Hacking: The Basics

Zachary Wilson
April 4, 2001

Introduction

An intrusion can be defined as an attempt to break into or misuse a computer system. The word "misuse" is broad, and can reflect meaning to something as severe as stealing confidential data to something as minor such as misusing your email system for spam. Today, both the Internet and corporate intranets are simply crawling with people from all walks of life that are continuously trying to test the security of various systems and networks. Some of these people are seeking some sort of intellectual high, while others are fueled by more treacherous motives such as revenge or stealing for profit. In any event, no intrusion is innocent and no intrusion is benign. There is no silver bullet available out there that will totally secure our networks and systems. The only thing that we can do as IT professionals is make sure all of the doors are locked, the alarm is turned on, and educate ourselves on what to look for. The primary focus of this practical paper is to educate the less security conscious IT professionals and end-users on exactly who is out there and what they are doing to get in. By attempting to establish this baseline of security knowledge we essentially extend the arm of IT security to include the very users who today present the greatest danger: the uneducated user.

Attacker Profiles

There are two words to describe the attacker: **hacker** and **attacker**. A hacker is a generic term for a person who likes getting into things. The benign hacker is the person who likes to get into his/her own computer and understand how it works. The malicious hacker is the person who likes getting into other people's systems. The benign hackers wish that the media would stop bad-mouthing all hackers and use the term 'attacker' instead. Unfortunately, this is not likely to happen. In any event, the word used to denote anybody trying to get into your system in this paper is 'attacker'.

Attackers can be classified into two categories.

Outsiders

These are attackers from outside your network attempting to attack you external presence (deface web servers, forward spam through e-mail servers, etc.). They may also attempt to go around the firewall to attack machines on the internal network. Outside attackers may come from the **Internet**, **dial-up** lines, **physical break-ins**, or from **partner** (vendor, customer, reseller, etc.) network that is linked to your corporate network.

Insiders

These are attackers that have legitimate reasons to use/access your internal network. These include users who misuse privileges or who impersonate higher privileged users. A frequently quoted statistic cites that insiders commit 80% of security breaches.

Intrusion Techniques

These are the primary ways an attacker can get into a system:

Physical Intrusion - If an attacker has physical access to a machine (i.e. they can use the keyboard or take apart the system), they will be able to get in. Techniques range from special

privileges the console has, to the ability to physically take apart the system and remove the disk drive (and read/write it on another machine).

System Intrusion - This type of hacking assumes the attacker already has a low-privilege user account on the system. If the system doesn't have the latest security patches, there is a good chance the attacker will be able to use a known exploit in order to gain additional administrative privileges.

Remote Intrusion - This type of hacking involves an attacker who attempts to penetrate a system remotely across the network. The attacker begins with no special privileges. There are several forms of this type of hacking. Note that Network Intrusion Detection Systems are primarily concerned with Remote Intrusion.

Possible vulnerabilities and ways to exploit them.

Software bugs

Software always has bugs. System administrators and programmers can never track down and eliminate all possible software vulnerabilities. Attackers have to only find one hole to break in. Software bugs are often exploited in the server daemons, client applications, operating systems, and the network stack. Software bugs can be classified in the following manner:

Buffer overflows - Almost all the security holes you read about are due to this problem. A typical example is a programmer who sets aside 256 characters to hold a login username. However, if an attacker tries to enter in a false username longer than that you might have a problem. All the attacker has to do is send 300 characters, including code that will be executed by the server, and voila, game over. Hackers find these bugs in several ways. First, the source code for a lot of services is available on the net. Hackers routinely look through this code searching for programs that have buffer overflow problems. Secondly, hackers may look at the programs themselves to see if such a problem exists. Thirdly, hackers will examine every place the program has input and try to overflow it with random data. If the program crashes, there is a good chance that carefully constructed input will allow the attacker to gain access.

Unexpected combinations - Programs are usually constructed using many layers of code, including the underlying operating system as the bottom most layer. Attackers can often send input that is meaningless to one layer, but meaningful to another layer. The most common language for processing user input on the web is PERL. Programs written in PERL will usually send this input to other programs for further evaluation. A common hacking technique would be to enter something like "`| mail < /etc/passwd`". This gets executed because PERL asks the operating system to launch an additional program with that input. However, the operating system intercepts the pipe '|' character and launches the 'mail' program as well, which causes the password file to be emailed to the attacker.

Race conditions - Most systems today are "multitasking/multithreaded". This means that they can execute more than one program at a time. There is a danger if two programs need to access the same data at the same time. Imagine two programs, ABC and XYZ, each program attempts to modify the same file. In order to modify a file, each program must first read the file into memory, change the contents in memory, then copy the memory back out into the file. The race condition occurs when program ABC reads the file into memory and then makes the change. However, before ABC gets to write the file, program XYZ steps in and does the full read/modify/write on the file. Now program ABC writes its copy back out to the file. Since

program ABC started with a copy before XYZ made its changes, all of XYZ's changes will be lost. Since you need to get the sequence of events in just the right order, race conditions are very rare. Attackers usually attempt such actions thousands of times before they get it right, and gain access to the system.

Unexpected input - Most programs are written to handle valid input. Most programmers do not consider what happens when somebody enters input that doesn't match the specification.

System configuration bugs

Default configurations - Most systems are shipped to customers with default, easy-to-use configurations. Unfortunately, "easy-to-use" means "easy-to-break-in". Almost any UNIX or WinNT machine shipped to you can be hacked in easily.

Poor system administrator practices - A surprising number of machines are configured with an empty root/administrator password. This is because the administrator is too lazy to configure one right now and wants to get the machine up and running quickly with minimal fuss. Unfortunately, they never get around to fixing the password later, allowing attackers easy access. One of the first things an attacker will do on a network is to scan all machines for empty passwords.

Running unnecessary services - Virtually all programs can be configured to run in a non-secure mode. Sometimes administrators will inadvertently open a hole on a machine. Most administration guides will suggest that administrators turn off everything that doesn't absolutely positively need to run on a machine in order to avoid accidental holes. Note that security-auditing packages (such as Enterprise Security Manager from Symantec) can usually find these holes and notify the administrator.

Trust relationships - Attackers often "island hop" through the network exploiting trust relationships. A network of machines trusting each other is only as secure as its weakest link.

Password cracking

Easy-to-guess passwords - These are passwords where people use the names of themselves, their children, spouse/SO, pet, or car model as their password. Then there are the users who choose "password" or simply null passwords.

Dictionary attacks - With this attack, the attacker will use a program that will try every possible word in the dictionary. Dictionary attacks can be done either by repeatedly logging into systems, or by collecting encrypted passwords and attempting to find a match by similarly encrypting all the passwords in the dictionary. Attackers usually have a copy of the English dictionary as well as foreign language dictionaries for this purpose. They all use additional dictionary-like databases, such as names (see above) and lists of common passwords.

Brute force attacks - Similar to a Dictionary attack, an attacker may try all possible combinations of characters. A short 4-letter password consisting of lower-case letters can be cracked in just a few minutes. A long 7-character password consisting of upper and lower case, as well as numbers and punctuation can take months to crack assuming you can try a million combinations a second (in practice, a thousand combinations per second is more likely for a single machine).

Sniffing unsecured traffic

Shared medium - On traditional Ethernet, all you have to do is put a sniffer on the wire to see all the traffic on a segment. This is getting more difficult now that most corporations are transitioning to switched Ethernet.

Server sniffing - However, on switched networks, if you can install a sniffing program on a server (especially one acting as a router), you can probably use that information to break into client machines and trusted machines as well. For example, you might not know a user's password, but sniffing a Telnet session when they log in will give you that password.

Remote sniffing - A large number of boxes come with RMON enabled and public community strings. While the bandwidth is really low (you can't sniff all the traffic), it presents interesting possibilities.

Design flaws

Even if a software implementation is completely correct according to the design, there still may be bugs in the design itself that leads to intrusions.

TCP/IP protocol flaws - The TCP/IP protocol was designed before we had much experience with the wide-scale hacking we see today. As a result, there are a number of design flaws that lead to possible security problems. Some examples include smurf attacks, ICMP Unreachable disconnects, IP spoofing, and SYN floods. The biggest problem is that the IP protocol itself is very "trusting": hackers are free to forge and change IP data with impunity. IPsec (IP security) has been designed to overcome many of these flaws, but it is not yet widely used.

UNIX design flaws - There are number of inherent flaws in the UNIX operating system that frequently lead to intrusions. The chief problem is the access control system, where only 'root' is granted administrative rights.

Acquiring Passwords

Clear-text sniffing - A number of protocols (Telnet, FTP, HTTP Basic) use clear-text passwords, meaning that they are not encrypted as the go over the wire between the client and the server. An attacker with a protocol analyzer can watch the wire looking for such passwords. No further effort is needed; the attacker can start immediately using those passwords to log in.

Encrypted sniffing -Most protocols, however, use some sort of encryption on the passwords. In these cases, the attacker will need to carry out a Dictionary or Brute Force attack on the password in order to attempt decryption. Note that you still don't know about the attacker's presence, as he/she has been completely passive and has not transmitted anything on the wire. Password cracking does not require anything to be sent on the wire as attacker's own machine is being used to authenticate your password.

Replay attack - In some cases, attackers do not need to decrypt the password. They can use the encrypted form instead in order to login to systems. This usually requires reprogramming their client software in order to make use of the encrypted password.

Password file stealing - The entire user database is usually stored in a single file on the disk. In UNIX, this file is `/etc/passwd` (or some mirror of that file), and under WinNT, this is the SAM file. Either way, once an attacker gets hold of this file, he/she can run cracking programs in order to find some weak passwords within the file.

Observation - One of the traditional problems in password security is that passwords must be long and difficult to guess (in order to make Dictionary and Brute Force cracks unreasonably

difficult). However, such passwords are often difficult to remember, so users write them down somewhere. Attackers can often search a persons work site in order to find passwords written on little pieces of paper (usually under the keyboard). Attackers can also train themselves to watch typed in passwords behind a user's back.

Social Engineering – One successful and common technique is to simply call the helpdesk and say "Hi, this is Ron Smith the senior director for IT in San Jose. I have a presentation to give my boss, the CIO, and I can't log into server XYZ to get my notes. Would you please reset my password now? I have to be in this meeting in 2 minutes." Many unsuspecting operators would simply reset Ron's password in this situation. Most corporations have a policy where they tell users/operators/helpdesk to never give out or reset passwords, even to their own IT director, but this technique is still successful.

Typical intrusion scenarios

Footprinting - The attacker will find out as much as possible without actually giving themselves away. They will do this by finding public information or appearing as a normal user. In this stage, you really can't detect them. The attacker will do a 'whois' lookup to find as much information as possible about your network as registered along with your Domain Name (such as `foobar.com`. The attacker might walk through your DNS tables (using 'nslookup', 'dig', or other utilities to do domain transfers) to find the names of your machines. The attacker will browse other public information, such as your public web sites and anonymous FTP sites. The attacker might search news articles and press releases about your company.

Scanning - The attacker uses more invasive techniques to scan for information, but still doesn't do anything harmful. They might walk through all your web pages and look for CGI scripts (CGI scripts are often easily hacked). They might do a 'ping' sweep in order to see which machines are alive. They might do a UDP/TCP scan/strobe on target machines in order to see what services are available. They'll run utilities like 'rcpinfo', 'showmount', 'snmpwalk', etc. in order to see what's available. At this point, the attacker has done 'normal' activity on the network and has not done anything that can be classified as an intrusion. At this point, a NIDS will be able to tell you that "somebody is checking door handles", but nobody has actually tried to open a door yet.

Running exploits - The attacker crosses the line and starts exploiting possible holes in the target machines. The attacker may attempt to compromise a CGI script by sending shell commands in input fields. The attacker might attempt to exploit well-known buffer-overrun holes by sending large amounts of data. The attacker may start checking for login accounts with easily guessable (or empty) passwords. The attacker may go through several stages of exploits. For example, if the attacker was able to access a user account, they will now attempt further exploits in order to get root/admin access.

Establishing a foothold - At this stage, the attacker has successfully gained a foothold in your network by hacking into a machine. The attacker's main goal is to hide evidence of the attacks (doctoring the audit trail and log files) and make sure they can get back in again. They may install 'toolkits' that give them access, replace existing services with their own Trojan horses that have backdoor passwords, or create their own user accounts. System Integrity Verifiers (SIVs) can often detect an attacker at this point by noting the changed system files. The hacker will then use the system as a stepping-stone to other systems, since most networks have fewer defenses from inside attacks.

Playing for profit - The attacker takes advantage of their status to steal confidential data, misuse system resources (i.e. stage attacks at other sites from your site), or deface web pages.

Another scenario starts differently. Rather than attack a specific site, and attacker might simply scan random Internet addresses looking for a specific hole. For example, an attacker may attempt to scan the entire Internet for machines that have the SendMail DEBUG hole. They simply exploit such machines that they find. They don't target you directly, and they really won't even know who you are. (This is known as a 'birthday attack'; given a list of well-known security holes and a list of IP addresses, there is a good chance that there exists some machine somewhere that has one of those holes).

Common intrusion signatures

There are three types of attacks:

Reconnaissance - These include ping sweeps, DNS zone transfers, e-mail recons, TCP or UDP port scans, and possibly indexing of public web servers to find cgi holes.

Exploits - Attackers will take advantage of hidden features or bugs to gain access to the system.

Denial-of-service (DoS) attacks - Where the attacker attempts to crash a service (or the machine), overload network links, overloaded the CPU, or fill up the disk. The attacker is not trying to gain information, but to simply act as a vandal to prevent you from making use of your machine.

Common exploits

CGI scripts

CGI programs are notoriously insecure. Typical security holes include passing tainted input directly to the command shell via the use of shell metacharacters, using hidden variables specifying any filename on the system, and otherwise revealing more about the system than is good. The most well-known CGI bug is the 'phf' library shipped with NCSA httpd. The 'phf' library is supposed to allow server-parsed HTML, but can be exploited to give back any file. Other well-known CGI scripts that an attacker might attempt to exploit are: TextCounter, GuestBook, EWS, info2www, Count.cgi, handler, webdist.cgi, php.cgi, files.pl, nph-test-cgi, nph-publish, AnyForm, FormMail. If you see somebody trying to access one or all of these CGI scripts (and you don't use them), then it is clear indication of an intrusion attempt (assuming you don't have a version installed that you actually want to use).

Web server attacks

Beyond the execution of CGI programs, web servers have other possible holes. A large number of self-written web servers (include IIS 1.0 and NetWare 2.x) have hole whereby a file name can include a series of "../" in the path name to move elsewhere in the file system, getting any file. Another common bug is buffer overflow in the request field or in one of the other HTTP fields.

Web servers often have bugs related to their interaction with the underlying operating system. An old hole in Microsoft IIS have been dealing with the fact that files have two names, a long filename and a short 8.3 hashed equivalent that could sometimes be accessed bypassing permissions. NTFS (the new file system) has a feature called "alternate data streams" that is similar to the Macintosh data and resource forks. You could access the file through its stream name by appending "::$DATA" in order to see a script rather than run it.

Servers have long had problems with URLs. For example, the "death by a thousand slashes" problem in older Apache would cause huge CPU loads as it tried to process each directory in a thousand slash URL.

Web browser attacks

It seems that all of Microsoft's and Netscape's web browsers have security holes (though, of course, the latest ones never have any that we know about -- yet). This includes URL, HTTP, HTML, JavaScript, Frames, Java, and ActiveX attacks.

URL fields can cause a buffer overflow condition, either as it is parsed in the HTTP header, as it is displayed on the screen, or processed in some form (such as saved in the cache history). Also, an old bug with Internet Explorer allowed interaction with a bug whereby the browser would execute .LNK or .URL commands.

HTTP headers can be used to exploit bugs because some fields are passed to functions that expect only certain information.

HTML can be often exploited, such as the MIME-type overflow in Netscape Communicator's <EMBED> command.

JavaScript is a perennial favorite, and usually tries to exploit the "file upload" function by generating a filename and automatically hidden the "SUBMIT" button. There have been many variations of this bug fixed, then new ways found to circumvent the fixes.

Frames are often used as part of a JavaScript or Java hack (for example, hiding web-pages in 1px by 1px sized screens), but they present special problems. For example, a savy attacker can include a link to a trustworthy site that uses frames, then replace some of those frames with web pages from my own site, and they will appear to you to be part of that remote site.

Java has a robust security model, but that model has proven to have the occasional bug (though compared to everything else, it has proven to be one of the most secure elements of the whole system). Moreover, its robust security may be its undoing: Normal Java applets have no access to the local system, but sometimes they would be more useful if they did have local access. Thus, the implementation of "trust" models that can more easily be hacked.

ActiveX is even more dangerous than Java as it works purely from a trust model and runs native code. You can even inadvertently catch a virus that was accidentally imbedded in some vendor's code.

SMTP (SendMail) attacks

SendMail is an extremely complicated and widely used program, and as a consequence, has been the frequent source of security holes. In the old days (of the '88 Morris Worm), hackers would take advantage of a hole in the DEBUG command or the hidden WIZ feature to break into SMTP. These days, they often try buffer overruns. SMTP also can be exploited in reconnaissance attacks, such as using the VRFY command to find user names.

IMAP

Users retrieve e-mail from servers via the IMAP protocol (in contrast, SMTP transfers e-mail between servers). Hackers have found a number of bugs in several popular IMAP servers.

IP spoofing

There is a range of attacks that take advantage of the ability to forge (or 'spoof') your IP address. While a source address is sent along with every IP packet, it isn't actually used for routing. This means an attacker can pretend to be you when talking to a server. The attacker never sees the response packets (although your machine does, but throws them away because they don't match any requests you've sent). The attacker won't get data back this way, but can still send commands to the server pretending to be you.

IP spoofing is frequently used as part of other attacks:

SMURF

Where the source address of a broadcast ping is forged so that a huge number of machines respond back to victim indicated by the address, overloading it (or its link).

TCP sequence number prediction

In the startup of a TCP connection, you must choose a sequence number for your end, and the server must choose a sequence number for its end. Older TCP stacks choose predictable sequence numbers, allowing attackers to create TCP connections from a forged IP address (for which they will never see the response packets) that presumably will bypass security.

DNS poisoning through sequence prediction

DNS servers will "recursively" resolve DNS names. Thus, the DNS server that satisfies a client request will become itself a client to the next server in the recursive chain. The sequence numbers it uses are predictable. Thus, an attacker can send a request to the DNS server and a response to the server forged to be from the next server in the chain. It will then believe the forged response, and use that to satisfy other clients.

Common reconnaissance scans

Ping sweeps

This simple scan simply pings a range of IP addresses to find which machines are alive. Note that more sophisticated scanners will use other protocols (such as an SNMP sweep) to do the same thing.

TCP scans

Probes for open (listening) TCP ports looking for services the attacker can exploit. Scans can use normal TCP connections or stealth scans that use half-open connections (to prevent them from being logged) or FIN scans (never opens a port, but tests if someone's listening). Scans can be sequential, randomized, or configured lists of ports.

UDP scans

These scans are a little bit more difficult because UDP is a connectionless protocol. The technique is to send garbage UDP packets to the desired port. Most machines will respond with an ICMP "destination port unreachable" message, indicating that no service is listening at that port. However, many machines throttle ICMP messages, so you can't do this very fast.

OS identification

By sending illegal (or strange) ICMP or TCP packets, an attacker can identify the operating system. Standards usually state how machines should respond to legal packets, so machines tend to be uniform in their response to valid input. However, standards omit (usually intentionally) the response to invalid input. Thus, each operating system's unique responses to invalid inputs form a signature that attackers can use to figure out what the target machine is. This type of activity occurs at a low level (like stealth TCP scans) that systems do not log.

Account scans

Attempting to login to…..

- Accounts with no passwords
- Accounts with password same as username, or "password".
- Default accounts that were shipped with the product (a common problem on SGI, done to make setup easier)
- Accounts installed with software products (common on Microsoft as well as Unix, caused by products that run under their own special user account).
- Anonymous FTP problems (CWD ~root)
- Scan for rlogin/rsh/rexec ports, that may supported trusted logins.

Common DoS (Denial of Service) attacks

Ping-of-Death

Sends an invalid fragment, which starts before the end of packet, but extends past the end of the packet.

SYN Flood

Sends TCP SYN packet (which start connections) very fast, leaving the victim waiting to complete a huge number of connections, causing it to run out of resources and dropping legitimate connections. A new defense against this is the "SYN cookies". Each side of a connection has its own sequence-number. In response to a SYN, the attacked machine creates a special sequence number that is a "cookie" of the connection then forgets everything it knows about the connection. It can then recreate the forgotten information about the connection when the next packets come in from a legitimate connection.

Land/Latierra

Sends forged SYN packet with identical source/destination address/port so that system goes into infinite loop trying to complete the TCP connection.

WinNuke

Sends OOB/URG data on a TCP connection to port 139 (NetBIOS Session/SMB), which cause the Windows system to hang.

Conclusion

As I stated in my opening paragraph, no computer or computer network is completely secure. There are new vulnerabilities found or created everyday. The only way we as IT professionals can rest easy when we go home at night is to know that we are employing a minimal amount of security today and working towards more security tomorrow. However, all of the hard work and money spent on the best security tools available doesn't do us any good if users don't do minimal things like securing passwords and locking down workstations when they leave at night. Therefore, it is our responsibility as IT security professionals to educate these users to the best of our ability thus ensuring IT security is being employed even when nobody is watching. Hopefully, this paper will have the ability to make less educated users think about everything they do. And impress upon them consider security in all of their everyday practices.

Works consulted

Computer Incident Advisory Committee (CIAC) (1995). Advisory Notice F-08 Internet Spoofing and Hijacked Session Attacks.
[On-line], Available: http://ciac.llnl.gov/ciac/bulletins/f-08/shtml

Pethia, Richard. "Removing Roadblocks to Cyber Defense." 3/28/2000.
URL: http://www.cert.org/congressional_testimony/Pethia_testimony_Mar28-2000.html

CERT Incident Note 99-07. Distributed Denial of Service Tools. Nov 18, 1999.
URL: http://www.cert.org/incident_notes/IN-99-07.html

Csdweb@unb.ca "Passwords – Why yours is important."
http://www.unb.ca/csd/student/unix/passwords.html

Schneier, Bruce. "Security is not a product, it is a process". Crypto-Gram. 15 Dec 1999.
URL: http://www.counterpane.com/crypto-gram-9912.html

Vigilante. "Social Engineering." Internet Security.
URL: http://www.vigilante.com/inetsecurity/socialengineering.htm (12 February 2001).

Ryder, Josh. "Preventing Information Loss: Strengthening a Weak Link." Security Portel. 22 August 2000.
URL: http://www.securityportal.com/topnews/infoloss20000822.html (9 February 2001).