6.857 Computer and Network Security	November 13, 2001

Lecture Notes 19 : Practical Insecurity

Lecturer: Roger Dingledine, Ron Rivest *Scribe: Cody/Cotler/Hastings/Langer*

1 Outline

- Practical Insecurity[1]
- Video
- Brazilian Voting Scheme

2 Practical Insecurity

2.1 Packet level protocol issues

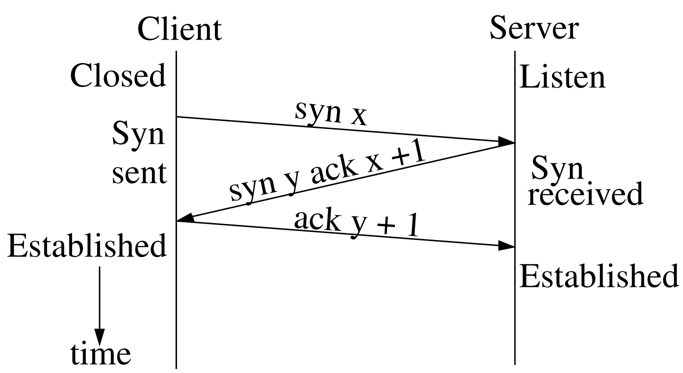

Figure 1: TCP Handshake

The TCP protocol is a connection-oriented, reliable protocol built on top of the IP protocol. Connections are created by the three-way handshake (illustrated above). A connection is requested in step 1, when host A sends to host B a SYN packet. Host B responds with a SYN/ACK packet, and then host A completes the handshake using an ACK packet. Layered on top of this exchange is the use of sequence numbers by both sides. Sequence numbers are used to describe the order of packets, so both sides can take steps to make sure that all the data is getting through in the right order.

There are a number of possible ways to exploit the TCP protocol, either to deny service from a legitimate user or to obtain unauthorized privileges.

[1]These notes contain only the topics discussed in lecture and are derived from Roger Dingledine's document in Handout 30 (http://web.mit.edu/6.857/www/handouts/H30-insecurity.pdf).

IP address spoofing

In general, an attacker can claim to come from an IP address different from his actual address by setting the packet source address to the IP address of a different (possibly non-existent) host.

1. Allows trust-based attacks (described later)
2. Hard to trace
3. Works particularly well when spoofing rhost

Note that if the attacker does spoof his IP address, he needs to either be good at guessing responses, or have some way of intercepting or overhearing the response packets.

Question: What is a rhost relationship?
Answer: Rhost is an IP-based trusted relationship, with no authentication of users required from trusted computers.

SYN flooding

A computer keeps track of pending connections in a buffer. This buffer generally remembers between 5 and 15 simultaneous connections. Connections are removed from the queue as they are processed, or after a timeout period (generally 75 seconds). If more SYN packets arrive while the buffer is full, these packets are dropped.

1. Little bandwidth needed by attacker
2. Use IP address spoofing from an unreachable host

An effective solution to this problem is to use "SYN cookies," which allow the machine to not have to remember an incoming connection until the handshake is completed. This is done by using a cryptographic hash (of the IP address, the ports involved, and a server secret) as the sequence number. If the ACK in the third phase of the handshake has the correct hash value, then the host can assume that the first two phases of the handshake were performed correctly.

Full connection flooding

Inetd is typically susceptible to a denial of service (DoS) attack involving many consecutive connects to the same service; this is sometimes called a "flooding" attack. When too many connections are made within a specified short period of time, inetd will terminate that service for a short period of time (typically five minutes).

There are attack programs which are capable of opening and maintaining hundreds or even thousands of connections at once. These programs will reliably terminate services that inetd controls, and for stand-alone daemons it can be used to overflow the target's process table or file descriptors. Indeed, it's particularly effective against sshd and sendmail, because they fail to exit cleanly when the descriptor table is full or nearly full.

Tying up the resources of the target machine is a very effective means of attack. Since it's working within the intended specifications of the protocol, there's nothing to "fix" – and it's difficult to defend against because most solutions tend to involve potentially reducing convenience or accessibility for legitimate users.

Apache was designed to be resistant to these kinds of attacks.

Sniping

Sniping is possible if an attacker can

- Read packets of a TCP connection and keep track of the sequence numbers
- Guess the sequence numbers

From there, it's a simple matter to send a RST packet with the correct sequence number (from the right IP address, of course). This will close down one side of the connection; when data is sent from the other side, a RST packet will be generated.

Hijacking

More complex than simply sniping the connection is hijacking the connection. By knowing the current sequence numbers, an attacker can snipe one end of the connection and take over talking to the other side as if nothing had changed.

In a more sophisticated attack, the cracker spoofs the hardware address of one side to convince the target that that side has changed hardware addresses. This causes the target machine to send its replies to this (new, possibly non-existent) address. This means that you don't have to snipe one of the connections, and you can conceivably reconnect the original client later by correcting the ARP tables and sending acknowledgments with updated sequence numbers.

Question: Why does TCP allow the hardware address of one party to change during a connection?
Answer: Apparently, just for this exploit (laugh). I have never seen a legitimate use of this feature during a session. In order to support any kind of hardware address change, it has to be able to be supported everywhere.

Question: It seems to me you can convince a gateway to send the data packets elsewhere, without ever having to go to either side?
Answer: Yes, this attack also works on a switch and router level. The attacker convinces the switch that the destination IP address lives elsewhere, and the connection has been hijacked.

A particular Telnet exploit listened for a connection on the local network, hijacked the connection to the server, presented the server packets containing instructions to rewrite the permissions file (.rhosts file), and then gave the connection back to the original client.

Question: If a connection is hijacked, would the sequence numbers match?
Answer: Typically no, the sequence numbers would not match. The server and client would disagree over the correct sequence numbers, spawning an ACK storm as both sides send ACK packets until the connection times out or closes.

Question: How do you prevent an ACK storm after attacking?
Answer: It is not easy. The only solution is to ARP spoof both sides of a connection and send an equal number of false packets to both sides.

Question: Doesn't a firewall stop ACK storm?
Answer: Yes, but the attack is likely to occur on a local level.

2.2 Lower-level Protocols

Packet Fragmenting

One of the options on a packet that can be set in the IP header is whether this packet is fragmented (often a router will need to fragment a packet to allow it through that section of the network, e.g. Ethernet to ATM).

The "offset" field in the header is used to put all the fragments of a datagram back together. One simple but effective attack that exploited this bug in a surprisingly large number of TCP implementations a few years ago was to provide faulty (for instance, negative) offset numbers in the packets. When reassembling the packet, the target machine would accidentally overwrite some other part of its kernel memory, causing arbitrary reboots or freezes. This problem is rooted in the implementations of TCP, which are designed to be as fast as possible.

One example of this attack occurred at MIT several years ago. At 3am, every Athena machine on campus suddenly crashed. Machines were rebooted, and the machines promptly crashed again. The problem was eventually traced to a fragmented packet attack being sent to all machines. These machines would automatically reassemble the packets (without any authentication), which would then overwrite random parts of the kernel.

This attack is typically used by script kiddies to crash remote computers.

Question: Is it possible to do more than just causing crashes, such as rewriting the kernel?
Answer: Yes, but it is very hard to do. The kernel looks different on different machines, and the memory space allocated for packet reassembly constantly changes, in a hard to predict manner.

Question: Do corporate firewalls protect against this kind of attack?
Answer: A few years ago, the answer was no. Now, maybe. Corporate firewalls can handle a large amount of traffic, but reassembling fragmented packets at the firewall level can be very computationally expensive. I suspect most corporate firewalls basically do not work against an intelligent adversary.

Question: How does this differ from buffer overrun attacks?
Answer: Buffer overruns are less hardware specific, and buffer overruns also tend to allow the attacker to overwrite the stack and execute code, which is almost impossible here.

ICMP exploits

The Internet Control Message Protocol (ICMP) is another protocol on top of IP (just like TCP is a protocol on top of IP). ICMP is used to communicate information about network topology changes, routing suggestions, etc. Common ICMP packet types are 'destination unreachable', 'source quench', 'redirect message', and 'routing advertisement'.

ICMP packets are notorious for their lack of authentication, allowing a wide variety of exploits and attacks. However, most routers these days talk to each other using the Border Gateway Protocol (BGP) or similar routing protocols, rather than using ICMP.

Echo port attacks exploit a relatively high-level protocol to deny service. The echo service (port 7) simply echoes whatever text is sent to it. If you convince two machines that they are both talking to each other on port 7 (through IP address spoofing), they will keep the conversation going until enough packets are dropped. More directly, if you send a packet to a target IP address spoofed from that same IP address, the machine may well spend all of its resources in a tight loop echoing to itself. This bug is fixed in most modern systems.

2.3 More Packet-level Issues

Probing

The TCP protocol can also be used to gain more information about a target machine. Following are a number of different types of probes

- Simple probing. By making a connection to a port on the target's machine, you can determine if this port is listening. On the other hand, the target will be able to easily identify (and log) the fact that you made that connection.
- A slightly more subtle approach is to send just the initial SYN packet of a TCP connection. If you receive a SYN/ACK, then chances are good that the target is listening on that port. On the other hand, if you receive a RST packet, the port is probably closed. By not responding to the SYN/ACK, you make it less likely that the target machine will log your probe.
- FIN scanning: send a FIN packet to the port in question. If the port is closed, generally the target will be polite enough to respond with a RST. If the port is open, however, generally the target will silently drop the FIN. This approach to probing is even less likely to be logged; however, note that if your FIN packet is dropped en route to the target (either because of firewall filtering or because of general network congestion), you will falsely conclude that the port is available.
- ICMP echo reply scanning: because many firewalls block incoming ICMP echo requests (pings), scanning machines behind a firewall with ping doesn't work very well. But since the firewall configuration generally wants users behind the firewall to be able to ping *out*, then ICMP echo replies (answers to pings) are not filtered. By sending echo replies to various hosts behind a firewall, it's possible to determine which of these hosts exist (are reachable) and which do not exist – for each host which doesn't exist, some router along the way is kind enough to send back an ICMP unreachable packet.

Logging

Of course, there are programs (known broadly as Intrusion Detection Systems) designed to listen for these sorts of probes (as well as other attacks) and notify the host's administrators. If the target machine is very lightly loaded, then it will be difficult to do probing without getting noticed. However, if the target has thousands of connections a minute (like `cnn.com`), the odds of a probe being noticed is low.

Other ways of avoiding logging by IDS systems are

1. Space out probes over time
2. Use multiple machines to probe
3. Hide your probe scans with duplicate scans from non-existent IP addresses
4. Use packet fragmentation on probe packets

Sniffing

The bodies of the first few packets in a telnet or ftp session can often yield a username/password pair, allowing for simpler methods of gaining unauthorized access to the target.

2.4 Up a layer of abstraction

Of course, the above are just some of the attacks you can make against the underlying TCP protocol. There are a number of other interesting ways besides sniffing and low-level probing to gain information about a computer system.

- Finger: you can gain a surprising amount of information from simply fingering a machine to see the current logins, and sometimes personal information about users.
- TCP fingerprinting: Different operating systems have different TCP responses, programs like queso (and now nmap) can identify the operating system of the remote machine based on its replies, narrowing down the types of attacks that are likely to succeed.
- Banners: you can learn a lot from a machine by simply connecting to its ftp port, telnet port, sendmail port, or httpd port, then collecting the information. The sendmail and telnet banners will often tell you the operating system flavor (including distribution) and architecture of the target.
- DNS Spoofing: name servers tend not to authenticate connections as strongly as they should (routers have this same problem – it stems from the fact that they need to support as wide a variety of systems as possible, and some of those systems don't have a good way of authenticating in the first place), which means that attacks are often possible that add or change a line in the DNS tables. Specifically, you might convince a local name server that `www.microsoft.com` points to a different place. At the least, you have just denied service from normal Microsoft

customers. A more subtle attack would be to mirror Microsoft's page onto your fake one, modifying the service packs or other patches to have bugs or backdoors.

2.5 Direct attacks

Rather than exploiting some protocol to gain information or deny a service, there are a number of exploits which attempt to actually obtain control of a machine.

- rlogin: by guessing the correct sequence number (in some TCP implementations this is relatively easy to do), an attacker can take advantage of trust relationships between computers. If one computer allows rlogin without a password from a certain domain, the attacker can spoof his IP address to be that of a machine without that domain; even if he can't read the response packets, he can login and blindly open up another hole in the system or perform other actions.

- cgi: early Apache web servers (and others) included cgi programs such as phf, handler, etc. These programs were intended to make certain operations and interactions with the server more convenient, but they also included a number of security holes that would allow an attacker to request a copy of the password file, or sometimes even execute arbitrary commands (under the permissions of the Web server, which were typically nobody, but sometimes root on the remote machine).

- Other remote services that are frequently exploited include sendmail (this program is enormous, and contains a very wide variety of vulnerabilities), mountd (a couple of remote buffer overflow bugs have been found recently in both of these), ftp (a remote buffer overflow bug was found recently that involved creating long path names in the incoming directory), imapd and popd, etc. Exploits on RPC services on Suns have been common in the past year. Buffer overflow bugs giving access to /shell can be used locally or remotely to great effect.

2.6 User-level Local Attacks

In most cases, it's easier to obtain access to a target machine at the user level of privilege than the root level. (Often, this can be done by sniffing passwords and then simply logging in.) Once in, however, the attacker has a number of avenues available to "break root" from a local shell. A few examples follow.

telnet ld_preload

In general, telnet reads an environment variable which can specify which dynamic libraries should be used on the remote side during the login process. In some older systems, these libraries are loaded by root before control is transferred to the user. If the attacker puts a modified libc library on the target machine (via anonymous ftp) and then logs in specifying he wants that library as his libc, he can force the machine to execute instructions as root.

Dictionary Attacks

If the local password file is readable, the attacker can simply grab a copy and start an offline dictionary attack on it, hoping that root's password is something easily guessed.

One solution to this vulnerability is to use "shadow" password files, where the actual process of comparing to see if a password is valid is done only by root, and therefore only root needs access even to the hashed version of the passwords.

Another way of gaining the password file is to cause the process doing the password comparison to segmentation fault, leaving a copy of the shadow password file in the core image.

Another solution is to use a different scheme for encrypting passwords. For instance, Red Hat 6.x offers the option of using MD5 to produce the password hash, rather than the traditional DES. While this is still vulnerable to a dictionary attack, the MD5 hash allows for passwords longer than 8 characters, and also is a slower hash, making dictionary attacks more computationally challenging.

X-Windows key sniffing

If a user is running X on the target machine and using an xterm, a local attacker can watch the events and record one side of an outgoing telnet or ssh session, capturing keystrokes directly and bypassing all encryption.

In addition to simply monitoring the X events, an attacker can also generate new X events and insert them into the stream. This means that the attacker can execute commands on xterms owned by other users, opening up several new vulnerabilities.

This is solved in part by using "X cookies," an authentication mechanism to make sure that only truly authenticated hosts can interact with the X events. The X authentication system is complex enough that holes show up every so often, though.

2.7 Root-level Local Attacks

Rootkits

Once an attacker has obtained root on a system, he can do pretty much anything he wants. However, many crackers don't have a good idea of how to hide their tracks once they're in, or how to open other security holes to make returning an easier process. Instead, they use packages called "rootkits," which the cracker can simply transfer to the compromised machine and install. A typical rootkit will install new binaries on the target machine which make it more difficult to detect unusual behavior, as well as change the timestamps of files which might indicate a break-in, or even remove relevant lines from the system logs. More advanced rootkits are available to do a wide variety of cover-ups. Most rootkits used by script kiddies are of poor quality.

Kernel Modules

Perhaps the most insidious attack against a modern Unix machine is to install a new kernel module once you've compromised root. Kernel modules live in kernel-space rather than normal user-space. They can intercept system calls and modify (or completely replace) their behavior. In this way, a cracker can truly take control of a machine and prevent him from being noticed.

2.8 Conclusion

- **Don't Do This!**: The FBI will come knocking
- Be Careful: use chains of multiple computers in different jurisdictions to avoid being tracked down (computers Iraq and Russia make great sources to launch attacks).
- This stuff is old: most attacks described here have been fixed
- Not just UNIX: Attacks on Windows work just as well, and many windows bugs have not been fixed.
- Physical access trumps: If an attacker can gain physical access to a target machine, he wins.

Question: Where can I learn more about this?
Answer: Web sites (`www.packetstorm.com`, `www.l0pht.com`, `www.defcon.org`), ask people in the know, and Google searches on items above.

3 Brazilian Voting System

Caltech MIT Voting Study

- Voter id cards
- Everyone is required to vote
- Simple electronic system
  - Voter types 5-digit numbers corresponding to candidate
  - Vote recorded to flash memory
  - Flash memory is collected and tallied
- Audit: Two machines store vote tally for redundancy
  - Complicated flash memory in voting machine
  - External, very simple tally machine stores votes as they occur.

Question: Can an authority correlate votes with voters?
Answer: No, both machines just keep running tallies and do not record times of votes or other details.