FAQ: Network Intrusion Detection Systems

Version 0.8.3, March 21, 2000

This FAQ answers simple questions related to detecting intruders who attack systems through the network, especially how such intrusions can be detected. **Questions? Feedback? Send mail to *nids-faq* @ *robertgraham.com***

0. Information about this FAQ

0.1 Copyright

0.6 Where to get it

My homepage: (slow link)
http://www.robertgraham.com/pubs/network-intrusion-detection.html (HTML)
http://www.robertgraham.com/pubs/network-intrusion-detection.txt (text)
TICM (fast link)http://www.ticm.com/kb/faq/
Shake Communications (Australia)http://www.shake.net/misc/network-intrusion-detection.htm
IT Sec (Germany)http://www.it-sec.de/mirrors/ids/network-intrusion-detection.html
Russian translation: http://www.citforum.ru/internet/securities/faq_ids.shtml
Japanese translation: http://www.sfc.keio.ac.jp/~keiji/ids/ids-faq-j.html

0.7 Thanks to

Thanks to the following people for helpful info and comments (note: to avoid automated spam address collection systems, I've munged their e-mail addresses in an obvious way).

Olaf Schreck <chakl at syscall de>
John Kozubik <john_kozubik at hotmail com> (see http://www.networkcommand.com/john/index.html for NT login-script tips).
Aaron Bawcom <abawcom at pacbell net>
Mike Kienenberger <mkienenb at arsc edu>
Keiji Takeda <keiji at sfc keio ac jp>
Scott Hamilton <sah at uow edu au>
Holger Heimann <hh at it-sec de>
Bennett Todd <bet at mordor dot net>

0.8 Version History

Version 0.7, October 9, 1999
Added info on limitations.
Version 0.6, July 17, 1999
Updated info from NAI and NFR straight from the vendors (hope I got it right). Added 8.7 and 8.8.
Version 0.5, May 19, 1999
Russian and Japanese translations available. Added some new IDS products.
Version 0.4, April 8, 1999
Section 8. Fixed TOC
Version 0.3, January 1, 1999
Minor updates
Changed format of hyper-links so I can create a text-only version of the FAQ.
Changed embedded e-mail address so that spam-trollers can't extract them.
Added TOC.
Version 0.2, November 1, 1998
Minor updates
Version 0.1, August 1, 1998
The first version.

1. Introduction

1.1 What is a "network intrusion detection system (NIDS)"?

An **intrusion** is somebody (A.K.A. "hacker" or "cracker") attempting to break into or misuse your system. The word "misuse" is broad, and can reflect something severe as stealing confidential data to something minor such as misusing your email system for spam (though for many of us, that is a major issue!).

An "Intrusion Detection System (IDS)" is a system for detecting such intrusions. For the purposes of this FAQ, IDS can be broken down into the following categories:

network intrusion detection systems (NIDS) monitors packets on the network wire and attempts to discover if a hacker/cracker is attempting to break into a system (or cause a denial of service attack). A typical example is a system that watches for large number of TCP connection requests (SYN) to many different ports on a target machine, thus discovering if someone is attempting a TCP port scan. A NIDS may run either on the target machine who watches its own traffic (usually integrated with the stack and services themselves), or on an independent machine promiscuously watching all network traffic (hub, router, probe). Note that a "network" IDS monitors many machines, whereas the others monitor only a single machine (the one they are installed on).

system integrity verifiers (SIV) monitors system files to find when a intruder changes them (thereby leaving behind a backdoor). The most famous of such systems is "Tripwire". A SIV may watch other components as well, such as the Windows registry and chron configuration, in order to find well known signatures. It may also detect when a normal user somehow acquires root/administrator level privleges. Many existing products in this area should be considered more "tools" than complete "systems": i.e. something like "Tripwire" detects changes in critical system components, but doesn't generate real-time alerts upon an intrusion.

log file monitors (LFM) monitor log files generated by network services. In a similar manner to NIDS, these systems look for patterns in the log files that suggest an intruder is attacking. A typical example would be a parser for HTTP server log files that looking for intruders who try well-known security holes, such as the "phf" attack. Example: swatch

deception systems (A.K.A. decoys, lures, fly-traps, honeypots) which contain pseudo-services whose goal is to emulate well-known holes in order to trap hackers. See *The Deception ToolKit* *http://www.all.net/dtk/* for an example. Also, simple tricks by renaming "administrator" account on NT, then setting up a dummy account with no rights by extensive auditing can be used. There is more on "deception" later in this document. Also see http://www.enteract.com/~lspitz/honeypot.html

other

For more info, see http://www.icsa.net/idswhite/.

1.2 Who is misusing the system?

There are two words to describe the intruder: **hacker** and **cracker**. 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 'cracker' instead. Unfortunately, this is not likely to happen. In any event, the word used in this FAQ is 'intruder', to generically denote anybody trying to get into your systems.

Intruders can be classified into two categories.

Outsiders

Intruders from outside your network, and who may 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 intruders 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

Intruders that legitimately use your internal network. These include users who **misuse priviledges** (such as the Social Security employee who marked someone as being dead because they didn't like

that person) or who **impersonate** higher privileged users (such as using someone else's terminal). A frequently quoted statistic is that 80% of security breaches are committed by insiders.

There are several types of intruders **Joy riders** hack because they can. **Vandals** are intent on causing destruction or marking up your web-pages. **Profiteers** are intent on profiting from their enterprise, such as rigging the system to give them money or by stealing corporate data and selling it.

1.3 How do intruders get into systems?

The primary ways a intruder can get into a system:

Physical Intrusion If a intruders have 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). Even BIOS protection is easy to bypass: virtually all BIOSes have backdoor passwords.

System Intrusion This type of hacking assumes the intruder 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 intruder will be able to use a known exploit in order to gain additional administrative privileges.

Remote Intrusion This type of hacking involves a intruder who attempts to penetrate a system remotely across the network. The intruder begins with no special privileges. There are several forms of this hacking. For example, a intruder has a much more difficult time if there exists a firewall on between him/her and the victim machine.

Note that Network Intrusion Detection Systems are primarily concerned with Remote Intrusion.

1.4 Why can intruders get into systems?

Software always has bugs. System Administrators and Programmers can never track down and eliminate all possible holes. Intruders have only to find one hole to break in.

1.4.1 Software bugs

Software bugs are exploited in the server daemons, the client applications, the operating system, and the network stack. Software bugs can be classified in the following manner:

Buffer overflows: Almost all the security holes you read about in the press are due to this problem. A typical example is a programmer who sets aside 256 characters to hold a login username. Surely, the programmer thinks, nobody will ever have a name longer than that. But a hacker thinks, what happens if I enter in a false username longer than that? Where do the additional characters go? If they hackers do the job just right, they can send 300 characters, including code that will be executed by the server, and voila, they've broken in. Hackers find these bugs in several ways. First of all, 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, though reading assembly output is really difficult. 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 hacker to break in. Note that this problem is common in programs written in C/C++, but rare in programs written in Java.

Unexpected combinations: Programs are usually constructed using many layers of code, including the underlying operating system as the bottom most layer. Intruders 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 intruder.

Unhandled 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.

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, A and B, who need 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 A reads the file into memory, then makes the change. However, before A gets to write the file, program B steps in and does the full read/modify/write on the file. Now program A writes its copy back out to the file. Since program A started with a copy before B made its changes, all of B's changes will be lost. Since you need to get the sequence of events in just the right order, race conditions are very rare. Intruders usually have to tries thousands of time before they get it right, and hack into the system.

1.4.2 System configuration

System configuration bugs can be classified in the following manner:

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.

Lazy administrators: 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 intruders easy access. One of the first things a intruder will do on a network is to scan all machines for empty passwords.

Hole creation: 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 can usually find these holes and notify the administrator.

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

1.4.3 Password cracking

This is a special category all to itself.

Really weak passwords: Most 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 nothing. This gives a list of less than 30 possibilities that a intruder can type in for themselves.

Dictionary attacks: Failing the above attack, the intruder can next try a "dictionary attack". In this attack, the intruder 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. Intruders 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, a intruder 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 (roughly, half a million possible combinations). A long 7-character password consisting of upper and lower case, as well as numbers and punctuation (10 trillion combinations) 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).

1.4.4 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.

1.4.5 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 protocool 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. As a result,

1.5 How do intruders get passwords?

Intruders get passwords in the following ways:

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. A intruder with a protocol analyzer can watch the wire looking for such passwords. No further effort is needed; the intruder 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 intruder 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 intruder'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 intruder's own machine is being used to authenticate your password.

Replay attack: In some cases, intruders 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 a intruder gets hold of this file, he/she can run cracking programs (described above) 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. Intruders can often search a persons work site in order to find passwords written on little pieces of paper (usually under the keyboard). Intruders can also train themselves to watch typed in passwords behind a user's back.

Social Engineering: A common (successful) technique is to simply call the user and say "Hi, this is Bob from MIS. We're trying to track down some problems on the network and they appear to be coming from your machine. What password are you using?" Many users will give up their password in this situation. (Most corporations have a policy where they tell users to never give out their password, even to their own MIS departments, but this technique is still successful. One easy way around this is for MIS to call the new employee 6-months have being hired and ask for their password, then criticize them for giving it to them in a manner they will not forget :-)

1.6 What is a typical intrusion scenario?

A typical scenario might be:

Step 1: **outside reconnaissance** The intruder 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 intruder 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 intruder might walk through your DNS tables (using 'nslookup', 'dig', or other utilities to do domain transfers) to find the names of your machines. The intruder will browse other public information, such as your public web sites and anonymous FTP sites. The intruder might search news articles and press releases about your company.

Step 2: **inside reconnaisance** The intruder 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 intruder 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.

Step 3: **exploit** The intruder crosses the line and starts exploiting possible holes in the target machines. The intruder may attempt to compromise a CGI script by sending shell commands in input fields. The intruder might attempt to exploit well-known buffer-overrun holes by sending large amounts of data. The intruder may start checking for login accounts with easily guessable (or empty) passwords. The hacker may go through several stages of exploits. For example, if the hacker was able to access a user account, they will now attempt further exploits in order to get root/admin access.

Step 4: **foot hold** At this stage, the hacker has successfully gained a foot hold in your network by hacking into a machine. The intruder'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 intruder 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.

Step 5: **profit** The intruder 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 intruder might simply scan random internet addresses looking for a specific hole. For example, an intruder 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).

1.7 What are some common "intrusion signatures"?

There are three types of attacks:

reconnaisance 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 Intruders will take advantage of hidden features or bugs to gain access to the system.

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

1.8 What are some common exploits?

1.8.1 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 intruder 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).

1.8.2 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 server 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.

1.8.3 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 both 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, I 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.

1.8.4 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.

1.8.5 Access

Failed login attempts, failed file access attempts, password cracking, administrative powers abuse

1.8.6 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.

1.8.7 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 intruder can pretend to be you when talking to a server. The intruder never sees the response packets (although your machine does, but throws them away because they don't match any requests you've sent). The intruder 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 intruders 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 intruder 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.

1.8.8 Buffer Overflows

Some other buffer overflow attacks are:

DNS overflow
: Where an overly long DNS name is sent to a server. DNS names are limited to 64-bytes per subcomponent and 256-bytes overall.

statd overflow
: where an overly long filename is provided

1.8.9 DNS attacks

DNS is a prime target because if you can corrupt the DNS server, you can take advantage of trust relationships.

DNS cache poisoning
: Every DNS packet contains a "Question" section and "Answer" section. Vulnerable servers will believe (and cache) Answers that you send along with Questions. Most, but not all, DNS servers have been patched as of November, 1998.

DNS poisoning through sequence prediction
: See above

DNS overflow
: See above

1.9 What are some common reconnaisance scans?

1.9.1 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.

1.9.2 TCP scans

Probes for open (listening) TCP ports looking for services the intruder 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 either sequential, randomized, or configured lists of ports.

1.9.3 UDP scans

These scans are a little bit more difficult because UDP is a connectionless protocol. The technique is to send a garbage UDP packet 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.

1.9.4 OS identification

By sending illegal (or strange) ICMP or TCP packets, an intruder 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 forms a signature that hackers 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.

1.9.5 Account scans

Tries to log on with accounts

- 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.

1.10 What are some common DoS (Denial of Service) attacks?

1.10.1 Ping-of-Death

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

1.10.2 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 are "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.

1.10.3 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.

1.10.4 WinNuke

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

1.11 How much danger from intrusions is there?

I frequently hear from people the statement "There's nothing on the system that anybody would want anyway". I walk them through various scenarios, such as simple ones if they've ever paid for anything on-line with a credit card or if they have any financial records or social security number on their personal machine.

More importantly, there is the issue of legal liability. You are potentially liable for damages caused by a hacker using your machine. You must be able to prove to a court that you took "reasonable" measures to defend yourself from hackers. For example, consider if you put a machine on a fast link (cable modem or DSL) and left administrator/root accounts open with no password. Then if a hacker breaks into that machine, then uses that machine to break into a bank, you may be held liable because you did not take the most obvious measures in securing the machine.

There is a good paper *http://www.cert.org/research/JHThesis/Start.html* by John D. Howard that discusses how much hacking goes on over the Internet, and how much danger you are in.

1.12 Where can I find current statistics about intrusions?

CyberNotes by NIPC (http://www.fbi.gov/nipc/welcome.htm)

CyberNotes is published every two weeks by the National Infrastructure Protection Center (NIPC). Its mission is to support security and information system professionals with timely information on cyber vulnerabilities, hacker exploit scripts, hacker trends, virus information, and other critical infrastructure-related best practices.

The NIPC was set up by the FBI in mid 1998, and its first major activity was to help track down the source of the Melissa virus (W97M.Melissa). The CyberNotes archive goes back to January 1999.

AusCERT Consolidated Statistics Project (http://www.auscert.org.au/Information/acsp/index.html)

A project to collect intrusion statistics from around the web and consolidate them. They want people to join and send them info.

An Analysis Of Security Incidents On The Internet 1989 - 1995 (http://www.cert.org/research/JHThesis/Start.html)

A dissertation by John D. Howard, Carnegie Mellon University

CERT Reports, Articles, and Presentations (http://www.cert.org/nav/reports.html)

CERT has a number of historical statistics on intrusions, but they aren't nearly as up-to-date as the NIPC.

1999 CSI-DBI Survey (http://www.gocsi.com/summary.htm) or (http://www.gocsi.com/prelea990301.htm

CSI (Computer Security Institute) does a number of surveys about intrusions and security

2. Architecture

2.1 How are intrusions detected?

2.1.1 Anomaly detection

The most common way people approach network intrusion detection is to detect statistical anomalies. The idea behind this approach is to measure a "baseline" of such stats as CPU utilization, disk activity, user logins, file activity, and so forth. Then, the system can trigger when there is a deviation from this baseline.

The benefit of this approach is that it can detect the anomalies without having to understand the underlying cause behind the anomalies.

For example, let's say that you monitor the traffic from individual workstations. Then, the system notes that at 2am, a lot of these workstations start logging into the servers and carrying out tasks. This is something interesting to note and possibly take action on.

2.1.2 Signature recognition

The majority of commercial products are based upon examining the traffic looking for well-known patterns of attack. This means that for every hacker technique, the engineers code something into the system for that technique.

This can be as simple as a pattern match. The classic example is to example every packet on the wire for the pattern "/cgi-bin/phf?", which might indicate somebody attempting to access this vulnerable CGI script on a web-server. Some IDS systems are built from large databases that contain hundreds (or thousands) of such strings. They just plug into the wire and trigger on every packet they see that contains one of these strings.

2.2 How does a NIDS match signatures with incoming traffic?

Traffic consists of IP datagrams flowing across a network. A NIDS is able to capture those packets as they flow by on the wire. A NIDS consists of a special TCP/IP stack that reassembles IP datagrams and TCP streams. It then applies some of the following techniques:

Protocol stack verification A number of intrusions, such as "Ping-O-Death" and "TCP Stealth Scanning" use violations of the underlying IP, TCP, UDP, and ICMP protocols in order to attack the machine. A simple verification system can flag invalid packets. This can include valid, by suspicious, behavior such as severally fragmented IP packets.

Application protocol verification A number of intrusions use invalid protocol behavior, such as "WinNuke", which uses invalid NetBIOS protocol (adding OOB data) or DNS cache poisoning, which has a valid, but unusually signature. In order to effectively detect these intrusions, a NIDS must re-implement a wide variety of application-layer protocols in order to detect suspicious or invalid behavior.

Creating new loggable events A NIDS can be used to extend the auditing capabilities of your network management software. For example, a NIDS can simply log all the application layer protocols used on a machine. Downstream event log systems (WinNT Event, UNIX syslog, SNMP TRAPS, etc.) can then correlate these extended events with other events on the network.

2.4 What happens after a NIDS detects an attack?

Reconfigure firewall
: Configure the firewall to filter out the IP address of the intruder. However, this still allows the intruder to attack from other addresses. Checkpoint firewall's support a "Suspicious Activity Monitoring Protocol (SAMP)" for configuring firewalls. Checkpoint has their "OPSEC" standard for re-configuring firewalls to block the offending IP address.

chime
: Beep or play a .WAV file. For example, you might hear a recording "You are under attack".

SNMP Trap
: Send an SNMP Trap datagram to a management console like HP OpenView, Tivoli, Cabletron Spectrum, etc.

NT Event
: Send an event to the WinNT event log.

syslog
: Send an event to the UNIX syslog event system.

send e-mail
: Send e-mail to an administrator to notify of the attack.

page
: Page (using normal pagers) the system administrator.

Log the attack
: Save the attack information (timestamp, intruder IP address, victim IP address/port, protocol information).

Save evidence
: Save a tracefile of the raw packets for later analysis.

Launch program
: Launch a separate program to handle the event.

Terminate the TCP session
: Forge a TCP FIN packet to force a connection to terminate.

2.5 What other countermeasures besides IDS are there?

Firewalls
: Most people think of the firewall as their first line of defense. This means if intruders figure out how to bypass it (easy, especially since most intrusions are committed by employees inside the firewall), they will have free run of the network. A better approach is to think of it as the *last* line of defense: you should be pretty sure machines are configured right and intrusion detection is operating, and then place the firewall up just to avoid the wannabe script-kiddies. Note that almost any router these days can be configured with some firewall filtering. While firewalls protect external access, they

leave the network unprotected from internal intrusions. It has been estimated that 80% of losses due to "hackers" have been internal attacks.

authentication

You should run scanners that automated the finding of open accounts. You should enforce automatically strict policies for passwords (7 character minimum, including numbers, dual-case, and punctuation) using crack or built in policy checkers (WinNT native, add-on for UNIX). You can also consider single-sign on products and integrating as many password systems as you can, such as RADIUS/TACACS integration with UNIX or NT (for dial-up style login), integrating UNIX *and* WinNT authentication (with existing tools are the new Kerberos in Windows 2000). These authentication systems will help you also remove "clear-text" passwords from protocols such as Telnet, FTP, IMAP, POP, etc.

VPNs (Virtual Private Networks)

VPNs create a secure connection over the Internet for remote access (e.g. for telecomuters). Example #1: Microsoft includes a a technology called PPTP (PPP over TCP) built into Windows. This gives a machine two IP addresses, one on the Internet, and a virtual one on the corporate network. Example #2: IPsec enhances the traditional IP protocol with security. While VPN vendors claim their product "enhance security", the reality is that they decrease corporate security. While the pipe itself is secure (authenticated, encrypted), either ends of the pipe are wide open. A home machine compromised with a backdoor rootkit allows a hacker to subvert the VPN connection, allow full, undetectable access to the other side of the firewall.

encryption

Encryption is becoming increasingly popular. You have your choice of e-mail encryption (PGP, SMIME), file encryption (PGP again), or file system encryption (BestCrypt, PGP again).

lures/honeypots

Programs that pretend to be a service, but which do not advertise themselves. It can be something as simple as one of the many BackOrifice emulators (such as NFR's Back Officer Friendly), or as complex as an entire subnet of bogus systems installed for that purpose.

2.6 Where do I put IDS systems on my network?

network hosts

Even though network intrusion detection systems have traditionally been used as probes, they can also be placed on hosts (in non-promiscuous mode). Take for example a switched network where an employee is on the same switch as the CEO, who runs Win98. The windows machine is completely defenseless, and has no logging capabilities that could be fed to a traditional host-based intrusion detection system. The employee could run a network-based password cracker for months without fear of being caught. A NIDS installed like virus scanning software is the most effective way to detect such intrusions.

network perimeter

IDS is most effective on the network perimeter, such as on both sides of the **firewall**, near the **dial-up** server, and on links to **partner** networks. These links tend to be low-bandwidth (T1 speeds) such that an IDS can keep up with the traffic.

WAN backbone