UnixWorld Online: Internet Security: Column No. 001

Sequence Number Attacks

By Rik Farrow

Kevin Mitnick's alleged attack on Tsutomu Shimomura's Computers used a vulnerability in TCP/IP and mistaken trust.

Questions regarding this article should be directed to the author at *rik@spirit.com*.

December 25, 1994 found Tsutomu Shimomura, a computational physicist for the San Diego Supercomputer Center, on his way to the Sierra Nevadas to go skiing. He had left his personal network of computers running at his beach cottage in Del Mar, just north of San Diego. Perhaps it is fortunate for us he did so.

Just after two o'clock in the afternoon, Shimomura's home systems were probed, then successfully attacked using something new in Internet attacks, sequence number guessing. Shimomura also works as a security expert, which made his systems both desirable targets for attack, and allows us to understand in detail what happened. Because, unlike most networks, Shimomura was using `tcpdump` to monitor traffic incoming from his Internet connection, and routinely sent his logs to an offsite location.

Sequence number guessing is not really new. Steve Bellovin, a researcher at Bell Labs, and co-author of the *Firewalls and Internet Security* book (Addison-Wesley, 1994, ISBN 0-201-63357-4), included details of an attack scenario in his 1989 paper entitled ''Security Problems in the TCP/IP Protocol Suite''. But the Christmas day attack is the first known use of the technique.

To better understand what happened, it helps to understand a little about how TCP (Transport Control Protocol) works. TCP is used for establishing bidirectional streams, like those used for remote terminal connections (established with `telnet` or `rlogin` utilities). TCP is also used for transferring large amounts of data, for example with FTP or connecting to a Web server.

TCP provides a reliable connection. That is, unlike most other parts of the Internet Protocol suite (such as ICMP, Internet Control Message Protocol, or UDP, User Datagram Protocol), TCP establishes a connection between the local and remote site. Once the connection has been successfully established, groups of bytes of data are acknowledged by sending a sequence number back to the sending site. If the sending site does not receive an acknowledgement quickly enough, it will resend the data. If the sending site has resent the same data several times unsuccessfully, it will send an error to the application saying that the connection has been broken.

The sequence number is used to acknowledge receipt of data. At the beginning of a TCP connection, the client sends a TCP packet with an initial sequence number, but no acknowledgement (there can't be one yet). If there is a server application running at the other end of the connection, the server sends back a TCP packet with its own initial sequence number, and an acknowledgement: the initial sequence number from the client's packet plus one. When the client system receives this packet, it must send back its own acknowledgement: the server's initial sequence number plus one. Thus, it takes three packets to establish a TCP connection (see Part A of Figure 1 which shows the time-line diagram.

There's more to TCP, of course. You won't learn all about TCP in this short article (try Doug Comer's book *Internetworking with TCP/IP, Volumme 1, Principles, Protocols, and Architecture. Second Edition* (Prentice Hall, 1991 ISBN 0-13-468505-9) or W. Richard Steven's *TCP/IP Illustrated, Volume 1* (Addison-Wesley, 1993, ISBN 0-201-63346-9). For now, it's important to understand that TCP packets include flag bits that get set to indicate conditions. When you read Shimomura's account of the attack, he makes reference to several flags.

The SYN flag (shown as a capital ''S'' in `tcpdump` command output) indicates the initiation of a connection, and that an initial sequence number is included. When the first packet is sent from the client, only the SYN flag is set. When the server responds, both the SYN flag and the ACK flag, indicating that a valid acknowledgement is included, are set. From then on, the ACK flag will be set, showing that each packet includes an acknowledgement of a received packet.

The PUSH (shown as a capital ''P'') flag means that the data in this packet should be pushed to the application, rathered than queued until more data arrives. The RESET (''R'') flag tells TCP to break (reset) the connection, and is sent when a client attempts to connect to a server application that is not running. In the attack, RESETs are used to close the half-open connections used to keep the server busy.

The FIN bit (''F'') is used to close a connection. Each end of the connection sends a packet with the FIN flag, which must be acknowledged, so four packets are used to close a TCP connection. Of course in the attack you won't see two FIN packets, because the attacker never sees the responses from the target system, the X terminal.

With this background, you are ready to read Shimomura's own description, if you haven't already. Essentially, the attack begins when several probes were launched from `toad.com` (a site registered by Nebula Consulting). Although I don't know this for sure, it is likely that `toad.com` had been broken into previously using other techniques.

The probes, using `finger`, `showmount`, and `rpcinfo` (similar to probes from SATAN, but not automatic) apparently helped the attacker to determine a trust relationship between Shimomura's X terminal (actually another workstation used as an X terminal), and a local server. This was the real weakness exploited in the attack. Shimomura's systems trusted one another, using the trust mechanism exploited in the ''r'' commands like `rsh` and `rcp`. Although convenient, and safe behind a strong perimeter defense, trust has been used to break into systems for many years. The November 1988 Internet Worm exploited trust in its automated attacks.

In the next phase of the attack, thirty TCP SYN packets are sent to the `rlogin` port of Shimomura's server. These packets come from an unused Internet address, and their initial sequence numbers are incremented by one instead of the more common 128,000. The purpose of these packets are to fill the queue on the server with half-open connections, so when the spoofing begins, the server won't be able to respond to the packets being sent as acknowledgements from the X terminal.

Next, a system at Loyola University of Chicago (`apollo.it.luc.edu`) was used to probe the X terminal. Once again, a synthetic series of TCP packets (initial sequence numbers incremented by one) gets sent, but this time responses get sent back to a real site. It is the responses that the attacker is after, because each response contains an initial sequence number from the X terminal. In Shimomura's paper `tcpdump` labels the responses with `x-terminal.shell > apollo.it.luc.edu.1000 S`, indicating a packet with the SYN flag set. If we subtract the first initial sequence number from the second, 2021952000-2021824000, we get 128,000, a pattern that holds for all twenty probes. So now the attacker knows that the next initial sequence number will be 128,000 greater than the previous one.

The stage is now set for abusing the trust between the X terminal and the server. The attacker generates packets that appear to come from the server to open a TCP connection with the X terminal `rshell` daemon. The X terminal sends an acknowledgement back to the server, but this acknowledgement gets stuck in the queue. Next, the attacker generates the acknowledgement the server might have sent (if it had really been the server initiating the connection). Now there is an open TCP connection from the X terminal to the server, which is being spoofed by the server. Figure 1B shows how the packets actually traveled.

Because the X terminal trusts the server, the attacker, masquerading as root, sends the command `"echo + + >> /.rhosts"` to the X terminal, extending trust to ANY root user with access to this system. Then the attacker closes the connection by sending a packet with FIN set, and acknowledging the FIN packet never seen from the X terminal. Finally, thirty RESETs are sent to the server to clear its queue. At this point, the server would send a RESET to the X terminal to close the `rshell` connection it never made (the one spoofed by the attacker), but it is now too late. The attacked can now log in as root on the X terminal using `rlogin`.

There was more to the attack, but I'd like to save taking over TCP connections for another day. There is also a CERT advisory about this attack, which you can read.

Edited by Becca Thomas / Online Editor / UnixWorld Online / beccat@wcmh.com