes the (now decrypted) original IP packet or upper-layer protocol data as per the normal IP protocol specifications. RFC 1827 specifies the format of the ESP, whereas RFC 1829 specifies the use of the Data Encryption Standard (DES) in Cipher Block Chaining (CBC) mode for ESP encryption and decryption. Although other algorithms and modes may be used as well, there are export and import controls in some countries that must be considered. There are even countries that regulate the use of encryption for private use.

The AH and ESP mechanisms may be used together or separately. In either case, it should be kept in mind that none of the mechanisms are able to provide protection from traffic analysis. It is not clear whether meaningful protection from traffic analysis can be provided economically at the Internet layer, and it also appears that only few Internet users are actually concerned about traffic analysis.

In August 1995, the Internet Engineering Steering Group (IESG) approved the RFCs related to

IPSP as proposed standards for the Internet standards track. In addition to RFC 1828 and RFC 1829, there are two experimental RFCs that specify the use of the Secure Hash Algorithm (SHA) instead of MD5 (RFC 1852) and Triple DES instead of DES (RFC 1851) for the AH and ESP mechanisms.

In the simplest case, IPSP is used with manually configured keys. However, when it comes to a wide deployment of IPSP, a standardized key management protocol for the Internet is required. This key management protocol is to specify a method for the management of cryptographic keys as required by security associations in IPSP.

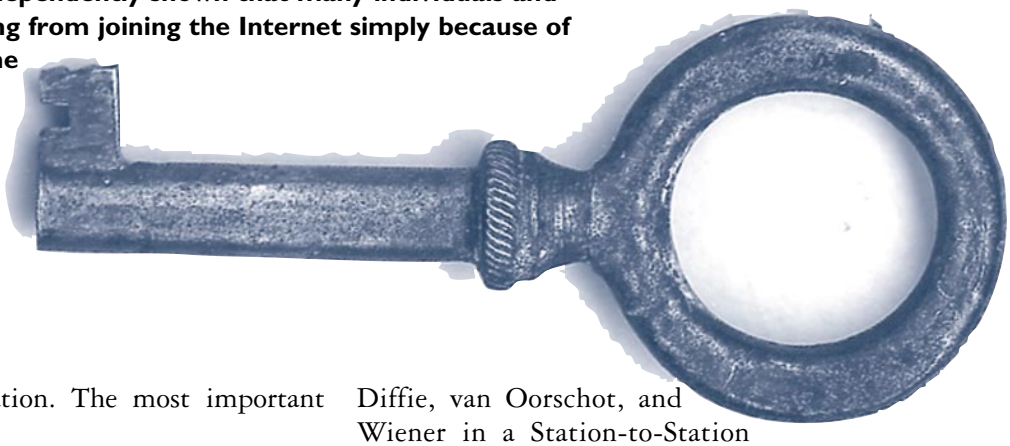
Consequently, the IPSEC working group is also standardizing an Internet Key Management Protocol (IKMP) and several protocols have been submitted to the group for consideration. The most important proposals are:

- The Modular Key Management Protocol (MKMP) from IBM
- The Simple Key-Management for Internet Protocols (SKIP) from Sun Microsystems
- The Photuris[3] Key Management Protocol from Phil Karn
- The Secure Key Exchange Mechanism (SKEME) from Hugo Krawczyk
- The Internet Security Association and Key Management Protocol (ISAKMP) from the NSA, and
- The OAKLEY Key Determination Protocol from Hilarie Orman

Again, the protocol proposals are more alike than they are different. Except for the MKMP, they all require an existent and fully operable public key infrastructure (PKI). The MKMP does not have this requirement, because it assumes two parties to already share a master key that may be distributed manually. SKIP requires Diffie-Hellman certificates, whereas all other protocol proposals require RSA cerificates.

In September 1996, a decision was made to make the OAKLEY protocol in ISAKMP framework the mandatory key management approach for IPSEC, and to make SKIP an elective standard for IPv4 and IPv6 implementations. Several vendors are implementing the combined ISAKMP/OAKLEY approach today. The basic idea of Photuris and Photuris-like protocols is to use a Diffie-Hellman key exchange for every new session key, and to use a subsequent signature exchange to authenticate the Diffie-Hellman parameters and to protect against "man-in-the-middle" attacks accordingly. This combination was originally proposed by Diffie, van Oorschot, and Wiener in a Station-to-Station (STS) protocol. Photuris additionally uses a so-called "cookie" exchange to provide anti-clogging, meaning to protect against denial of service attacks.

Several studies have independently shown that many individuals and companies are abstaining from joining the Internet simply because of security concerns. At the same time, analysts are warning companies about the dangers of not being connected to the Internet.



The fact that Photuris and Photuris-like protocols use a Diffie-Hellman key exchange for every new session key provides back-traffic protection (BTP) and perfect-forward secrecy (PFS). In essence, this means that whenever an attacker breaks a long-term private key, such as a RSA key in the case of Photuris or a Diffie-Hellman key in the case of SKIP, all the attacker can do is masquerade, thus pretending to be the entity who owns the broken key. But the attacker is not necessarily able to decrypt either past or future messages sent or received by that entity.

Note that SKIP does not provide BTP and PFS. Yet, it uses a Diffie-Hellman key exchange, but this key exchange is done implicitly, meaning that two entities who know each other's long-term Diffie-Hellman public key in a certified form implicitly share a master key. This master key can be used to derive a key to encrypt a packet key, and this packet

[3]"Photuris" is the Latin name for the firefly, and "Firefly" is the name for a classified key exchange protocol designed by the National Security Agency's STU-III secure telephone.

key can then be used to encrypt an IP packet. If a long-term Diffie-Hellman key gets lost or broken, any communication that has been or will be protected with a key that is protected by that Diffie-Hellman key may be decrypted accordingly. Moreover, SKIP is stateless in a sense that it is not based on security associations. Each IP packet may be encrypted and decrypted individually, eventually with a different key.

The fact that SKIP does not provide BTP and PFS has been criticized within the IPSEC working group, and the protocol has been extended to provide it. However, the resulting SKIP protocol version is actually a trade-off between the provision of BTP and PFS on the one hand, and the statelessness of the protocol on the other hand. The SKIP protocol that provides BTP and PFS is, in fact, very similar to Photuris and Photuris-like protocols, except for the main difference that SKIP (still) requires Diffie-Hellman certificates. It should be noted at this point that any security protocol that requires RSA certificates may be easier to implement and deploy within the Internet today.

Most implementations of IPSP and corresponding key management protocols are based on Unix systems. Probably the main reason for this fact is that any IPSP implementation should be entwined in the source code of the corresponding protocol stack, and that this source code is available for Unix systems. However, if security protocols are to be used and widely deployed within the Internet, they must be available for MS-DOS and Windows systems, too. An immediate problem that one faces when trying to implement Internet layer security protocols for these systems is due to the fact the source code of the most widely deployed TCP/IP implementation (which is PC/TCP from FTP Software) is not publicly available. In order to overcome this problem, Wagner and Bellovin have implemented an IPSEC module that works as a device driver, entirely below IP [12].

The major advantage of Internet layer security is transparency, meaning that security services can be provided without requiring changes to applications, to any other communication layers, or to network components that do not need security at all. A major disadvantage, however, is the Internet layer, in general, does not discriminate between packets that belong to different processes and corresponding associations. It uses the same encryption keys and access policies to all packets destined to the same address. This may not provide the function desired, and may be costly in performance, too. In addition to this host-oriented keying, RFC 1825 also allows (and even recommends) user-oriented keying, where different connections may get different keys. However, user-oriented keying requires some major modifications in the corresponding operating system kernels.

While IPSP specification is mostly done, the key management situation is somehow more fluid and more work needs to be done here. One important problem that has not been addressed so far is key distribution in multicast environments, such as the Internet Multicast Backbone (MBone) or IPv6 networks.

In short, the Internet layer is well suited to provide security services on a host-to-host basis. The corresponding security protocols can be used to set up secure IP tunnels and virtual private networks within the Internet. For example, it is simple and straightforward to enhance a firewall system with the ability to encrypt and decrypt IP packets. In fact, several vendors have done so, and RSA Data Security has started an initiative to promote multivendor virtual private networks among firewall and TCP/IP software vendors. The initiative is called S/WAN (Secure WAN). It aims at making recommendations and additions to Internet layer security protocol standards.

Transport Layer Security

In Internet application programming it is common to use a generalized interprocess communications facility (IPC) to work with different transport layer protocols. Two popular IPC interfaces are BSD sockets and the transport layer interface (TLI), found on System V Unix variants.

One idea that one may think of first when trying to provide security services within the Internet is to enhance an IPC interface, such as BSD sockets, with the ability to authenticate peer entities, as well as to exchange a secret key that can be used to encrypt and decrypt a data stream transmitted between communicating peers. Netscape Communications has followed this approach and has specified a Secure Sockets Layer (SSL) on top of a reliable transport service, such as provided by TCP/IP. SSL version 3 (SSL v3) was specified in December 1995. It basically consists of two protocols:

- The SSL record protocol deals with fragmentation, compression, data authentication and encryption of messages provided by applications. SSL v3 provides support for keyed MD5 and SHA for data authentication, as well as RC4 and DES for data encryption. The keys that are used for data authentication and encryption are negotiated

by the SSL handshake protocol.
- The SSL handshake protocol deals with the exchange of protocol version numbers and supported cryptographic algorithms, as well as (mutual) authentication and key exchange. SSL v3 provides support for Diffie-Hellman key exchange, an RSA-based key exchange mechanism, and another key exchange mechanism implemented in the Fortezza chip.

Netscape Communications has made publicly available a reference implementation of SSL (which is called SSLref). There is also a freely available SSL implementation called SSLeay. Both SSLref and SSLeay can be used to incorporate SSL functionality into arbitrary TCP/IP applications. The Internet Assigned Numbers Authority (IANA) has assigned various port numbers to applications that incorporate SSL. For example, port number 443 has been assigned for HTTP with SSL (https), 465 for SMTP with SSL (ssmtp), and 563 for NNTP with SSL (snntp).

Microsoft has proposed an updated version of SSL, version 2, called PCT (Private Communication Technology). At least with regard to the record format they use, SSL and PCT are very similar. The main difference between them is related to the way they use the most significant bit in the version number field (in PCT this bit is set to 1). Thus, an Internet server can dynamically decide whether a request refers to SSL (bit is set to 0) or PCT (bit is set to 1). Support, therefore, can be provided for both protocols.

In April 1996, the IETF has chartered a Transport Layer Security (TLS) working group to specify a Transport Layer Security Protocol (TLSP) that can be submitted to IESG for consideration as a proposed standard. TLSP will greatly resemble SSL version 3.

We have seen the major advantage of Internet layer security is transparency, meaning that security services can be provided without requiring changes to the applications. This is not true for transport layer security. In principle, any TCP/IP application that is to make use of a transport layer security protocol, such as SSL or PCT, must be modified to incorporate the corresponding functionality and to use a (slightly) different IPC interface. Thus, the main disadvantage of transport layer security is that modifications (or additions) are required for both the transport layer IPC interface and the application programs. However, compared to modifications required to provide Internet or application layer security these modifications are rather small. Another disadvantage is due to the fact that UDP-based communication is difficult to secure at the transport layer. The main advantage of transport layer security protocols compared to network layer security protocols is due to the fact that they provide security services on a process-to-process basis (instead of a host-to-host basis). This improvement can be taken one step further by providing security services on the application layer.

Application Layer Security

One thing that should be kept in mind (and considered with care) is the fact that network (or transport) layer security protocols allow to attach security properties to network (or transport) data channels between hosts (or processes). Essentially, this means that either authentic or authentic and confidential data channels can yet be built between hosts (or processes), but that it is not possible to differentiate among the security requirements of individual documents that are transmitted on these channels. For example, if a host uses Internet layer security protocols to build secure IP tunnels to other hosts, every IP packet sent to one of these hosts will automatically be encrypted. Similarly, if a process uses transport layer security protocols to build secure data channels to other processes, every message sent to one of these processes will automatically be encrypted.

If one really wanted to take into account the different security requirements of individual documents, one would have to go for application layer security. Providing security services on the application layer is in fact the most flexible way to handle individual security requirements. For example, an email system may need to digitally sign specific portions of outgoing messages. Security functions provided at the lower layers would not, in general, know anything about the structure of the messages, or which portions should be signed. The only layer that is able to provide this security service is in fact the application layer.

In general, there are several possibilities to provide security services on the application layer, and one possibility that one might think of first is to modify each application (and application protocol) accordingly. This approach has been followed for some important TCP/IP applications. In RFCs 1421 to 1424, the IETF has specified Privacy Enhanced Mail (PEM) to provide security services for SMTP-based email systems [4]. For several reasons, the acceptance of PEM within the Internet community has turned out to be rather slow, and one of the main reasons is due to the fact that PEM depends on an existant and fully operable PKI. The PEM PKI is hierarchically structured and consists of three levels:

- An Internet Policy Registration Authority (IPRA) on the top level
- Policy Cerification Authorities (PCAs) on the second level
- Cerification Authorities (CAs) on the third level

The creation of a PKI that conforms to the PEM specification is also a political process, as it requires different parties to agree on common points of trust. Unfortunately, history has shown that political processes always require some time, and as an intermediate step Phil Zimmermann has developed a software package called Pretty Good Privacy (PGP). PGP conforms to most parts of the PEM specification, but doesn't require a PKI to be in place. Instead, it uses a distributed trust model, meaning every user decides what other users he or she is willing to trust. Thus, instead of promoting a global PKI, PGP has its users establish their own web of trust. One problem that immediately arises in a distributed trust model is related to key revocation.

S-HTTP is a security-enhanced version of the Hypertext Transfer Protocol (HTTP) that is used on the Web. Designed by Enterprise Integration Technologies, S-HTTP provides security at the document level, thus every document can be marked as private and/or signed. The algorithms to encrypt or sign documents are negotiated by the corresponding senders and receivers. S-HTTP provides support for several one-way hash functions, such as MD2, MD5, and SHA, secret key cryptosystems, such as DES, Triple-DES, RC2, RC4, and CDMF, and digital signature systems, such as RSA and DSS.

There is currently no agreed standard for Web security. Such a standard would have to be defined by the WWW Consortium, the IETF, or any other relevant standardization organization. While the formal process of standardization is lengthy, and could run into years, all the standards groups recognize the importance of securing the Web. S-HTTP and SSL follow different approaches to provide security for the Web. While S-HTTP marks individual documents as private or signed, SSL mandates the data channel used for communication between the corresponding processes as private and/or authenticated. Terisa's SecureWeb toolkits can be used to incorporate security functions into arbitrary Web applications. The toolkits include functionality provided by cryptographic libraries from RSA Data Security and provide support for both S-HTTP and SSL.

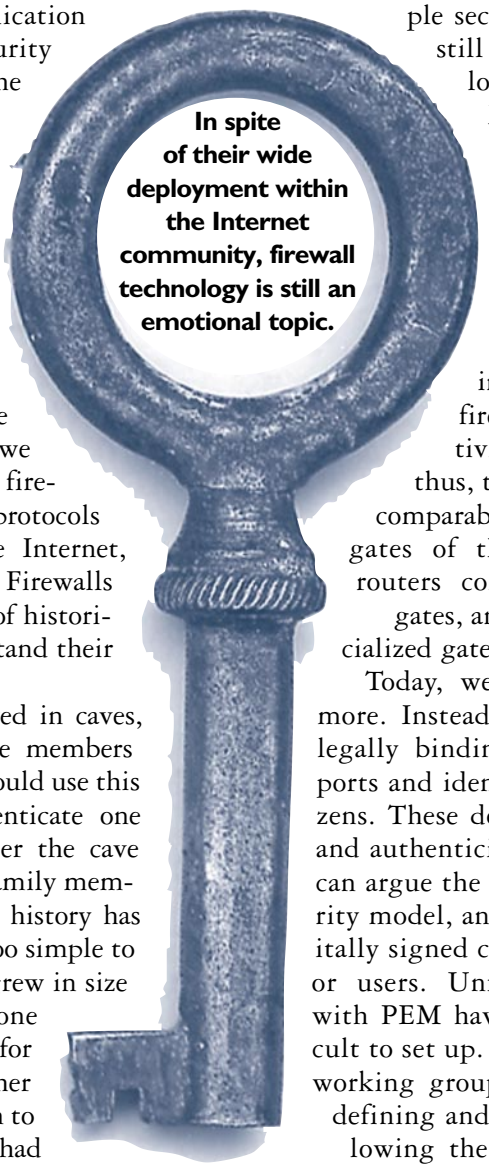
Another important application is electronic commerce in general, and credit card transactions in particular. In order to secure credit card transactions over the Internet, MasterCard (together with IBM, Netscape Communications, GTE, and CyberCash) has specified a Secure Electronic Payment Protocol (SEPP), and Visa International and Microsoft (together with some other companies) have specified a similar Secure Transaction Technology (STT) protocol. Meanwhile, MasterCard, Visa International and Micrososft have agreed to cooperate in providing secure credit card transactions over the Internet. They released the specification of a corresponding Secure Electronic Transactions (SET) protocol that specifies a way in which a credit card holder can use a credit card to make payments over the Internet. There is a certification infrastructure running in the background, and this infrastructure is providing support for X.509 certificates.

Probably the main problem related to all security-enhanced applications mentioned here is the fact that modifications must be specified (and implemented) individually for each application. Thus, it would be better to have a more generic approach. One step in that direction is the secure shell (SSH) developed by Tatu Yloenen from the University of Helsinki. SSH allows its users to securely log on to remote hosts, execute commands, and transfer files. It implements a key exchange protocol, as well as host and client authentication protocols. SSH is freely available for most Unix systems in use today, and a commercial version of SSH is marketed by Data Fellows.

Taking the idea of SSH one step further leads to authentication and key distribution systems. In essence, an authentication and key distribution system provides an application programming interface (API) that can be used by arbitrary network applications to incorporate security services, such as authentication, data confidentiality and integrity, access control, and nonrepudiation services. There are several authentication and key distribution systems available today, with Kerberos (V4 and V5) from MIT, KryptoKnight or Network Security Program from IBM, SPX from DEC, and The Exponential Security System (TESS) from the University of Karlsruhe being the most widely deployed examples. Some systems have even seen modifications and extensions. For example, SESAME and OSF DCE extend Kerberos V5 to additionally provide access control services, and Yaksha extends Kerberos V5 to additionally provide nonrepudiation services. (See [6, 8] for more details.)

One question that often arises in the context of authentication and key distribution systems is related to their poor deployment within the Internet. One reason for this astonishing fact is the use

of an authentication and key distribution system still requires applications to be modified. Taking this into account, it is important for an authentication and key distribution system to provide a standardized security API. It should not be necessary for a developer to modify an application for every single authentication system that might be in use. Thus, one of the main results in the area of authentication system design has been the specification of a standardized security API called Generic Security Services API (GSS-API). The GSS-API (v1 and v2) may still provide an API that is too technical for an application developer who might not be a security expert, however, researchers at the University of Texas at Austin have developed an interface for Secure Network Programming (SNP) that is layered on top of GSS-API.



Conclusions

In this article we have focused on the various techniques that are available and can be used to provide Internet security. In particular, we have overviewed and discussed the firewall technology and the security protocols that have been proposed for the Internet, transport, and application layers. Firewalls are omnipresent today, and a pair of historical analogies help us better understand their roles within the Internet [7].

Our Stone Age predecessors lived in caves, each inhabited by a family whose members knew each other quite well. They could use this knowledge to identify and authenticate one another. Someone wanting to enter the cave would have to be introduced by a family member trusted by the others. Human history has shown that this security model is too simple to work on a large scale. As families grew in size and started to interact with one another, it was no longer possible for all family members to know all other members of the community, or even to reliably remember all persons who had been introduced to them.

In the Middle Ages, our predecessors lived in castles and villages surrounded by town walls. The inhabitants were acquainted with each other, but this web of knowledge was not trusted. Instead, identification and authentication, as well as authorization and access control, were centralized at a front gate. Anyone who wanted to enter the castle or village had to pass the front gate and was thoroughly checked there. Those who managed to pass the gate were implicitly trusted by all inhabitants. But human history has shown that this security model doesn't work either. For one thing, town walls don't protect against malicious insider attacks; for another, the use of town walls and front gates doesn't scale easily. Many remnants of medieval town walls bear witness to this lack of scalability.

Referring to this pair of analogies, the Internet has just entered the Middle Ages. The simple security model of the Stone Age still works for single hosts and local area networks. But it no longer works for internets in general, and the Internet in particular. As a first (and let's hope intermediate) step, firewalls have been erected at the border gateways to the Internet. Because they are capable of selectively forwarding or dropping IP packets, firewalls also restrict the connectivity of the Internet as a whole, thus, the Internet's firewalls are thus comparable to the town walls and front gates of the Middle Ages (screening routers correspond to general-purpose gates, and proxy servers to highly specialized gates).

Today, we don't see town walls anymore. Instead, countries issue official and legally binding documents, such as passports and identification cards, to their citizens. These documents certify the identity and authenticity of particular citizens. One can argue the Internet needs a similar security model, and that a PKI has to issue digitally signed certificates to Internet citizens or users. Unfortunately, the experiences with PEM have shown that a PKI is difficult to set up. There are currently two IETF working group chartered with the task of defining and setting up a PKI; one is following the X.509 track and another is trying to do it simpler.

Once a PKI is put in place, the deployment of cryptographic security protocols can start. However, the question remains: What layer is best suited to provide security services? A possible answer is there is no single best layer, but that each application has to make its choices based on its security require-

ments. For example, an application that requires nonrepudiation services may go for application layer security, whereas an application that requires IP tunneling between mobile stations and corporate firewalls may be well served with Internet layer security. It is even possible and likely that an application can go for different layers for different security services. For example, it can make a lot of sense to use SSL to provide data confidentiality services to Web applications, and S-HTTP to provide nonrepudiation services to the same applications. In this case, an appropriate security protocol suite would be S-HTTP over SSL.

Another analogy helps us better understand the choices an application developer has to make with regard to the provision of security services. Let's assume you have some valuable goods to transport, and you can build a railway system. The question that arises immediately is how to secure the goods. There are several possibilities to address this question. The two most obvious possibilities are to secure the railway system as a whole, or to secure the goods that are transported on the railway system. If you opt to secure the railway system, then, in principle, you are going for network layer security, whereas if you opt to secure the goods you are going for application layer security. In the first case, you have to modify the railway system but can leave the goods as they are, whereas in the second case, you have to modify the goods but can leave the railway system as it is. There is yet another possibility somewhere in between: You may think of securing some parts of the railway system only and use these parts to transport the most valuable goods, whereas you use the unsecured parts to transport goods that are not valuable. In this case, however, you have to decide for each good on what parts of the system it should be transported. This possibility is essentially what happens if you opt for transport layer security. You have to provide a transport layer interface that allows the deployment of either a secure or a nonsecure transport system. It is then up to the application to decide which transport system to use, and the ability to make this decision requires applications to be modified.

Let's go back to the introductory remarks that have compared the Internet with an information superhighway. This comparison can be used to come up with yet another analogy that helps us better understand the role of security in general. It is common to use several controls to provide security on a real highway system. We have drivers pass an examination to get a driver licence, and we have vehicles checked periodically to see whether they still conform to their technical specifications. We also have a set of rules that specify the correct behavior on the highway system, and a police to enforce it. But in spite of all these controls we (still) don't believe it is possible to come up with a highway system that is completely safe and secure. The same insight should be achieved with regard to Internet security. It is not possible to achieve Internet security in a complete sense. Nevertheless, we should try to come up with an architecture for the Internet that provides as much security as possible. We have yet to see effective, integrated approaches to provide Internet security. Hopefully, this will change in the future. C

Acknowledgments

I would like to thank Boris Bremer, Jean-Luc Nottaris, and David Wagner for their review activities, and Dieter Hogrefe, Hansjuerg Mey, and Kurt Bauknecht for their encouragement and support.

References

1. Chapman, D. and Zwicky, E. *Internet Security Firewalls*. O'Reilly, Sebastopol, Calif., 1995.
2. Cheswick, W., and Bellovin, S. *Firewalls and Internet Security: Repelling the Wiley Hacker*. Addison-Wesley, Reading, Mass., 1994.
3. IEEE. Local and Metropolitean Area Networks: Interoperable LAN/MAN Security (SILS). IEEE Std 802.10, 1990.
4. Kent, S. Internet privacy enhanced mail. *Commun. ACM 36*, 8 (Aug. 1993), 48–60.
5. Morris, R. A Weakness in the 4.2BSD UNIX TCP/IP Software. Computing Science Technical Report No. 117, AT&T Bell Laboratories, Murray Hill, NJ, Feb. 1985.
6. Oppliger, R. Authentication and key distribution in computer networks and distributed systems. In *Communications and Multimedia Security*. R. Posch, Ed. Chapman & Hall, London, UK, 1995.
7. Oppliger, R. Internet Kiosk: Internet security enters the Middle Ages. *IEEE Comput. 28*, 10 (Oct. 1995), 100–101.
8. Oppliger, R. Authentication Systems for Secure Networks. *Artech House*, Norwood, Mass. 1996.
9. Schneier, B. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley, New York, 1994.
10. Spafford, E. The Internet worm: Crisis and aftermath. *Commun. ACM 32*, 6 (June 1989), 678–688.
11. Tsudik, G. Message authentication with one-way hash functions. *ACM Computer Communication Review 22*, 5 (1992), 29–38.
12. Wagner, D., and Bellovin, S. A "bump in the stack" encryptor for MS-DOS systems. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security* (Feb. 1996).

Rolf Oppliger (oppligerr@acm.org) is with the Swiss Federal Agency for Information Technology and Systems, and at the universities of Berne and Zurich.