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
: Another high-value point is the corporate WAN backbone. A frequent problem is hacking from "outlying" areas to the main corporate network. Since WAN links tend to be low bandwidth, IDS systems can keep up.

server farms
: Serves are often placed on their own network, connected to switches. The problem these servers have, though, is that IDS systems cannot keep up with high-volume traffic. For extremely important servers, you may be able to install dedicate IDS systems that monitor just the individual server's link. Also, application servers tend to have lower traffic than file servers, so they are better targets for IDS systems.

LAN backbones
: IDS systems are impractical for LAN backbones, because of their high traffic requirements. Some vendors are incorporating IDS detection into switches. A full IDS system that must reassemble packets is unlikely to keep up. A scaled-down system that detects simpler attacks but can keep up is likely to be a better choice.

2.7 How does IDS fit with the rest of my security framework?

1. Put firewalls between areas of the network with different security requirements (i.e. between internet-localnet, between users-servers, between company-parterns, etc).
2. Use network vulnerability scanners to double check firewalls and to find holes that intruders can exploit.
3. Use host policy scanners to make sure they conform to accepted practices (i.e. latest patches).
4. Use **Network intrusion detection systems** and other packet sniffing utilities to see what is actually going on.
5. Use **host-based intrusion detection systems** and virus scanners to flag successful intrusions.
6. Create an easy to follow policy that clearly states the response to intrusions.

2.8 How can I detect if someone is running a NIDS?

A NIDS is essentially a sniffer, so therefore standard sniffer detection techniques can be used. Such techniques are explained in http://www.robertgraham.com/pubs/sniffing-faq.html#detect.

An example would be to do a traceroute against the victim. This will often generate a low-level event in the IDS. Traceroutes are harmless and frequent on the net, so they don't indicate an attack. However, since many attacks are preceded by traceroutes, IDSs will log them anyway. As part of the logging system, it will usually do a reverse-DNS lookup. Therefore, if you run your own DNS server, then you can detect when somebody is doing a reverse-DNS lookup on your IP address in response to your traceroute.

3. Policy

3.1 How do I increase intrusion detection/prevention under WinNT?

The following lists items that make WinNT more secure, including detection as well as prevention. These are roughly listed in order of importance.

1. Install the latest service packs and "hot fixes". These are listed at http://www.microsoft.com/security/. If you are using WinNT 4.0 and you don't have Service Pack #3 (SP3) installed, an intruder can break into your system.
2. INSTALLATION: Use NTFS instead of FAT. NTFS allows permissions to be set on a per-file/per-directory basis. NTFS also allows auditing on a per-file/per-directory basis. Note that many people recommend using FAT as the boot drive and NTFS for all other drives (due to the ease-of-use in using DOS to fix things on a FAT drive). However, using NTFS for all drives is definitely more secure.
3. USRMGR: Rename the "administrator" account. A common attack is to use a Dictionary or brute force attack on the "administrator" account. Normal accounts can be configured to automatically (and temporarily) "lock out" after a few failed password attempts. However, this feature isn't possible for the administrator account because this allows a denial of service attack (i.e. prevent administration of the machine by locking out the administrator account).
4. USRMGR: Create a new account named "administrator" for detecting intrusion attempts.
5. USRMGR: Disable the "guest" account. You may also want to rename this account as (much like "administrator"). Once you've renamed the "guest" account, you may want to create a new account named "guest" for detecting hacking attempts.
6. NTFS: Disable "write" access for "Everyone" on the `%systemroot%/system32` directory.
7. REGEDT32: Turn on auditing for "HKEY_LOCAL_MACHINE\Security" in order to detect remote registry browsing.
8. INSTALLATION: Do not install in "C:\WINNT" directory. Sometimes intruders will be able to access files if they know the filename; installing in some other directory prevents a priori knowledge. Better yet, install in C:\WINNT, then reinstall in some other directory, then turn auditing on within that directory to alert you to people accessing those older files.
9. INSTALLATION: Use the boot partition only for booting and for system files. Put data and applications on a separate partition. It is also a good idea to separate applications from data.
10. CONTROLPANEL: Enable "Password Protected" on the screensaver. The best screensaver is "Blank Screen". You would think that screensavers run at idle priority, but this isn't always the case, so you can increase the performance of your server by using "Blank Screen". Also, this

will reduce power consumption in monitors, especially those that can detect a blank screen and turn themselves off. Finally, some screensavers (i.e. PointCast) are probably hackable.

11. REGEDT32: Turn off automatic sharing of ADMIN$, C$, D$, etc. via the "AutoShare" parameter in the registry. This parameter is under "HKEY_LOCAL_MACHINE\System\CurrentControlSet \Services\LanmanServer\Parameters", and is "AutoShareServer" for WinNT Server or "AutoShareWks" for WinNT Workstation. This is a DWORD, with a value of '1' for enabled (default), or a value of '0' for disabled. You will have to add the value yourself because it doesn't already exist in the registry.
12. REGEDT32: Turn of account/share information via anonymous access. Add "RestrictAnonymous" DWORD with a value of "1" to the registry key "HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet\Control\LSA" Note that if you see an error "Could not find domain controller for this domain." while setting domain trust relationships, you may have to change it back.
13. USRMGR: If you are using Domains (rather than Workgroups), change the user right "Access this computer from the network" to "Authenticated Users" rather than "Everyone". This disables remote access via local accounts on your machine, and allows only access through domain accounts.
14. PASSPROP: Enable lockout of the "administrator" account for remote access. This enables the situation where the remote intruder fails to guess the correct password after three tries. After lock-out, the administrator can only log in locally at the system console. You can also disable remote administrator access completely in USRMGR by removing the right "Access this computer from the network" from "Administrators", but this disables all remote administration, which make administration too difficult in a large WinNT environment.

Also consider physical intrusion prevention network wide. John Kozubik suggests using login scripts to force the built-in password protected screen-saver. In the login script, include the line like:

```
regedit /s \\MY_PDC\netlogon\scrn.reg
```

And in the file "scrn.reg", put the text:

```
REGEDIT4
[HKEY_CURRENT_USER\Control Panel\Desktop]
"ScreenSaveTimeOut"="1800"
"ScreenSaveActive"="1"
"SCRNSAVE.EXE"="c:\winnt\system32\logon.scr"
"ScreenSaverIsSecure"="1"
```

This will trigger the password prompt to appear 30-minutes after a user is away from the desktop (it doesn't log them out; just forces them to re-enter the password before they have access again).

3.2 How do I increase intrusion detection/prevention under Win95/Win98?

This section assumes you are a home user using Win95/Win98 to access the Internet. Win95/Win98 has no auditing or logging capabilities; you really should upgrade to WinNT if you are using the system for any serious purpose.

The following are techniques for the typical user:

1. Install the latest patches (of course).
2. Turn off print sharing. When print sharing is turned on, the system creates a PRINTER$ share

that allows remote systems to access printer drivers from the local system32 directory. Unfortunately, this allows remote systems to access non-driver files, such as the Win95 password file (combined with other Win95 bugs).
3. Turn off file sharing. As a home user, you probably don't need it. If you must share files, make sure that you choose a strong password, and only turn it on for brief moments while you need to share the files, then turn it off again.
4. (more forthcoming)

John Kozubik suggests the following techniques for corporate users (who presumably run login scripts from the servers). Since Win95/Win98 is so vulnerable, they provide easy penetration to the rest of the corporate environment. Win95 caches passwords in easy-to-read formats, so you want to remove them.

`del c:\windows\*.pwl`

The password cache file will be the first one intruders look for. It has the same name as the user name, and poorly encrypts the cached passwords. Beware that this deletes dial-up passwords as well, so users that bring their notebooks into work and connect to the network will find their home dial-up passwords deleted.

Disable internal caching of passwords

Run:

```
REGEDIT /s \\MY_PDC\netlogon\nocache.reg
```

where "nocache.reg" consists of:

```
REGEDIT4
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Network]
"DisablePwdCaching"=dword:00000001
```

3.3 How do I increase intrusion detection/prevention under UNIX?

1. Do not install more services than you need. I installed everything on my RedHat Linux distribution and the machine lights up like a Xmas tree when port scanned. I already know of a few holes on that (test) machine that I can use to break in.
2. Use 'netstat' or a TCP/UDP scanner and '`rpcinfo`' to list all services on your machine. Again, make sure that everything you don't explicitly understand is turned off.
3. (more forthcoming; frankly, I've been more of an WinNT admin lately so my skills are getting rusty)
4. Read ftp://ftp.auscert.org.au/pub/auscert/papers/unix_security_checklist.

Of course, you might want to consider upgrading the system. There are a large number of SunOS 4.x systems out there, for example, even though Sun stopped "officially" supporting it many years ago.

3.4 How do I increase intrusion detection/prevention under Macintosh?

Macintoshes are 'end-user' systems, and support few services that can be hacked. In comparison, Windows machines are more numerous, and UNIX machines have a lot more interesting (hackable) services running on them. Thus, Macintoshes are frequently not the target of intruders.

Beyond that, I know of nothing in particular.

3.5 How do I increase intrusion detection/prevention for the enterprise?

First and foremost, create a security policy. Let's say that you are watching the network late in the evening and you see an intrusion in-progress. What do you do? Do you let the intrusion progress and collect evidence? Do you pull the plug? If so, do you pull the plug on the firewall between the intra- and extra- net? Or do you take down the entire Internet connection (preventing users from getting to you web site)? Who has the authority to pull the plug?

The priorities need to be set in place by the CEO of the corporation. Let's consider the scenario where you think you are being attacked, so you pull the plug. The users get up in arms, and complain. And, as it turns out, you were wrong, so your but gets fried. Even when blatant attacks are going on, few people pull the plug for fear of just such repercussions. Data theft is theoretical; ticked-off users are very real. Therefore, you need a policy from the very top that clearly states the importance of things and clearly lays out a procedure for what happens when an intrusion is suspected. [Author: does anybody have sample policies they can send me?]

Once you have the priorities straight, you need to figure out the technology. That's described in the next section.

3.6 How should I implement intrusion detection my enterprise?

Think about how you can configure the following systems in order to detect intruders:

1. **Operating Systems** such as WinNT and UNIX come with integrated logging/auditing features that can be used to monitor security critical resources. A section below discusses how to configure Windows and UNIX in order to enable intrusion detection.
2. **Services**, such as web servers, e-mail servers, and databases, include logging/auditing features as well. In addition, there are many tools that can be used to parse these files in order to discover intrusion signatures.
3. **Network Intrusion Detection Systems** that watch network traffic in an attempt to discover intrusion attempts. A section below lists a number of these products.
4. **Firewalls** usually have some network intrusion detection capabilities. After all, blocking intrusions is their primary purpose; it would be foolish not to detect intrusions as well.
5. **Network management platforms** (such as OpenView) have tools to help network managers set alerts on suspicious activity. At minimum, all SNMP devices should send "Authentication Failure" traps and management consoles should alert administrators when these go off.

3.7 What should I do when I've been hacked?

Read CERT's intruder detection checklist at ftp://ftp.cert.org/pub/tech_tips/intruder_detection_checklist.

For the most part, a good response requires that you've set up good defensive measures in the first place. These include:

incident response team

Set up an "incident response team". Identify those people who should be called whenever people suspect an intrusion in progress. The response team needs to be "inter-departmental", and include such people as:

upper management

Need to identify somebody with the authority to handle escalated issues. For example, if the company has an online trading service, you need to identify somebody with enough power to "pull the plug". Going off-line on such a service will have a major impact -- but would still be better than hackers trading away people's stocks.

HR (Human Resources)

Many attacks come from internal employees. This consists of both serious attacks (cracking into machines) as well as nuisance attacks, such as browsing inappropriate servers looking for files like customer lists that might be left open.

technical staff

Security is often separate from normal MIS activity. If security personel detects a compromised system, they need to know who in MIS they need to call.

outside members

Identify people outside the company that may be contacted. This might be a local ISP person (for example, helping against smurf attacks), the local police, or the FBI. These aren't necessarily "formal" team members. They might not know anything about this, or they might simply be a "role" (like support@localisp.net). But put their names on the list so that everyone knows who to call.

security team

Of course, the most important team members will be the security people themselves.

Note that not all "team members" need to be involved with every incident. For example, you only need to ping upper management on serious attacks. They may never be called upon, but they do need to be identified, and they do need to be prepared as to the types of decisions they will have to make.

response procedure

Figure out guidelines now for the response action. For example, you need to decide now what your priorities are between network uptime and intrusion: can you pull the network plug whenever you strongly suspect intrusion? Do you want to allow continued intrusion in order to gather evidence against the intruder? Decide now, and get the CEO's approval now, because you won't have time during the attack.

lines of communication

Figure out guidelines for communication. Do you propagate the information up the corporate food chain from your boss up to the CEO, or horizontally to other business units? Do you take part in incident reporting organizations such as FIRST (Forum of Incident Response and Security Teams) at http://www.first.org? Do you inform the FBI or police? Do you notify partners (vendors/customers) that have a connection to your network (and who may be compromised, or from whom the attack originated)? Do you hide the intrusion from the press? Note that the FBI has a webpage for reporting crime at: http://www.usdoj.gov/criminal/cybercrime/reporting.htm

logging procedures

Set up your logging/auditing/monitoring procedures now; one of the most common thoughts after an attack is how much they wished they had adequate logging in the first place in order to figure out what happened.

training/rehearsal

Get training on all these issues. Each person involved needs to understand the scope of what they need to do. Also carry out dry runs. Assume a massive hacker penetration into your network, and drill what happens. Most hacker penetrations succeed because companies practice at being unprepared for their attack.

Since computer networks are growing so fast, there are not enough trained people to handle intrusions. Likewise, networks grow in an ad hoc fashion, so logging/auditing is haphazard. These conditions lead to the state that people don't know what to do when they've been attacked, and their networks aren't robust enough to recover well from the attack.

3.8 How should I respond when somebody tells me they've been hacked from my site?

On the IDS mailing list, someone asked how they should respond to the following e-mail:

```
Below is a log showing a telnet connection from a machine within your
domain.  The machine it connected to does not offer this service publicly so
this can only be assumed to be an IP space probe for vulnerable machines.
We take this matter seriously, and hope that you will as well. Please take
action on this issue as is appropriate and respond to this address with your
 actions.
Nov  6 07:13:13 pbreton in.telnetd[31565]: refused connect from  xx.xx.xx.xx
```

This log entry was likely generated by tcpwrappers, a facility that enhances logging and access control to services on UNIX. It shows an unauthorized attempt from your site to the specified machine. As claimed in the e-mail message, it may be an automated sweep of some sort. The most popular protocols people sweep with are ICMP, FTP, SMTP, NNTP, and Telnet.

In any case, this is evidence of a probe, not an attack. Furthermore, there is no other corroborating evidence. As pointed out by Greg Drew <gdrew at computer dot org> there could be a number of benign reasons:

- Somebody typed "telnet xx.xx.xx.xx" and mistyped the IP address.
- Somebody meant to type "telnet xx.xx.xx.xx 25" to connect to the STMP service in response to receiving spam from the site. The person might have forgotten the "25" or mistyped "23".
- Somebody might have actually done a more extensive scan on the target machines in response to spam. I've personally done light scans before (finger, rusers, etc.) to track down the source of spam.
- May have been an honest mistake (i.e. somebody used to have an account on that machine, but no longer does).

But there are also some nefarious possibilities:

- Your site may have already been hacked, and the hacker is running scans from the compromised machine.
- One of your employees is using the machine to hack (I've worked at a company where this happened -- though since the company made protocol analyzers, it was kinda stupid and they were quickly detected).

<vick at macdoon dot lerc dot nasa dot gov> pointed out another possibility: this might be a social engineering attack. The message asks (commands) you to contact them to describe what actions you have taken. If you do so, it will tell a lot about your network:

- The target is a legal IP address (though not so interesting).
- Your IP address (the above message was likely sent to "postmaster" or some such well-known address, but you will likely respond using your own address.
- Your readiness level: if you come back with a lame response (such as "we can't take action because we have no log files") then they know that your network is prime hacking territory.
- This may be "social engineering spam". The sender of the message may be a company looking to resell intrusion detection products.

Like responding to spam, there is probably little good that can come about responding to this e-mail message (unless you find evidence that some hacker has been using your network as a stepping stone). It probably would be a good idea to check you system logs for the data/time in question, and if you don't have logs, now might be a good time to turn logging on.

As it turns out, the incident was benign. The target network had reconfigured itself, and the "unauthorized" user didn't know about it yet, and wasn't logging in correctly.

3.9 How do I collect enough evidence about the hacker?

An interesting field of IDS is collecting enough information about the incident to identify the hacker. This can be very hard because truely elite hackers will be bouncing their attacks from another compromised system. Hackers will also often employ IP address spoofing, which may appear as if attacks are coming from machines that aren't even turned on.

As far as I can tell, the best technique is to collect as much information as you can. For example, I've put a packet sniffer capturing to tracefiles on our T-1 line saving to files on a 16-gigabyte disk (most any sniffing program on most platforms can do this). You may not think it fun, but I enjoy perusing these files. It's amazing how many TCP/UDP scans and other probes I see on a regular basis.

Likewise, you should make sure you have full auditing and logging enabled on any/all systems exposed to the Internet. These will help you figure out what happened when you were hacked.

4. Products

This section discusses the major network IDS products.

4.1 What freeware/shareware intrusion detection systems are available?

The most complete list on the net seams to be the *COAST Intrusion Detection System Resources page* at http://www.cs.purdue.edu/coast/ids.

See sections 4.4 and 4.5 below for a discussion of some freeware technologies.

4.2 What commercial intrusion detection systems are available?

Note: I've removed the table of info because it has gotten dangerously out-of-date

Reviews can be found at:

- http://www.nwc.com/1023/1023f19.html

Several of these have comments from the vendors themselves that they e-mailed me. Also note that this information can quickly become out of date. The industry has gone through several major changes since I started this document.

The site http://www.internations.net/uk/talisker/ has done a good job of wading through the marketing hype and pulling out the salient points about each of the commercial products.

4.2.0 BlackICE by Network ICE

Vendor comments:

BlackICE has multiple versions. The core is built around "BlackICE Sentry", a full network-based intrusion detection system. There are also host/hybrid versions that run on Windows desktops with a built-in personal firewall.

The list of intrusions it detects is at: http://www.networkice.com/advICE/Intrusions

Distinguishing features of BlackICE Sentry are:

- Full 7-layer, stateful, protocol analysis
- Anti-evasion techniques (handles fragmentation, whisker scans, a whole suite of signature changing attacks)
- Extremely fast, easily handles full 100-mbps bandwidth.

Goto http://www.networkice.com for more information.

4.2.1 CyberCop Monitor by Network Associates, Inc.

Vendor comments:

CyberCop Monitor is a hybrid host/network based IDS that analyzes network traffic to and from the host as well as Windows NT EventLog audit trails and Windows NT authentication activity.

- Developed under the Microsoft Management Console user interface, both CyberCop Monitor and the SMI Console integrate to provide an easy to use graphical interface for local / remote reporting, and remote installation.
- Configuration editor allows for custom settings and thresholds to suit every environment, including security profiles, account groups, time and subnets.
- Extensive filtering using ordered filter rules for each signature.
- Report coalescing feature suppresses denial of service on the IDS itself.
- Report collating of monitoring and scanning information per system with trend analysis options, including 3D charting and graphing from an SQL database.

Goto <http://www.nai.com> for more information.

CyberCop Monitor was written from the ground up by NAI. There is NO connection with the CyberCop Network v.1.0 product developed by Network General/WheelGroup or the Haystack product from TIS - This was aging technology and shelved some months after each subsequent acquisition.

4.2.2 RealSecure by Internet Security Systems (ISS), Inc.

Vendor comments:

Internet Security Systems is the first and only company that has tied both intrusion detection (ISS RealSecure) and vulnerability detection (ISS Internet Scanner) into an integrated security platform for organization to help plan, analyze, and manage their security on a continuous basis. ISS RealSecure is a component of ISS SAFEsuite family of products that cover managing security risk across the enterprise. ISS RealSecure is the market-leader in Intrusion Detection with an integrated host and network based solution. ISS RealSecure comes with over 400 attack signatures with the ability for customers in both the network and host based solution to add or modify their own signatures.

4.2.3 NetRanger by WheelGroup/Cisco

Originally by Wheelgroup, bought by Cisco. It has been recently renamed, though I'm not sure to what. Goto http://www.wheelgroup.com.

4.2.4 eTrust Intrusion Detection by Computer Associates

Formerly Memco/Abirnet/PLATINUM SessionWall, this is now owned by Computer Associates and marketed as *eTrust Intrusion Detection*.

Goto http://www.cai.com/solutions/enterprise/etrust/intrusion_detection.

Originally, SessionWall started out as more of a firewall/content-inspection platform that interposed itself in the stream of traffic. I'm not sure where it is now.

4.2.6 NetProwler by Axent

Goto http://www.axent.com.

4.2.7 Centrax by Cybersafe

Goto http://www.cybersafe.com/solutions/centrax.html.

4.2.9 NFR by Network Flight Recorder

Vendor comments:

NFR is available in multiple forms: a freeware/research version (see below), the "NFR Intrusion Detection Appliance" which comes as bootable CD-ROM, and bundles from 3rd party resellers that add their own features on top of it (like Anzen).

One of the popular features of NFR is "N-code", a fully featured programming language optimized for intrusion detection style capabilities. They have a fulll SMTP parser written in the N-code. Most other systems have either simply add signatures or force you to use raw C programming. Numerous N-code scripts are downloadable from the Internet from sources such as L0pht.

NFR does more statistical analysis than other systems. The N-code system allows easy additions into this generic statistical machine.

A general description can be found at http://www.nfr.net/forum/publications/LISA-97.htm

4.2.10 Dragon by Security Wizards

Goto http://www.network-defense.com

4.3 What is a "network grep" system?

A "network grep" system is based around raw packet capture pumped through a "regular expression" parser that finds patterns in the network traffic. An example pattern would be: "`/cgi-bin/phf`", which would indicate an attempt to exploit the vulnerable CGI script called "phf". Once building such a system, you would then analyze well-known attacks, extract strings specific to those attacks, and add them to your databse of patterns. See http://www.packetfactory.net/ngrep/ for an example.

"Regexp" (regular expression) is a common pattern-matching language in the UNIX environment. While it has traditionally been used for searching text files, it can also be used for arbitrary binary data. In truth, such systems have more flexible matching criteria, such as finding ports or matching TCP flags.

"libpcap" (library for packet capture) is a common library available for UNIX systems that "sniffs" packets off a wire. Most UNIX-based intrusion detection systems (of any kind) use libpcap, though many also have optimized drivers for a small subset of platforms.

The source code for both modules is freely available. A large number of intrusion detection systems simply feed the output of libpcap (or tcpdump) into the regular expression parse, where the expressions come from a file on the disk. Some even simpler systems don't even use regular expressions and simply compare packets with well-known byte patterns. If you want to build a system like this yourself, read up on 'tcpdump' and regular expressions. To understand libpcap/tcpdump, the following document will be helpful: http://www.robertgraham.com/pubs/sniffing-faq.html.

This class of intrusion detection system has one advantage: it is the easiest to update. Products of this class will consistently have the largest number of "signatures" and be the fastest time-to-market for detecting new popular attack "scripts".

However, while such systems may bost the largest number of "signatures", they detect the fewest number of "serious" intrusions. For example, the 8 bytes "CE63D1D2 16E713CF" when seen at the start of UDP data indicates Back Orifice traffic with the default password. Even though 80% of Back Orifice attacks use the default password, the other 20% use different passwords and would not be detected by the system. For example, changing the Back Orifice password to "evade" would change the pattern to "8E42A52C 0666BC4A", and would go undetected by "network grep" systems.

Some of these systems do not reassemble IP datagrams or TCP streams. Again, a hacker could simply reconfigure the MTU size on the machine in order to evade regexp-pcap systems.

Such systems result in larger numbers of false positives. In the BackOrifice example above, the 64-bit pattern is not so uncommon that it won't be seen in other traffic. This will cause alarms to go off even when no Back Orifice is present.

Systems based upon protocol analysis do not have these problems. They catch all instances of the attack, not just the common varieties; they result in fewer false positives; and they often are able to run faster because a protocol decode doesn't have to "search" a frame. They are also able to more fully diagnose the problem; for example distinguish between a "Back Orifice PING" (which is harmless) and a "Back Orifice compromise" (which is an extreme condition). On the other hand, it can often take a week to add a new protocol analysis signature (rather than hours) due to the design and testing involved. Also, overly-agressive attempts to reduce false positives also leads to missing real attacks in some cases.

However, such systems have an advantage over protocol analysis systems. Because they do not have pre-conceived notion about what network traffic is supposed to look like, they can often detect attacks that other systems might miss. For example, if a company is running a POP3 server on a different port, it is likely that protocol analysis systems will not recognize the traffic as POP3. Therefore, any attacks against the port will go undetected. On the other hand, a network-grep style system doesn't necessarily care about port numbers and will check for the same signatures regardless of ports.

4.3.1 Dragon

See above.

4.3.2 Bro

Vern Paxson's Bro intrusion detection system. Vern Paxson wrote large portions of libpcap that many other intrusion detection systems are based on (like NFR and Dragon). I haven't heard of anyone actually using Bro itself. Read the paper http://ftp.ee.lbl.gov/papers/bro-usenix98-revised.ps.Z for more information.

4.3.3 Snort

http://www.clark.net/~roesch/security.html

Snort has recently become very popular, and is considered really cool by a lot of people. It contains over 100 of its own signatures, and others can be found on the Internet.

Following is an example rule:

```
# here's an example of PHF attack detection where just a straight text string
# is searched for in the app layer
alert tcp any any -> 192.168.1.0/24 80 (msg:"PHF attempt"; content:"/cgi-bin/phf";)
```

It says to alert an a TCP connection from any IP address and any port to the 192.168.1.x subnet to port 80. It searches for the content "/cgi-bin/phf" anywhere in the content. If it find such content, it will alert the console with a message "PHF attempt".

Usage of snort is usually done in the following manner:

- BPF filters (part of libpcap) are configured to narrow down the focus to cetain types of traffic.
- A decision is made about which IP addresses are internal and which are external to further narrow down the focus.
- Rules are edited to fit the local environment.
- System runs
- Rules are further edited to remove false positives.

Also, snort has a number of options to be used just to sniff network traffic.

Rules:

- http://snort.rapidnet.com/
- http://www.whitehats.com

4.3.4 Argus

Argus isn't an intrusion detection system itself. However, it monitors packets off the wire and generates logfile events. You can then process those log entries (or peruse them yourself) to find intrusions.

See ftp://coast.cs.purdue.edu/pub/tools/unix/argus for more info. Also see ftp://ftp.sei.cmu.edu/pub/argus-1.5

4.4 What tools do intruders use to break into my systems?

4.4.1 UNIX utilities

These utilities either come with your favorite UNIX platform or you can download them for free.

ping
: to see if a host is alive.

traceroute
: to find the route to the host

nslookup/dig
: to discover all your DNS information

whois
: finds out Internic registration information

finger
: finds out who is logged in and info about users

rpcinfo
: finds out what RPC services are running

showmount
: display shares on a machine

SAMBA
: displays info about WinNT SMB shares

telnet
: the granddaddy of them all -- allows you to connect and play with any text-based protocol (HTTP, FTP, SMTP, etc.)

4.4.2 WinNT utilities

All of the UNIX utilities mentioned above can be used with WinNT. There are also some WinNT specific ones.

nbtstat
: discovers NetBIOS information on remote machine

net view
: is the LANMAN program that allows you to remotely view WinNT shares

4.4.3 Hacking-specific utilities

The standard toolkit for a intruder.

netcat
: is characterized as a "TCP/IP" Swiss Army Knife, allows intruders to script protocol interactions, especially text-based protocols.

crack / NTcrack / L0phtCrack / etc.
: that crack network passwords (Dictionary or Brute Force). These packages also contain utilities for dumping passwords out of databases and sniffing them off the wire.

Sniffing utilities
: for watching raw network traffic, such as **Gobbler**, **tcpdump**, or even an honest-to-god Network Associates **Sniffer© Network Analyzer**

TCP and UDP port scanners
: for scanning/strobing/probing which TCP ports are available. TCP port-scanners can also run in a number of stealth modes to evade/elude loggers.

Ping sweepers
: for pinging large numbers of machines to see which ones are active.

Exploit packs
: which are a set of one or more programs that know how to exploit holes on systems (usually, once the user is logged in).

Remote security auditors
: such as SATAN that look for a number of well known holes in machines all across the network.

War dialers
: that dial lots of phone numbers looking for dial-in ports.

NAT
: is based upon the SAMBA code, and is useful for discovering NetBIOS/SMB info from Windows and SAMBA servers.

Scanners

are programs (like SATAN, ISS, CyberCop Scanner) that probe the system for vulnerabilities. That have a huge number of vulnerabilities they check for and are generally automated, giving the hacker that highest return for the minimal effort.

4.5 What other free/shareware intrusion detection products should I be aware of?

4.5.0 NFR, Research Version

The "NFR Research Version" is a configurable toolkit, available from the Internet for research and noncommercial use. It is "as is" software that requires expertise from the end user to install and configure. It is not a "plug and play" intrusion detection system. (quote from NFR)

See above for info on the commercial version.

4.5.1 tcpwrappers

Tcpwrappers are an add-in for UNIX, and sit between `inetd` and services (like ftp, telnet, etc.). The `inetd` will first call tcpwrappers, which will do some authentication (by IP address) and logging. Then, tcpwrappers will call the actual service, if need be.

4.5.2 IDS for Checkpoint Firewalls

Log file analysis of firewalls is very similar to network analysis. See http://www.enteract.com/~lspitz/intrusion.html for an example.

4.5.3 Shadow

I think it is a project used in the Navy to track intrusions, and generate reports on them. They have an interesting report at http://www.nswc.navy.mil/ISSEC/CID/co-ordinated_analysis.txt where they describe coordinated, slow attacks they have detected using this system.

4.5.4 AAFID

Purdue's COAST distributed agent idea. I'm not sure how much of this is proposals, and how much is real.

4.6 Are there NIDS available for my host?

A new class of NIDS runs on hosts in non-promiscuous mode.

4.6.1 Network ICE / BlackICE Defender

The first such system was BlackICE Defender from Network ICE released in mid-1999. The system also contains a personal firewall. It runs on Win95, Win98, WinNT, and Win2k. It is targetted at both end-nodes and servers.

4.6.2 Network Associates / CyberCop Monitor v2.0

The second system is CCM from Network Associates, released in late 1999. While billed primarily as a "host-based IDS", the majority of the intrusions it detects are network-based. It currently supports WinNT (and presumably Win2k) and they have announced support for Solaris.

4.6.3 CyberSafe / Centrax NNID

In February of 2000, CyberSafe announced their "network node intrusion detection (NNID)". Apparently, versions of their Centrax NIDS come in both promiscuous and non-promiscuous licenses starting with version 2.3.

4.6.4 ISS / RealSecure Micro-Agent

ISS has pre-announced a "Micro-Agent" version of the RealSecure NIDS. The announcement indicates that it will also contain "blocking" features, which presumably will consist of some sort of personal firewall. They have announced that this will be available for both WinNT (and presumably Win2k) as well as Solaris.

6. Resources

6.1 Where can I find updates about new security holes?

6.1.1 CERT (Computer Emergency Response Team)

If it is a security problem, you will eventually see it appear in a CERT advisory. CERT (Computer Emergency Response Team) was set up by a number of universities and DARPA in response to the Morris Worm of 1988. Goto http://www.cert.org.

6.1.2 AUSCERT (AUStralian Computer Emergency Response Team)

AUSCERT is the AUStralian Computer Emergency Response Team. For registration information, see their web site on:

http://www.auscert.org.au/

For more details, contact AUSCERT directly on auscert@auscert.org.au.

6.1.3 CIAC (Computer Incident Advisory Capability) by US Department of Energy

Has a number of useful advisories. Goto http://www.ciac.org/.

6.2 What are some other security and intrusion detection resources?

6.2.1 Purdue's COAST archive

This is the best site on the net for learning about IDS and security in general. See http://www.cs.purdue.edu/coast, http://www.cs.purdue.edu/coast/intrusion-detection, and http://www.cs.purdue.edu/coast/ids.

6.2.2 SANS Institute

I think this may be the best site for security information for people who are not themselves hackers. Their target audience is MIS professionals who have to defend their networks. Goto http://www.sans.org/

6.2.3 L0pht Heavy Industries

These are some hackers with some pretty good tools and useful alerts, targeted at Windows. Goto http://www.l0pht.com

6.2.4 Technical Incursion Countermeasures

I like this site; it has a bunch of well organized info on intrusion (A.K.A. incursion). Goto http://www.ticm.com

6.2.5 IDS mailing list

Email "subscribe ids" to majordomo@uow.edu.au
Email questions to ids-owner@uow.edu.au
This is a nice, low-volume list with interesting discussions from time-to-time.

6.2.6 Michael Sobirey's Intrusion Detection Systems page

http://www-rnks.informatik.tu-cottbus.de/~sobirey/ids.html

6.2.7 advICE database

http://www.networkice.com/advice/Countermeasures/Intrusion%20Detection/default.htm

6.3 What are some sites that are interesting?

Here are some sites that aggregate info from other sites. Could be worth a look.

6.3.1 NIH security site

Goto http://www.alw.nih.gov/Security/

6.3.2 NTSecurity.net

Goto http://www.ntsecurity.net/.

7. IDS and Firewalls

7.2 Why do I need IDS if I already have a firewall?

A common misunderstanding is that firewalls recognize attacks and block them. This is not true.

Firewalls are simply a device that shuts off everything, then turns back on only a few well-chosen items. In a perfect world, systems would already be "locked down" and secure, and firewalls would be unneeded. The reason we have firewalls is precisely because security holes are left open accidentally.

Thus, when installing a firewall, the first thing it does is stops ALL communication. The firewall administrator then carefully adds "rules" that allow specific types of traffic to go through the firewall. For example, a typical corporate firewall allowing access to the Internet would stop all UDP and ICMP datagram traffic, stops incoming TCP connections, but *allows* outgoing TCP connections. This stops all incoming connections from Internet hackers, but still allows internal users to connect in the outgoing direction.

A firewall is simply a fence around you network, with a couple of well chosen gates. A fence has no capability of detecting somebody trying to break in (such as digging a hole underneath it), nor does a fence know if somebody coming through the gate is allowed in. It simply restricts access to the designated points.

In summary, a firewall is not the dynamic defensive system that users imagine it to be. In contrast, an IDS *is* much more of that dynamic system. An IDS *does* recognize attacks against the network that firewalls are unable to see.

For example, in April of 1999, many sites were hacked via a bug in ColdFusion. These sites all had firewalls that restricted access only to the web server at port 80. However, it was the web server that was hacked. Thus, the firewall provided no defense. On the other hand, an intrusion detection system would have discovered the attack, because it matched the signature configured in the system.

Another problem with firewalls is that they are only at the boundary to your network. Roughly 80% of all financial losses due to hacking come from inside the network. A firewall a the perimeter of the network sees nothing going on inside; it only sees that traffic which passes between the internal network and the Internet.

Some reasons for adding IDS to you firewall are:

- Double-checks misconfigured firewalls.

- Catches attacks that firewalls legitimate allow through (such as attacks against web servers).
- Catches attempts that fail.
- Catches insider hacking.

"Defense in depth, and overkill paranoia, are your friends." (quote by Bennett Todd <bet at mordor dot net>). Hackers are much more capable than you think; the more defenses you have, the better. And they still won't protect you from the determined hacker. They will, however, raise the bar on determination needed by the hackers.

7.2.1 How is it that hackers can get through firewalls?

Editors Note: This just clarifies the point above.

Consider bridge building throughout history. As time goes on, technology improves, and bridges are able to span ever larger distances (such as the Golden Gate bridge in SF, whose span is measured in kilometers). Bridge builders are very conservative due to the immense embarassment (not to mention loss of life) should the bridges fail. Therefore, they use much more material (wood, stone, steel) than they need, and they don't create spans nearly as long as they think they can. However, as time goes on, as bridges prove themselves, engineers take more and more risks, until a bridge fails. Then all the engineers become much more conservative again. As has been quoted "It's easy to build a bridge that doesn't fall down; what takes skill is building a bridge that just *barely* doesn't fall down."

In much the same way, most firewall administrators take the conservative approach. It is easy to build a firewall that can't be hacked by being overly conservative and paranoid, and simply turn off all but the absolutely necessary services.

However, in the real world, engineers are not allowed to be sufficiently paranoid. Just like bridge builders want to span ever wider rivers and gorges, corporations want to ever expand the services of the Internet. This puts immense pressure on firewall admins to relax the barriers. This process will continue up to the point where there system is hacked, at which point the corporation will become much more conservative. From this perspective, one could say that corporate dynamics are such that they will generally force the system to the point where it gets hacked.

As every firewall admin knows, the system is under constant attack from the Internet. Hackers from all over the world are constantly probing the system for weaknesses. Moreover, every few months a new security vulnerability is found in popular products, at which point the hackers simply scan the entire Internet looking for people with that hole, causing thousands of websites to be hacked. Such recent holes have been the ColdFusion cfmdocs bug and the Microsoft .htr buffer overflow.

7.3 If I have a intrusion detection, do I need firewall?

Of course. Every corporation needs a well managed, single point of entry.

There are a huge number of "script-kiddies" that are always running automated programs (like SATAN) on the Internet looking for holes. Without a firewall, these automated programs can detect and exploit holes literally in the blink of an eye. Even dial-up users who use the Internet only a few hours a week are getting scanned on a regular basis; high-profile corporate sites will be scanned by script-kiddies much more often.

7.4 Where does the intrusion detection system gets its information? The firewall?

No. While some log file analysis program do scan firewall logs for signs of intrusions, most intrusion detection systems get their information elsewhere.

Remember that firewalls are simple rule-based systems that allow/deny traffic going through them. Even "content inspection" style firewalls do not have the capability to clearly say whether the traffic constitues an attack; they only determine whether it matches their rules or not.

For example, a firewall in front of a web server might block all traffic except for TCP connections to port 80. As far as the firewall is concerned, any port-80 traffic is legitimate. An IDS, on the other hand, examines that same traffic and looks for pattern of attack. An IDS system doesn't really care if the manager decided to allow port 80 and deny the rest: as far as the IDS is concerned, all traffic is suspicious.

This means that an IDS must look at the same source of data as the firewall: namely, the raw network traffic on the wire. If an IDS sat "downstream" from the firewall isntead of side-by-side, it would be limitted to only those things the firewall considered attacks. In the above example, the firewall would never pass port 80 traffic to the IDS.

```
                               +-+                             .
                               |F|  +-----+                    .
                               |I+--+IDS#1|                    .
/============\                 |R|  +-----+    /============\  .
H            H                 |E|             H  corporate H  .
H  internet  H--------+--------+ +------+------H   network  H  .
H            H        |        |W|      |      H         +-----+
\============/     +--v--+     |A|   +--v--+   \=========+IDS#4|
                   |IDS#3|     |L|   |IDS#2|             +-----+
                   +-----+     |L|   +-----+                   .
                               +-+                             .
```

IDS #1
: Few IDSs work this way. Firewalls don't produce enough information in order to effectively detect intrusions.

IDS #2
: This popular placement of an IDS detects attacks that successfully penetrate the firewall.

IDS #3
: This placement detects attacks that are attempted against the firewall.

IDS #4
: By placing intrusion detection systems throughout a corporate network, attacks by insiders will be detected.

8. Implementation Guide

8.1 What questions should I ask my IDS vendor?

CSI (Computer Security Institute) has a good page on this, where they posed questions to IDS vendors, as well as asked them what the difficult questions are. This site is at http://www.gocsi.com/intrusion.htm.

Some common questions are:

What does it cost?
Of course.

What do signature updates and maintanance cost?
Intrusion detection is much like virus protection, a system that hasn't been updated for a year will miss common new attacks.

At what real-world traffic levels does the product become blind, in packets/second?
First, what segments do you plan on putting the IDS onto? If you have only a 1.5-mbps connection to the Internet that you want to monitor, you don't need the fastest performing system. On the other hand, if you are trying to monitor a server farm in your corporation in order to detect internal attacks, a hacker could easily smurf the segment in order to blind the sensor.

The most important metric is **packets/second**. Marketing people use weasle words to say that their products can keep up with a full 100-mbps networks, but that is only under ideal conditions. A Network World did a review in August of 1998 where products failed at roughly 30% network load (50,000 pacets/second). Likewise, Network Computing did a review in September of 1999 with real-world traffic where several products that claimed 100-mbps could still not keep up.

How easy is the product to evade?
Go down the list in section 9.4 and ask the vendor if such activities will evade the IDS. If you want to give the vendor heartburn, ask them about section 9.5 .

How scalable is the IDS system as a whole?
How many sensors does the system support? How big can the database be? What are the traffic levels when forwarding information to the management console? What happens when the management console is overloaded? These are tough questions.

How much will it cost to run and maintain the product?
How good is the reporting architecture? How easy is it to manage false positives? How long does it take to track down alerts and identify the situation? How many people do I need to use this product?

The following questions are commonly asked, but are less likely to produce meaningful answers:

How many signatures does the system support?
Unfortunately, vendors dramatically inflate their signature count. This is the game that all vendors must play, even though it is becoming less and less important.

What intrusion response features does the product have?
A feature like automatically reconfiguring a firewall sounds really cool, but in real life, few security managers implement it. Reconfiguring a corporate firewall is extremely dangerous.

8.2 How do I maintain the system on an on-going basis?

I put this question in here hoping for feedback; I really don't have an answer.

If you install an intrusion detection system, you WILL see intrusions on an on-going basis. In a SOHO environment, you will likely get scanned by a hacker once a week. On a well-known web-site, hackers will probe your site for vulnerabilies many times per day. On a large internal corporate network, you will find constant suspicious activities by internal employees.

The first problem that you are likely to be confronted with is employees surfing p-orn sites on the web. Just about every long-term administrator I know has interesting stories about this. Most don't care about p-orn, it just embarrassing knowing what people are up to.

It is interesting that many otherwise conservative corporations do not outright restrict such surfing -- because it is often the executives themselves that do it. Lower-level engineers detecting such activities usually fear to bring the subject up.

The next problem that engineers face is a Human Resource (HR) issue. You will find users doing things they shouldn't, so a lot of time is spent interfacing with HR working with the offending employee.

The last problem is what to do about Internet script-kiddies and hackers probing your system. Usually, a call to ISP in question or e-mail to their "abuse@" mail box suffices. Sometimes the ISP will be grateful -- because their own systems have been compromised.

Remember that even what appears to be the most egregious hack may, in fact, be innocuous, so aproach other people with dignity and respect.

8.4 How do I stop innapropriate web surfing?

One of the biggest concerns for corporations today is employees surfing "innapropriate" web sites. To some extent, companies are worried about employees wasting company time on the Internet, and to another extent, companies are worried about legal liability, such as when an employee surfs p-orn sites that causes a sexual harassment lawsuit.

However, companies do not like being in the position of being "big brother". Rules against inappropriate surfing inevitably lead to grey areas (for example: Playboy.com recently had an article on computer security, which an employee could easily have stumbled across while doing a legitimate search on the web).

Intrusion detection systems, firewalls, proxy servers, and sniffing programs can be configured to log all web surfing traffic to log files, including *who* accessed *which* websites. Most companies already have these logs, but few make use of this information. Network technicians do not want to take on the role of HR and prosecute people. (In many cases, the culprits are executives and going after them can be a career limiting move (CLM)).

One elegant solution is posting such information to a public internal website. This has been known to dramatically affect inappropriate surfing. Rather than having a central authority judging appropriateness, it leaves it up to the individual to make that judgement.

8.5 How can I build my own IDS (writing code)?

Simple intrusion detection systems are easy to build. Simply grab an input source (log files, network traffic) and pass it through a pattern match (regexp). Along with it, through the same data through some statistical analysis, much like how SETI@Home sends radio noise through some Fourier analysis looking for repeated patterns.

For example, section 4.3 above discusses a "network grep" system that passes network traffic through a pattern match system. Such a system could be built with some knowledge of C and a UNIX system.

Similarly, section 4.5.2 describes a PERL based system that parses log files from a firewall.

8.7 What is the legality of NIDS (since it is a form of wiretap)?

Different countries and states have different laws, but it is generally legal to monitor your OWN traffic for intrusions.

One concern that people have is that running a NIDS on a corporate network results in network managers viewing employee Internet surfing activity (sometimes network managers find top executives surfing porn sites). As the network equipement and the user's workstation belong to the company, the legal precident is that use of the corporate equipment implies consent to monitoring. However, it is recommended that companies explicitly state in employee handbooks that their network activity *will* be monitored. At minimum, it avoid embarrasing situations.

8.8 How do I save logfiles in a tamper-proof way?

The first thing a hacker does is delete/change the logfiles in order to hide evidence of the break in. Therefore, a common need is to have a "write-once" storage system whereby once data is written, it can never be altered.

WORM (Write-Once-Read-Many) drives have historically been used for this purpose, but they are expensive and finnicky. They probably don't have drivers for your system, and you software is likely incompatible with them in other ways (i.e. some systems do alter the files a little bit as they create them, which doesn't work on a worm).

One problem with any system is that entropy sets in. It may be provable secure today, but it is unlikely to stay that way. For example, one technique for logging would be to employ syslog where the receiver doesn't have a TCP/IP stack but instead uses TCPDUMP to save the raw packets to a file (presumably, a utility would be run a later date to reconstruct the syslog entries). From the entropy perspective, there is no guarantee that a TCP/IP stack won't be installed during an update, or when a new person joins the team, or when machines get shuffled around.

To combat such entropy, the model system uses the "snipped-wire" approach. In this model, an extra Ethernet adapter is installed in the machine who is generating the data, and the **receive** wire is cut. If an accident later happens such that the extra adapter is connected to an unsecured network, then few problems are likely to result.

In much the same way, the receiving system should have only a single Ethernet adapter, and its **transmit** wire should be cut. It would be best to also disable the TCP/IP stack and instead force the data through packet sniffing utilities. (Yes, there are attacks that can compromise the system even when no responses are ever received).

Normal TCP/IP won't work in this scenario. You will need to hard-code the route and ARP tables on the generating machine in order to force the traffic out the one-way wire. Similarly, you will need to use special utilities on the receiving machine in order to parse incoming packets back into useful data.

UDP-based transports like 'syslog' and SNMP Traps are the most useful transports in this situation. They are easy to generate on the outgoing machine as they are built into most systems. Since responses aren't generated anyway, it doesn't hamper the normal flow of applications. Likewise, they are easy to parse back into SNMP messages or syslog files on the receiving end, or at least, it is easy to harden a TCP/IP stack to receive only those ports. At very least, TFTP or NFS can be configured to transport files to a TCP/IP stack on the other side.

One problem that goes along with this is data management. You cannot connect the data repository to a network, so anything you use to backup the system must be installed on the system itself.

Personally, the system I use is an old Pentium-90 computer with a 6-gig drive, CD-ROM writer, and a sniffing utility that dumps all the network traffic (a 416-kbps DSL connection) to packet capture files on the disk. A couple simple filters remove a lot of the bulk so downloading the latest RedHat distribution doesn't fill up the disk. I prefer this solution over actual log files because it captures absolutely everything that happens on the wire, even all numerous so-called stealth attempts.

9 What are the limitations of NIDS?

Network intrusion detection systems are unreliable enough that they should be considered only as secondary systems designed to backup the primary security systems.

Primary systems such as firewalls, encryption, and authentication are rock solid. Bugs or misconfiguration often lead to problems in these systems, but the underlying concepts are "provably" accurate.

The underlying concepts bhind NIDS are not absolutely accurate. Intrusion detection systems suffer from the two problems whereby normal traffic causes many false positives (cry wolf), and careful hackers can evade or disable the intrusion detection systems. Indeed, there are many proofs that show how network intrusion detection systems will never be accurate.

This doesn't mean intrusion detection systems are invalid. Hacking is so pervasive on today's networks that people are regularly astounded when they first install such systems (both inside and outside the firewall). Good intrusion detection systems can dramatically improve the security of a site. It just needs to be remembered that intrusion detection systems are backup. The "proveably accurate" systems regularly fail (due to human error), and the "proveably incorrect" systems regularly work.

9.1 Switched network (inherent limitation)

Switched networks (such as 100-mbps and gigabit Ethernet switches) poses dramatic problems to network intrusion detection systems. There is no easy place to "plug in" a sensor in order to see all the traffic.

For example, somebody on the same switched fabric as the CEO has free reign to attack the CEO's machine all day long, such as with a password grinder targetting the File and Print sharing.

There are some solutions to this problem, but not all of them are satisfactory.

Embed IDS within the switch

Some vendors (Cisco, ODS) are imbedding intrusion detection directly into switches. As far as I can tell, however, these IDS systems do not have the broad range of detection as traditional NIDS.

Monitor/span port

Many switches have a "monitor port" for attaching network analyzers. A NIDS can easily be added to this port as well. An obvious problem is that the port runs at a much lower speed than the switch backplane, so the NIDS will not be able to see all the traffic on a heavily loaded switch. Moreover, such ports are often used by sniffers for network management purposes, and must often be swapped out occasionally.

Tap into the cable (for inter-switch or switch-to-node)

A monitor can be connected directly to the cable in order to monitor the traffic. These may be cables between switches or cables from the switch to a host. Different techniques would be:

inline taps
Inline taps are devices that insert themselves directly into the stream of communication and make a copy of it. A typical example would be the Shomiti Century Tap (http://www.shomiti.com/productsf/tapfamilyf.html) which plugs into a 100-mbps full duplex line, and allows a computer equipped with 2 adapters to read both channels.

vampire taps
In the olden days, vampire taps were a mainstay of thick coax Ethernet, and were the preferred way of connecting end-nodes to the network.

inductance taps
Most taps can be detected with TDR (Time Domain Reflectometer) equipement. Inductance taps do change the cable in any way, but instead site on the outside and monitor the electromagnetic noise emitted by the cables. Only used by spies.

The problem with tapping into the cable, especially those between switches, is that they generate huge amounts of traffic. Most NIDS cannot handle very high loads before going "blind".

Thanks to Christopher Zarcone < czarcone at acm dot org > for this info.

Host-based sensors

From a conceptual point of view, the only way to defeat the resource limitations of switched networks is to distribute host-based intrusion detection. Several host-based agents, such as BlackICE and CyberCop Monitor, contain a network-based component that monitors only that host's traffic. Others do the traditional logfile and audit analysis.

9.2 Resource limitations

Network intrusion detection systems sit at centralized locations on the network. They must be able to keep up with, analyze, and store information generated by potentially thousands of machines. It must emulate the combined entity of all the machines sending traffic through its segment. Obviously, it cannot do this fully, and must take short cuts.

This section lists some typical resource issues.

9.2.1 Network traffic loads

Current NIDS have trouble keeping up with fully loaded segments. The average website has a frame size of around 180-bytes, which translates to about 50,000 packets/second on a 100-mbps Ethernet. Most IDS units cannot keep up with this speed. Most customers have less than this, but it can still occasionally be a concern.

When buying an IDS, ask the vendor how many packets/second the system can handle. Many vendors will try to tell you how many bits/second, but per-packet is the real performance bottleneck. Virtually all vendors can handle 100-mbps traffic using 1500-byte packets, few can handle 100-mbps traffic using 60-byte packets.

9.2.2 TCP connections

IDS must maintain connection state for a large number of TCP connections. This requires extensive amount of memory. The problem is exacerbated by evasion techniques, often requiring the IDS to maintain connection information even after the client/server have closed it.

When buying an IDS, ask the vendor how many simultaneous TCP connections it can handle.

9.2.3 Other state information

TCP is the simplest example of state information that must be kept by the IDS in memory, but other examples include IP fragments, TCP scan information, and ARP tables.

9.2.4 Long term state

A classic problem is "slow scans", where the attacker scans the system very slowly. The IDS is unable to store that much information over that long a time, so is unable to match the data together.

9.3 Attacks against the NIDS

The intrusion detection system itself can be attacked in the following ways.

9.3.1 Blind the sensor

Network intrusion detection systems are generally built as "passive monitors" from COTS (commercial-off-the-shelf) computers. The monitors are placed alongside the networking stream, not in the middle. This means that if they cannot keep up with the high rates of traffic, they have no way to throttle it back. They must start dropping packets. This is known as trying to drink from a firehose. Few NIDS today can keep up with a fully saturated 100-mbps link (where "saturated" means average sized packets of 180 bytes, which is roughly 50,000 packets/second).

Not only will the sensor start dropping packets is cannot process, high traffic rates can completely shut down the sensor. For example, consider a sensor that can process a maximum of 20,000 frames/second. When the proferred load is 40,000 frames/second, it usually drops actual processing down to 10,000 frames/second or 5,000 frames/second, or maybe even zero. This is because frame reception and frame analysis are two different acitivities. Most architectures require the system to capture the packet even when it is too busy to analyze it, which takes even more time away from analysis.

Therefore, an intruder can attack the sensor by saturating the link. If the intruder is local, he/she can simply use a transmit program. A 400-Mhz box can fully saturate a link with 60-byte packets, breaking most IDS systems that might be attached to the system.

A remote attacker can execute smurf or fraggle attacks, likewise saturating links. It is unlikely an attacker will have a fast enough link themselves (100-mbps is quite rare) in order to be able to attack head-on in this manner.

9.3.2 Blind the event storage (snow blind)

The 'nmap' port scanning tool contains a feature known as "decoy" scans. It scans using hundreds of spoofed source addresses as well as the real IP address of the attacker. It therefore becomes an improbable task for the administrator to find discover which of the IP addresses was real, and which was one of the decoy addresses.

Any attack can be built from the same components. A massive attack with spoofed addresses can always hide a real attack inserted somewhere inside. Administrators would be hard pressed to discover the real attack inside of all that noise.

These two scenarios still retain forensics data, though. If the attacker is suspected, the data is still there to find. Another attack is to fill up event storage. When the database fills up, no more attacks will be discovered, or older attacks will be deleted. Either way, no evidence exists anywhere that will point to the intruder.

9.3.3 DoS (Denial of Service)

A NIDS is an extremely complex system, equivelent in complexity to an entire TCP/IP stack running numerous services. This means the NIDS is susceptible to such attacks as SYN floods and smurf attacks.

Moreover, the numerous protocols NIDS analyze leave them open to outright crashes when unexpected traffic is seen. Attackers can often buy the same intrusion detection systems used by their victim, then experiment in many ways in order to find packets that will kill the IDS. Then during the attack, the intruder kills the IDS, then continues undetected.

9.4 Simple evasion

This section describes simple evasion tactics that fool basic intrusion detection systems. The next section will describe advanced measures.

9.4.1 Fragmentation

Fragmentation is the ability to break up a single IP packet into multiple smaller packets. The receiving TCP/IP stack then reassembles the data back again before forwarding the data back up to the application. Most intrusion detection systems do not have the ability to reassemble IP packets. Therefore, there exist simple tools (like fragrouter) that can auto-fragment attacks in order to evade IDS.

Note that fragmenting the IP packets in the middle of the TCP header has long been used to evade firewall port filtering.

Some industrial grade NIDS can reassemble traffic. Also, some firewalls can "normalize" traffic by forcing reassembly before passing the traffic through to the other end.

9.4.2 Avoiding defaults

People often use firewalls as easy NIDS, where they make assumptions that the destination port uniquely identifies the protocol. A hacker who successfully installs a backdoor can run standard protocols on non-default ports. For example, a hacker may send a user a Back Orifice infected program, but change the port from the default of 31337. Most intrusion detection systems will no longer identify the traffic correctly (though a few do).

9.4.3 Slow scans

Because of the volume of traffic on the wire, NIDS have difficulty maintaining long-term traffic logs. It is therefore difficult to detect "slow scans" (ping sweeps or port-scans) where intruders scan one port/ address every hour.

9.4.4 Coordinated, low-bandwidth attacks

Sometimes hackers get together and run a slow scan from multiple IP addresses. This make it difficult for an intrusion detection system to correlate the information.

9.4.5 Address spoofing/proxying

One goal of intrusion detection is to point fingers at who is attacking you. This can be difficult for a number of reasons. In 'Smurf' attack, for example, you receive thousands of replies from a packet that you never sent. The NIDS and detect those replies, but cannot discover who sent the forged packet. In TCP Sequence Number Prediction, forged IP addresses are used so that the NIDS does not know precisely where the intruder is coming from. Finally, most intruders will 'bounce' their attacks via FTP or Web proxies, or stage their attacks from other sites they have broken into. Thus, it will be very difficult to find out who is attacking your site, and configuring IP address filters in your firewall won't help.

9.4.6 Pattern change evasion

Many simple network intrusion detection systems rely upon "pattern matching". Attack scripts have well know patterns, so simply compiling a database of the output of known attack scripts provides pretty good detection, but can easily be evaded by simply changing the script.

For example, some POP3 servers are vulnerable to a buffer overflow when a long password is entered. There exist several popular attack scripts for this vulnerability. One intrusion detection system might contain 10 patterns to match match the 10 most common scripts, while another intrusion detection system looks at the password field and alarms when more than 100 bytes have been entered. The first system is easy to evade simply by changing the attack script, while the second system catches any attack on this point.

The typical example is simple changes to the URL. For example, this document can be retrieved through the URL: http://www.robertgraham.com/pubs/./network-intrusion-detection%2Ehtml. Even though the exact pattern has changed, the *meaning* hasn't been altered. A NIDS looking for the original URL on the wire won't detect this alered one unless it has anti-evasion countermeasures.

9.5 Complex evasion

Talented hackers can direct their attacks at their victims in ways to bypass intrusion detection systems. An early paper by Vern Paxson on his NIDS called "Bro" describes some of these problems. The original PostScript version is at ftp://ftp.ee.lbl.gov/papers/bro-usenix98-revised.ps.Z.

The seminal paper on network intrusion detection "evasion" was written by Thomas H. Ptacek and Timothy N. Newsham. The original PostScript version is available at http://www.aciri.org/vern/Ptacek-Newsham-Evasion-98.ps, while an HTML mirror is available at http://www.robertgraham.com/mirror/Ptacek-Newsham-Evasion-98.html. Thomas H. Ptacek claims that many/most of the commercial products still (October 1999) have serious problems in this regard. Much this this section summarizes these two papers.

These papers describe the abstract concept that the network model used by the network intrusion detection system is different than the real world.

For example, an intruder might send a TCP FYN packet that the NIDS sees, but which the victim host never sees. This causes the NIDS to believe the connection is closed, but when in fact it isn't. Since TCP connections do not send "keep-alives", the intruder could wait hours or days after this "close" before continuing the attack. In practice, most interesting services do kill the connection after a certain time with no activity, but the inruder still can cause a wait of several minutes before continuing.

The first such attack is to find a way to pass packets as far as the NIDS, but cause a later router to drop packets. This depends upon the router configuration, but typical examples include low TTL fields, fragmentation, source routing, and other IP options. If there is a slow link past the NIDS, then the hacker can flood the link with high priority IP packets, and send the TCP FIN as a low priority packet -- the router's queuing mechanism will likely drop the packet.

Another approach is to consider what the host will or will not accept. For example, different TCP stacks behave differently to slightly invalid input (which programs like 'nmap' and 'queso' use to fingerprint operating systems). Typical ways of causing different traffic to be accepted/rejected is to send TCP options, cause timeouts to occur for IP fragments or TCP segments, overlap fragments/segments, send slight wrong values in TCP flags or sequence numbers.

The Ptacek/Newsham paper concentrated on IP fragmentation and TCP segmentation problems in order to highlight bugs in IDSs. For example, they noted that if overlapping fragments are sent with different data, some systems prefer the data from the first fragment(WinNT, Solaris), whereas others keep the data from the last fragment (Linux, BSD). The NIDS has no way of knowing which the end-node will accept, and may guess wrong.

Their TCP connection analysis was even more in depth, discussing ways of "de-synchronizing" TCP connections, which are much more fragile than one would think. Again, the IDS cannot correctly model all possible TCP/IP stack behavior and figure out what the end-node will accept as data. TCP also has the overlap problems that IP fragmentation has. For example, intrusion detection systems might accept the first segment and ignore later segments, but most hosts accept the later segmetns.

They ran tests against various intrusion detection systems in order to figure out if they could evade intrusion detection systems. Their results were dismal -- one major intrusion detection system could be completely evaded simply by fragmenting packets, others could be thrown off by "desynchronizing" from the data the end-node would accept.

9.9 Tools

The following tools may help in evaluating IDS systems for these problems.

Anzen NIDSbench

http://www.anzen.com/research/nidsbench/.

Contains the "fragrouter" that forces all traffic to fragment, which demonstrates how easy it is for hackers/crackers to do the same in order to evade intrusion detection. This accepts incoming traffic then fragments it according to various rules (IP fragmentation with various sizes and overlap, TCP segmentation again with various sizes and overlaps, TCP insertion in order to de-synchronize the connection, etc.).

Also contains the "tcpreplay" program, which dumps high loads onto an Ethernet segment in order to veriy a NIDS can keep up.

CASL

NAI's CyberCop Scanner comes with CASL built in. This was used in the Insertion/Evasion paper above to carry out validation tests. It allows scripting of low-level TCP/IP packets.

Some scripts for CASL are at: http://www.roses-labs.com/labs/labs.htm

10. Misc.

10.1 What are some standardization/interoperability efforts?

10.1.1 COAST audit trails format

A much more narrowly defined effort that solves a specific problem. Hasn't produce proposals yet. See http://www.cs.purdue.edu/coast/projects/audit-trails-format.html

10.1.2 Universal Format for Logger Messages

See http://www.ietf.org/internet-drafts/draft-abela-ulm-04.txt

10.1.3 IETF Intrusion Detection Working Group

Charter: http://www.ietf.org/html.charters/idwg-charter.html
Archive: http://www.semper.org/idwg-public/
Subscribe: idwg-public-request@zurich.ibm.com

10.1.4 CIDF (Common Intrusion Detection Framework)

Has specified a lisp-like format for messages between "Event Generators", "Event Analyzers", "Event Databases", and "Response Units". Currently very theoretical with little industry input.

10.1.5 SAF (Security Advisory Format)

An attempt to standardize security advisories, such as those that come from CERT, the FBI, etc. http://www.ietf.org/internet-drafts/draft-debeaupuis-saf-00.txt

10.1.6 Mitre's CVE effort

"Common Vulnerabilities and Exposures (CVE)" - aims to standardize the names for all publicly known vulnerabilities and security exposures. While this is primarily an academic effort, it does have some vendor input from the major vulnerability assessment and IDS vendors.

The CVE effort is best thought of as a "concordance": it allows people to sync up between the various advisories and IDS/scanner checks. It solves the problem that different products detect such things differently. For example, one intrusion detection system might detect a buffer overflow by examining the length of a field, and therefore map to multiple CVE entries and advisories for different products that have buffer overflows in the same field. Likewise, another IDS system might match the signatures of specific exploits (from published scripts) of a single vulnerability.

Therefore, there might be one-to-many, many-to-one, or many-to-many mappings between any product or set of advisories. The CVE provides a concordance between various systems.

http://www.cve.mitre.org/

11. Honeypots and Deception Systems

While not strictly sniffer-based intrusion detection systems, honeypots still process network protocols in much the same ways. Therefore, I've decided to add this section to my FAQ.

11.1 What is a honeypot?

A *honeypot* is a system designed to look like something that an intruder can hack. Examples can be:

- Installing a machine on the network with no particular purpose other than to log all attempted access.
- Installing an older unpatched operating system on a machine. For example, the default installation of WinNT 4 with IIS 4 can be hacked using several different techniques. A standard intrusion detection system can then be used to log hacks directed against the machine, and further track what the intruder attempts to do with the system once it is compromised.
- Install special software designed for this purpose. It has the advantage of making it look like the intruder is successful without really allowing them access.
- Any existing system can be "honeypot-ized". For example, on WinNT, it is possible to rename the default "administrator" account, then create a dummy account called "administrator" with no password. WinNT allows extensive logging of a person's activities, so this honeypot will track users attempting to gain adminstrator access and exploit that access.

11.2 What are the advantages of a honeypot?

- An early-alarm that will trip only upon hostile activity. Network intrusion detection systems have a problem distinguishing hostile traffic from benign traffic. Isolated honeypots have a much easier time because they are systems that should not normally be accessed. This means that *all* traffic to a honeypot system is already suspect. Network management discovery tools and vulnerability assessment tools still cause false positives, but they otherwise give a better detection rate.
- A hostile-intent assessment system. Honeypots often present themselves as easily hacked systems. One of the most common things hackers do is scan the Internet doing "banner checks". The honeypot can be setup to provide a banner that looks like a system that can easily be hacked, then to trigger is somebody actually does the hack. For example, the POP3 service reports the version of the software. Several versions of well-known packages have buffer-overflow holes. A hacker connections to port 110, grabs the version info from the banner, then looks up the version in a table that points to which exploit script can be used to break into the system.

11.3 What are the disadvantages of a honeypot?

- If the system does indeed get hacked, it can be used as a stepping stone to further compromise the network.
- Some people believe that since honeypots lure hackers in, that legal rights to prosecute hackers are

reduced. This is a misconception, because honeypots are not active lures -- they do not advertise themselves. A hacker can only find a honeypot in the first place by running search programs on a network.
- Honeypots add complexity. In security, complexity is bad: it leads to increased exposure to exploits.
- Honepots must be maintained just like any other networking equipment/services. This leads many people to turn them off after a while. You think that a 468 running RedHat Linux 4.2 that you setup 2 years ago doesn't require maintainance, but in reality it does. How do you know the logging is working right? What do you do when a new network management platform or vulnerability assessment system starts being used and alarmas start going off? What do you do when alarms stop coming in because a hacker has compromised the system and is using it launch other attacks against you (or worse, back out to the Internet)?

11.4 How can I setup my own honepot?

The thing to remember is that setting up honepots is really easy. While honeypot products are cool, virtually any existing hardware/software can be setup to be your honeypot.

Your gameplan should consist of the following steps:

documentation, documentation, documentation
The first step in any network management endeavor (actually, the last step when people discover the pain of not having done it in the first place).

maintainance plan
How do you plan on maintaining it?

alarm reporting
How do you plan on receiving alarms from the system?

reaction plan
What do you plan on doing when an alarm goes off?

11.5 What are the types of honeypots?

Port monitors
The simplest honeypot is simply a sockets-based program that opens up a listening port. A typical example of this is the program *NukeNabber* (for Windows) that listens on ports typically scanned for by hackers. This alerts the user that they are being scanned. The disadvantage of these programs are:
- In most cases where they are used, it is actually better to setup a personal firewall to block access from the attacker. Port monitors don't log any better than firewall would.
- They alert the hacker that such a system is running because they first accept, then drop, the connection.

Deception systems
The next logical step beyond the port monitor is a system that actually interacts with the hacker. For example, rather than simply accepting port 110 for POP3, then dropping it, a deception system will actually respond as if it were a POP3 server. It may give a generic banner, or it may generate a banner with a version number that hackers know they can hack. Since 99% of attacks against POP3 are buffer-overruns in the username or passwords, most deception systems only implement that portion of the protocol. Likewise, most deception systems implement only as much of the protocol machine as necessary to trap 90% of the attacks against the protocol.

Multi-protocol deception systems

Packages like Specter or Fred Cohen's Deception Toolkit offer most of the commonly hacked protocols in a single toolkit. Likewise, these systems come with multiple banners in order to emulate packages for different operating systems.

Full systems

Beyond products targetted directly at deception, you can also implement full systems. Most systems have the ability to alert on exception conditions. By using the native logging/auditing built into such

Full systems plus NIDS

Along with the full system mentioned above, you might want to include a full intrusion detection system to supplement the internal logging.

11.6 What are the pros/cons of setting up a system that can be hacked?

The three most commonly hacked servers on the net are unpatched systems running older Linux (like RedHat 5.0), Solaris 2.6, and Microsoft IIS 4.0. Therefore, as part of your honeypot plan, you might want to setup one or all three of these systems.

Remember: if you put one of these systems on the Internet, within a month it will be discovered and hacked.

Pros:

Learn about incidence response

Most people believe "it can't happen to them", and are unprepared when it does. Setting up systems that hackers break into will teach you about how to detect hacker breakins and how to clean up after them.

Learn about hacking techniques

Watching hackers break into your system teaches you a lot about hacking.

If you need a secure system inside your company (for example, one that holds financial information), setup a similar system outside your company with bogus data. If a hacker compromises that system, you'll learn how to protect the one inside your company from similar exploits.

Early warning systems

Setting up servers inside your company that can easily be hacked will alert you to hostile activity long before real systems get compromised. Hackers try the simpler techniques first before moving on to harder ways of breaking into system. Therefore, setting up an easily hacked system will clearly indicate the hostile intent of somebody.

Cons:

Launching Point

The biggest danger is that somebody could use that system to launch further attacks against either you or other people. In particular, there might be legal considerations when a system you control attacks a third party.

11.7 Are there examples of people using honeypots?

The book The Cuckoo's Egg by *Clifford Stoll* is an engaging story about a researcher who bumbles his way into tracking down a hacker who was abusing the university's computer systems. The researcher basically left the system open and vulnerable for about a year in order to track the hacker's activities.

The San Diego Supercomputer Center has left machines up that can be hacked. http://security.sdsc.edu/incidents/worm.2000.01.18.shtml

11.8 What honeypot products are available?

The following are products that I know of.

Fred Cohen's Deception Toolkit
http://www.all.net/dtk/

Specter
http://www.specter.ch/

NAI CyberCop Sting
http://www.nai.com/

netcat
The netcat tool can be used to respond with deceptive banners.

11.9 What are deception countermeasures?

Beyond honeypots in particular, you can setup "deception countermeasures". Your network "leaks" lots of information about itself, which hackers in turn use to break into your network. Therefore, if you leaks deceptive information about you network, then you'll at minimum misdirect your attackers, but hopefully trigger alerts.

I personally have done the following sorts of things:

E-mail headers
A classic problem on the web is that e-mail systems insert the IP address of the system sending the message to it. If you are inside a corporation and send e-mail out, you reveal internal e-mail servers. If you are using a free e-mail system like Yahoo mail or Hotmail, the IP address of the machine you used to send the mail is included in the header. This process can go several level deep as e-mail inside companies often travel several hops through gateway, firewalls, and anti-virus content scanners. It's difficult, but you can reprogram things in order to insert bogus IP addresses in to the headers.

DNS info
One of the first things a hacker will do against you is a DNS Zone Transfer. Many admins blocks access to TCP port 53 to stop this (though that breaks other DNS services). By inserting bogus machines or even entire bogus subdomains you misdirect the hacker. For example, I could setup a machine called "bogus.robertgraham.com" with an IP address of 192.0.2.132, then tell my IDS to trigger whenever it sees traffic to that address. Since my IDS already triggers on Zone Transfers, this'll catch somebody who is seriously trying to scope out my network.

anti-sniffers
Are you certain that your ISP isn't sniffing you? Well, in order to find out, setup machines elsewhere on the Internet to connect to some of your boxes using clear-text passwords. Then setup your IDS to trigger when anybody else uses those passwords. This is best used with a honeypot that doesn't have real services. For example, I've setup a virtual Telnet daemon on that another machines logs into every once-and-a-while. I've setup the IDS to trigger if anybody but that machine logs in using that account

name. When they log in, they will soon find out it isn't real account.

anti-sniffers, part deux

Similar to above, you can transfer password files across the network that contain easily crackable passwords, then have the IDS trigger whenever anybody attempts to login. For example, setup a batch file that regularly transfers files via FTP, one of which is /etc/passwd. This will tell you if anybody has sniffed that file.

11.10 What are the legal implication of honeypots?

11.10.1 Do honeypots constitute entrapment?

No. This is the most commonly asked question about honeypots, and the answer is a clear no. Entrapment has a clear legal definition whereby law enforcement officers encourage somebody to a commit a crime that they were not otherwise disposed to do. This means:

- If you are not a law enforcement officer, you cannot entrap.
- Affording the means for somebody to commit a crime is not the same as encouraging the crime. The FBI can setup a honeypot without risk of entrapment.
- If the FBI contacts somebody in alt.2600 and posts a bounty for cracking into a system, then it *would* be entrapment.

11.10.2 Am I aiding and abetting a crime?

Possibly. You are centainly not abetting the person breaking into your system. However, if the he/she uses your system to launch attacks against other systems, you might be partially liable for the actions.

http://www.nylj.com/stories/00/03/030200a5.htm

[fin]