Detecting Intrusions

Methods of detecting intrusion attempts and security breaches

Allaire Security White Papers Series

(Version 1.0)

Abstract

Title	Detecting Intrusions
Date	January 8, 2001
Product	None
Target Audience	Network Administrators
Abstract	Administering a networking environment can be a difficult process without having to worry about someone breaking into your network. This document and other lockdown documents are Allaire's effort toward making this job a little easier.

Allaire Corporation • One Riverside Center • 275 Grove Street • Newton • MA • 02466
www.allaire.com • info@allaire.com • (617) 219-2000 •
security issues: secure@allaire.com
document feedback: lockdown@neohapsis.com

Table of Contents

Detecting intrusions and intrusion attempts

This document is targeted at the system administrator who has been tasked with network security responsibilities. This document is broken into three primary sections:

- An overview of what an intruder can learn about your network, and how to prevent some of that exposure.
- The signs of being "probed" and how they can be detected
- OS specific considerations (UNIX and NT)

There is also a list of external web resources on the subject for further reading.

The anatomy of an attack

Before looking for the trails left by potential intruders it's important to understand intrusion methodology. Granted, techniques and levels of sophistication vary, but a general methodology is typically followed. By first understanding attack methodology, administrators can identify both intrusion attempts and network exposures with greater understanding.

DNS Tricks

Attackers who possess only minor details about a targeted network will start with basic reconnaissance techniques. They often begin by querying public information databases, such as ARIN or InterNIC for network information. For example, performing an ARIN query on a (fictitious) network will result in:

```
[user@server user]$ whois nobody@rs.arin.net
[rs.arin.net]
Nobody (NETBLK-NOBODY-21)     NOBODY-21       10.2.34.0 – 10.2.34.255
Nobody (NETBLK-NOBODY-25)     NOBODY-25       10.2.35.0 – 10.2.35.255
Nobody (NETBLK-NOBODY-61)     NOBODY-61       10.3.9.0 – 10.3.9.255
[user@server user]$
```

The results of this query give attackers relevant information detailing network address blocks. Unfortunately, there is no way to remove your organization from these databases, so work under the assumption that attackers **will** discover your IP address range. (*Note: the use of nobody.com and provider.net are fictitious and are used purely for example.*)

Next, attackers will frequently target public accessible services (web, mail, ftp, etc.) in an effort to retrieve more information about your network. Using the DNS utility dig we can demonstrate this:

```
[user@server user]$ dig nobody.com mx

; <<>> DiG 8.2 <<>> nobody.com mx
;; QUERY SECTION:
;;      nobody.com, type = MX, class = IN

;; ANSWER SECTION:
nobody.com.             30M IN MX       10 mail.nobody.com.

;; AUTHORITY SECTION:
nobody.com.             30M IN NS       ns.nobody.com.
nobody.com.             30M IN NS       ns.upstream.provider.net.

;; ADDITIONAL SECTION:
mail.nobody.com.         30M IN A    10.2.34.4
ns.nobody.com.                30M IN A    10.2.35.2
ns.upstream.provider.net.     30M IN A    10.0.1.1
```

Based on the above query, attackers can discover the location of relevant name server(s) and mail server(s), as well as the provider of secondary DNS services. Again, this information can not be contained and is difficult to track. Understand that public servers such as your www, ftp, mail or DNS server are high-profile systems—they will become targets; it's only a matter of time.

One last "DNS trick" attackers often use is an "unauthorized zone transfer." But this trick can be stopped (and detected). DNS zone transfers are legitimately used for synchronizing slave DNS servers with the appropriate master server. However, by not securing your name server, anyone on the Internet can request a copy of your DNS tables. This will basically give attackers a full list of all the hosts for which you have defined DNS records. The problems with this are:

- DNS zones may detail hosts that shouldn't be publicly known
- Many organizations label machine names by their function or platform—this can give attackers even more information about the role and significance of that particular system

For example:

```
[user@server user]$ nslookup – ns.nobody.com
Default Server: ns.nobody.com
Address:  10.2.35.2

> ls nobody.com
www                30M IN A    10.2.34.62
mail               30M IN A    10.2.34.4
ns                 30M IN A    10.2.35.2
finance            30M IN A    10.2.34.6
winnt1             30M IN A    10.2.34.30
winnt2             30M IN A    10.2.34.31
unix1              30M IN A    10.2.34.33
unix2              30M IN A    10.2.34.34
dialin             30M IN A    10.3.9.2
devel1             30M IN A    10.2.35.70
devel2             30M IN A    10.2.35.71
acct               30M IN A    10.2.34.7
```

Based on the above unauthorized zone transfer, the attackers have learned quite a bit about your network and hosts. Examining this in detail, we can deduce that 10.2.35.x appears to be a development network. Development networks typically house servers with weaker security because those servers are often not as watched or patched.

Based on the above output, any machine labeled “finance” or “accounting” is tempting since they may contain corporate and employee information. Machine types also provide aid in the attackers’ quest. If an intruder can transfer your DNS zone file, they can often get information to which they shouldn’t have access.

Securing the DNS service or package can block zone transfers. Most UNIX-based machines run a version of Bind (see http://www.isc.org/products/BIND/) that can be secured using the “allow-transfer” option in the named.conf file:

```
options {
        directory "/var/named";
        allow-transfer { 10.0.0.16; };
        // use "allow-transfer" to specify hosts authorized for zone
        transfer  - i.e.
        // secondary/slave nameservers - in this case, 10.0.0.16.
                  };
```

Windows NT users should check Microsoft KB article Q164488 for updated information on securing Microsoft’s implementation of DNS. Logging of DNS zone transfer requests should be enabled. For example:

```
Sep 10 16:47:28 ns named[301]: zone transfer (AXFR) of "nobody.com"
(IN) to [10.0.100.8].2700
Sep 11 02:24:31 ns named[170]: unapproved AXFR from [10.0.100.8].35224
for "nobody.com" (acl)
```

Here we demonstrate a successful (the transfer was granted) and unsuccessful (the transfer was blocked) zone transfer as they would appear in a UNIX hosts' syslog. If you see failed transfer requests in your DNS logs, they're either mis-configurations or attempts to gather more information about your network.

Port scanning

The next step of reconnaissance is usually port scanning. Attackers will use any one of a wide set of publicly available tools (strobe, nmap, etc) to determine what services and ports are available on the target systems. Based on this information, attack plans can be formulated. An example of a port scan with nmap[0]:

```
[user@server user]$ nmap www.nobody.com -O

Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on  (www.nobody.com):
Port    State       Protocol  Service
21      open        tcp       ftp
22      open        tcp       ssh
23      open        tcp       telnet
25      open        tcp       smtp
80      open        tcp       http

TCP Sequence Prediction: Class=truly random
                         Difficulty=9999999 (Good luck!)
Remote operating system guess: Linux 2.1.122 - 2.1.132; 2.2.0-pre1 -
2.2.2
Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

Now the attackers know what services that system is running (ftp, ssh, telnet, mail, and web), and the OS has been narrowed down (Linux 2.1.x-2.2.x). Another example:

```
[user@server user]$ nmap winnt1.nobody.com -O

Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on beastmaster.lab1.neohapsis.com (10.9.100.9):
Port        State       Protocol    Service
21    open  tcp         ftp
80          open        tcp         http
135         open        tcp         loc-srv
139         open        tcp         netbios-ssn

TCP Sequence Prediction: Class=trivial time dependency
                         Difficulty=9 (Trivial joke)
Remote operating system guess: Windows NT4 / Win95 / Win98
Nmap run completed -- 1 IP address (1 host up) scanned in 2 seconds
```

Here it's revealed that this is probably a Windows NT machine running ftp and http services. Attackers can also use this to determine what systems are online and available:

```
[user@server user]$ nmap –sP 10.3.9.0/24

Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Host (10.3.9.1) appears to be up.
Host (10.3.9.2) appears to be up.
Host (10.3.9.6) appears to be up.
Host (10.3.9.45) appears to be up.
Host (10.3.9.46) appears to be up.
Host (10.3.9.68) appears to be up.
Host (10.3.9.125) appears to be up.
Nmap run completed -- 256 IP addresses (7 hosts up) scanned in 5 seconds
```

Detecting port scanning is tricky at best. Unfortunately, there are no commercial applications on NT that detect port scans. The Unix platforms, however, have a few options. We recommend Abacus Portsentry[1], a port scan detection package that is configurable, thorough and runs on many flavors of Unix. Portsentry is free at http://www.psionic.com/abacus/portsentry/.

Banner grabbing

Next attackers will try to determine what versions of services the systems are running. For instance, checking the software version of the webserver:

```
[user@server user]$ telnet www.nobody.com 80
Trying 10.2.34.62...
Connected to www.nobody.com.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Fri, 10 Sep 1999 22:11:09 GMT
Server: Apache/1.3.6 (Unix)  (Red Hat/Linux)
Last-Modified: Wed, 07 Apr 1999 21:17:54 GMT
ETag: "36b6c-799-370bcb82"
Accept-Ranges: bytes
Content-Length: 1945
Connection: close
Content-Type: text/html

Connection closed by foreign host.
[user@server user]$
```

Here the attackers have learned even more -- specifically that the webserver is running RedHat Linux distribution of Apache 1.3.6. Combining this with the portscan OS detection guess (Linux 2.1.x-2.2.x), the attackers can deduce the specific OS distribution (RedHat Linux 6.0, which comes with Linux kernel 2.2.5 and Apache 1.3.6). Now your attackers can research and find information detailing vulnerabilities that specifically apply to that OS version.

This technique is usually referred to as banner checking or gathering, since attackers are just looking at the software banners provided to them to determine the version. Some software lets you customize the banner so it does not contain any version information—a good practice to consider implementing. Also, these banner checking connections are typically logged and that's good for the

administrator. Unfortunately, for web services they don't appear any different from a normal web request. For services that require authentication, however, they create a noticeable log anomaly. A good example is FTP. Since the service expects a login, the attacker has two options: close the connection without giving any login information, or attempt to give login information (which is most likely fake/wrong). Legitimate users of the service should actually login every time, so having a connection that closes without a login may mean someone has connected just to see the banner. Observe the following UNIX ftp logs:

```
Sep 10 17:21:38 wicca in.ftpd[2558]: connect from 127.0.0.1
Sep 10 22:21:41 wicca in.ftpd[2558]: ANONYMOUS FTP LOGIN FROM localhost
[127.0.0.1], user@host
Sep 10 22:21:43 wicca in.ftpd[2558]: FTP session closed

Sep 10 17:20:36 wicca in.ftpd[2553]: connect from 127.0.0.1
Sep 10 17:20:38 wicca in.ftpd[2553]: FTP session closed
```

In the first set, authentication was provided with a valid login. We can consider this legitimate usage. The second set, however, closes the connection without a login. This is suspicious activity.

At this point, attackers will launch into service/system specific attacks. One common attack on web servers would scan for vulnerable CGI programs. An example whisker[2] CGI scan:

```
[user@server user]$ ./whisker.pl -s scan.db -h www.nobody.com
-- whisker / v1.0.2 / rain.forest.puppy / ADM / wiretrip --

= - = - = - = - = - =
= Host: www.nobody.com
= Server: Apache/1.3.6 (Unix)  (Red Hat/Linux)

+ 200 OK: GET /robots.txt
+ 200 OK: HEAD /cgi-bin/test-cgi
+ 200 OK: HEAD /cgi-bin/finger
+ 200 OK: HEAD /cgi-bin/info2www
+ 200 OK: HEAD /wwwboard/wwwboard.pl
```

Here the attackers have a list of vulnerable CGIs found on the server. Listed below are the associated log entries for the scan.

```
127.0.0.1 - - [10/Sep/1999:17:36:39 -0500] "HEAD /cgi-bin/test-cgi
HTTP/1.0" 200 0
127.0.0.1 - - [10/Sep/1999:17:36:39 -0500] "HEAD /cgi-bin/test-cgi.tcl
HTTP/1.0" 404 0
127.0.0.1 - - [10/Sep/1999:17:36:39 -0500] "HEAD /cgi-bin/perl HTTP/1.0"
404 0
127.0.0.1 - - [10/Sep/1999:17:36:39 -0500] "HEAD /cgi-bin/sh HTTP/1.0"
404 0
127.0.0.1 - - [10/Sep/1999:17:36:39 -0500] "HEAD /cgi-bin/finger
HTTP/1.0" 200 0
127.0.0.1 - - [10/Sep/1999:17:36:39 -0500] "HEAD /cgi-bin/glimpse
HTTP/1.0" 404 0
```

```
127.0.0.1 - - [10/Sep/1999:17:36:39 -0500] "HEAD /cgi-bin/info2www
HTTP/1.0" 200 0
```

This is obviously detectable by looking for a barrage of requests (the exact number depends on the CGI scanner software ran) to /cgi-bin, /scripts or similar.

A popular attack on ColdFusion servers abuses the sample templates that come by default with the server. These are typically contained in the /cfdocs directory. You can detect attacks using these templates by viewing requests to the templates in your web-request logs. A sample web-log entry may look like:

```
10.9.100.4 - - [10/Sep/1999:17:36:40 –0500]  “GET
/cfdocs/expeval/openfile.cfm HTTP/1.0” 200 0
10.9.100.4 - - [10/Sep/1999:17:36:40 –0500]  “GET
/cfide/administrator/login.cfm HTTP/1.0” 200 0
```

The first request shows a user going to the Expression Evaluator pages. It is widely known that these pages contain vulnerabilities making them a popular target. The 200 return-code means the pages exist and were returned to the attacker. The second request is an attempt to log into the administrator web pages for the ColdFusion server. You should keep track of all requests to documents in /cfdocs/ or /cfide/ directories. No one but the proper administrators and users should use these—any suspicious requests within these directories should be considered a probe or possibly even an attack (if they succeed in using a vulnerable template). You should review the documents in the Allaire Security Zone (http://www.allaire.com/security) to familiarize yourself with the different vulnerable scripts and methods for fixing them.

You can use the native ColdFusion logs to detect misuse as well. ColdFusion’s application.log will warn you of potential problems and abuses of your script logic. Unfortunately, it does not always tell you which template was the cause of the problem (for example, when an unknown template is requested). ColdFusion’s webserver.log, however, will usually give you more information (especially on requested templates). Example entries in webserver.log may look like this:

```
"Error","TID=288","10/27/99","16:52:03","HTTP/1.0 404 Object Not Found.
The template specified, C:\Inetpub\wwwroot\nyevcxsxxhhutrefwgrf.cfm,
does not exist on the specified server."

"Error","TID=288","10/27/99","16:54:01","HTTP/1.0 404 Object Not Found.
The template specified, C:\Inetpub\wwwroot\xpmnmjfaxxmymikrdhyf.cfm,
does not exist on the specified server."
```

Here we see that the attackers tried to request non-existent templates. The request for a template of twenty random characters (as seen above) is a signature of someone using the ‘whisker’ CGI scanner on your server. The tool tries requesting a random template name (in this case, a twenty character random name) in an attempt to gauge server response. Consider entries similar to the one above a pre-cursor to an attack.

Listening services

It is possible to generate a list of listening services on Unix and Windows NT. This is useful in detecting if an attacker has loaded a service allowing remote interactions with your server. For example:

```
C:\> netstat –a

      Active connections

Proto Local Address          Foreign Address        State
TCP   10.9.100.8:22          10.9.100.21:1125       ESTABLISHED
TCP   0.0.0.0:3001           0.0.0.0:*              LISTENING
TCP   0.0.0.0:80             0.0.0.0:*              LISTENING
TCP   0.0.0.0:25             0.0.0.0:*              LISTENING
TCP   0.0.0.0:22             0.0.0.0:*              LISTENING
TCP   10.9.100.8:53          0.0.0.0:*              LISTENING
TCP   127.0.0.1:53           0.0.0.0:*              LISTENING
TCP   0.0.0.0:23             0.0.0.0:*              LISTENING
TCP   0.0.0.0:21             0.0.0.0:*              LISTENING
```

Here we see that the server is listening on multiple ports: 3001, 80, 25, 22, 53, 23, 21, etc. There is also an active connection between 10.9.100.8 and 10.9.100.21. Always be aware of the services you're running, so you can spot suspicious processes using unknown ports that are in the 'LISTEN' state. Note that 'netstat –a' is the same command for both Windows NT and most flavors of Unix.

Unix-Specific issues

Modified logs could signal an intrusion. If your log files ever skip a day, an hour or even a few minutes when you know you should have had entries, be suspicious. Sloppy attackers will remove large blocks of log information, not taking the time to edit out their single entries. Neophytes and panicked attackers may delete the entire log files. Experienced attackers are the most difficult to spot. They take the time to only remove, or even edit, entries related to them. While this is nearly impossible to spot, there is a caveat: While attackers are editing the log(s), there is a time gap—from when they start editing to the point of saving/overwriting the original logs with the modified versions. During this break, any new log entries will be lost, thereby creating a gap of missing entries. Some attackers use automated tools to minimize the time gap, but luckily, the gap is nearly impossible to prevent so it truly is a good indication for which to watch. A real example of a modified log (domains changed to protect identities):

```
Jun  1 02:45:12 Wicca named[170]: unapproved AXFR from [204.x.x.x].61773
for "Domain.Com" (acl)
```

```
Jun  1 07:45:48 Wicca ftpd[11756]: ANONYMOUS FTP LOGIN FROM
user.nets.net [38.x.x.x], abc@abc.com
Jun  1 02:55:12 Wicca named[170]: unapproved AXFR from [204.x.x.x].61851
for "Domain.Com" (acl)
Jun  1 07:55:50 Wicca ftpd[11756]: FTP session closed
Jun  1 03:01:18 Wicca named[170]: Cleaned cache of 160 RRs
```

Noticed the misplaced times. This particular host suffered a breach via a wu-ftpd bug, hence the anonymous ftp login. Fortunately, in our example, the attackers missed parts of the entries and we have an IP address. However, the attackers may have modified their logins and substituted another IP address to mislead anyone attempting to read the logs.

Another sign of a security breach is when unknown files are left on the system. These files can be CGI scripts or other information files found in web roots (if the attackers used the web server as a point of entry or a means of downloading information from the server), leftover source code and exploits, and copies of logs and system information in non-system directories.

Unknown or unaccountable system-configuration file modifications should also be suspect. These types of modifications can include:

- Cron/at run files. Cron and at run scripts and programs at particular intervals. An attacker can modify one of the programs/scripts to create a backdoor at a certain interval, etc. Since many administrators schedule tasks and then 'forget' about them, it is common to overlook any modifications in this respect. You should verify all scripts and programs that are used or called during an unattended scheduled task. Also verify the write permissions of all associated scripts and programs. If one element is world-writable, then the attacker can 'overwrite' the script/program with a trojan or malicious script at any time, which is run with root privileges at a scheduled time of execution.

- Inetd.conf. Attackers may add extra services to your inetd service that are available remotely—including a system shell bound to a port requiring no authentication and runs as a root. Another method is to enable other trojaned services (which may contain backdoors that require no authentication) or known-to-be-exploitable services that can be used to later regain entry. You should verify the integrity of all programs called from inetd(.conf) and make sure none are (over)writable by users. An advanced variant of this type of attack modifies the configuration file (inetd.conf), restarts the service (for the changes to take effect), and then restores the original configuration file. In this scenario it is virtually impossible to detect a change, and the service will continue using the compromised configuration until reset/reloaded, so for safety's sake, periodically reset the service (with a UNIX "kill –1" command combination).

- /etc/passwd. A common element to an attack is the addition of a new user account, so it's important to have proper accounting on valid accounts.

Unfortunately, this can be tough for ISPs or companies with large user-bases. Check for unauthorized account additions, accounts that have no password (may be in /etc/shadow), and accounts what have UID 0 other than root.

- .rhosts. Check for any additions to or creations of .rhosts files. These allow unauthenticated remote access to your server. It is especially important to check for a '+' in any .rhosts file, since this means **anyone** can log in with no authentication. Other files you should check for the addition of a '+' or an unauthorized entry are /etc/hosts.equiv and /etc/hosts.lpd. None of these files should be writable by anyone but the owner.

- PAM (pluggable authentication modules) files control the process of authentication for each service. Examine these files (often found in /etc/pam.d/) to see if any changes have been made. It is possible for attackers to add 'rules' so that authentication information for certain users is not required, or to use foreign servers for authentication instead.

Another element of an attack is the modification of system binaries, otherwise known as installing 'backdoors'. Many UNIX-based "rootkits" exist for this purpose. These packages contain replacements for common UNIX commands, such as ls, cp, rm, find, passwd and chmod that are designed to contain backdoors or hide other elements of the attack (hidden files, changed binary signatures, prevent removal of trojaned utilities or files, e-mail passwords or changed configuration info to the attacker, etc). This type of attack is virtually impossible to detect once it happens, unless you have taken appropriate precautions. One such precaution is the creation of a binary signature database using a program such as Tripwire[3]. Programs like Tripwire create a digital "signature" of every binary on the system and keep a database of the signatures. If your server is compromised, you can create a signature database of the current binaries and compare it to the original signature database. This will alert you to any changes. Note that faking timestamps, file sizes, etc is trivial; the only way to be positive of binary integrity is to compare it to a known signature produced by a non-compromised program that uses a strong hash algorithm (MD5 is common). For security reasons, it is best to keep the program or original signature database on a different server, since it is possible that the attacker could modify them—which compromises the whole procedure.

Another popular attack method leaves shells with the set-user-id (suid) bit set to root so that the shell runs as the root user, giving attackers instant access to root status. Here's an example listing of a suid/sgid (set-group-id) file:

```
-rwsr-sr-x   1 root     root       299364 Apr 19 15:38 sendmail
```

Notice the 's' in the permissions, the first is the suid bit, the second is the sgid bit.

Scan your system for suid and sgid files by running the following commands as root:

```
find / -perm –4000 –print          (looks for suid files)
find / -perm –2000 –print          (looks for sgid files)
```

Like binary signatures, make a list of suid/sgid programs when you first install your server, and then frequently rerun the command and compare the list. Any additions should be considered with extreme caution.

You can see if attackers have installed a network watcher (a.k.a. sniffer) on your machine by checking the Ethernet status. Notice the PROMISC below—it indicates that the network adapter is in promiscuous mode, allowing it to read all the traffic on the network. This status is from a Linux server:

```
[root@server /]# /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:A0:C9:95:65:7D
          inet addr:10.9.100.8  Bcast:10.9.100.255  Mask:255.255.255.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:51627 errors:0 dropped:0 overruns:0 frame:0
          TX packets:43198 errors:0 dropped:0 overruns:0 carrier:0
          collisions:5876 txqueuelen:100
          Interrupt:11 Base address:0xfce0
```

Other versions of Unix report in almost an identical fashion. If attackers can run a network sniffer, they can 'grab' information as it goes by, including login names and passwords. The damage can be reduced in a switched environment; however, attackers will still be able to watch all connections coming from, or going to, the compromised server.

Windows-NT-specific issues

As with Unix, modified logs could signal an intrusion. On NT there are multiple log locations and types. There are webserver-specific logs (which are in different places for IIS, Website and Netscape), FTP-server logs (again, different places depending on the server software) and the NT-system log which is accessible via the Event Viewer -- this is found in %systemroot%\system32\config. If your log files ever skip a day, an hour or even a few minutes when you know you should have had entries, be suspicious. Sloppy attackers will remove large blocks of log information, not taking the time to edit out their single entries. Neophytes and panicked attackers may delete entire log files. Experienced attackers are the most difficult to spot. They take the time to only remove, or even edit, entries related to them. While this is nearly impossible to spot, there is a caveat: While attackers are editing the log(s), there is a time gap—from when they start editing to the point of saving/overwriting the original logs with the modified versions. During this break any new log entries will be lost, thereby creating a gap of missing entries. Some attackers use automated tools to minimize the time gap, but luckily, the gap is nearly impossible to prevent so it truly is a good indication

for which to watch. The NT eventlogs are unique—they are a binary format requiring special applications to edit. This means attackers are limited to:

- Uploading a tool to modify the event logs. This requires access to upload files in some form, plus access to run the tool. Finding unknown event log tools may be a sign the logs were modified.

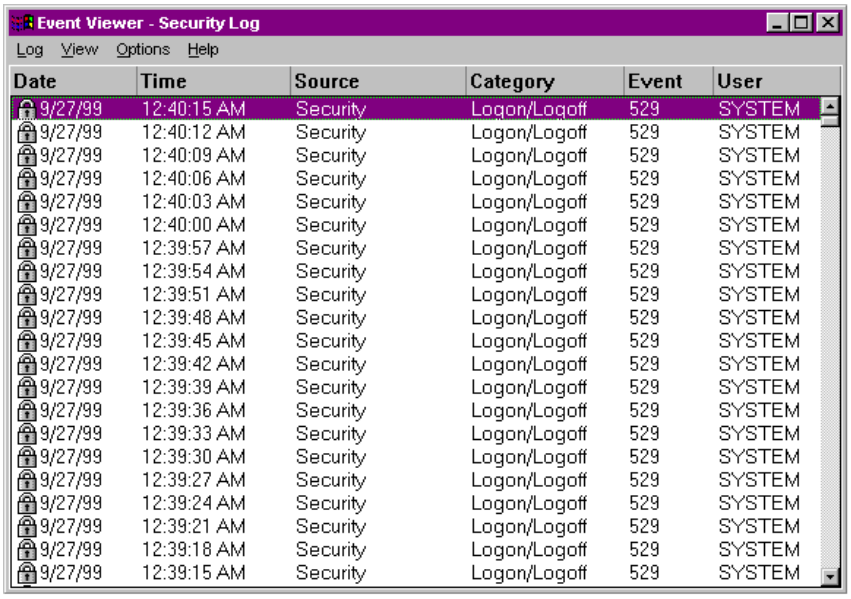
- Downloading the event logs, modifying them 'locally' and then re-uploading them. Again, this requires the ability to upload files, but does not require the ability to run programs. However, this can be time consuming and create a noticeable gap in time entries, as described above.

- Add bogus entries to the log. Uploading and running a tool that fills the logs with various fake entries to confuse or obfuscate the original entries can do this. Attackers may attempt actions that produce errors, such as failed logons, etc. just to fill the logs as well. Usually the attempt is to overflow the event log, either providing so much data that the admin will be to lazy to process all of it, or taking advantage of the 'overwrite oldest entry' log configuration.

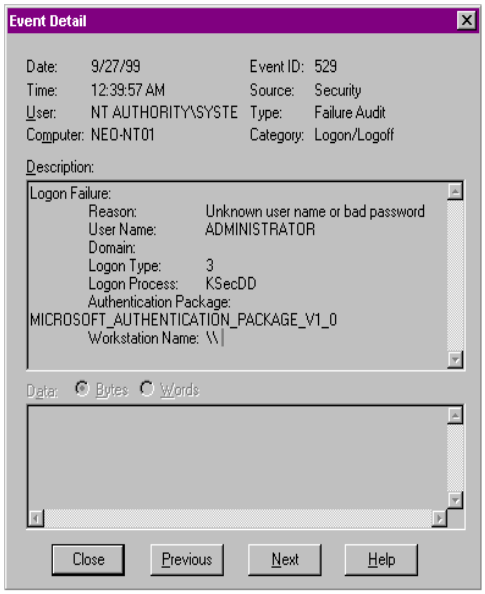
- Deleting the event logs altogether. While this is the easiest to execute, it is the most obvious to the administrator. If your event logs are ever reset, investigate how it happened.

Always follow-up on all event log messages, especially ones under 'Security Events'.

Auditing successful and failed logons is a good practice. This will detect when someone is trying to find valid password combinations with which to log on. A log file with repeated failed attempts might indicate someone is running a 'brute force' login program against your system, such as ADMsmb[4], NAT[5], etc. If this occurs, your log may look like the following:



Notice all the repeated events. A close up of event 529:



Here we see that event 529 is 'Unknown user name or bad password.' Since this occurred with such frequency, assume someone is trying to brute force access to the system.

Verify the ACLs on the event log files to make sure they weren't tampered with or modified. Appevent.evt, sysevent.evt, and secevent.evt should have Full Control for Administrators and System only. There should be no access for any other user type.

Another sign of a security breach is when unknown files are left on the system. These files may be CGI scripts and other information files found in web roots (if

the attacker used the web server as a point of entry or a means of downloading information from the server), leftover exploit applications, trojans, and copies of logs or system information in non-system directories. Especially suspicious are unknown ISAPI .DLLs, .ASPs, and .CFMs found in the web directory. Dealing with this is tough if you work for a web-hosting provider.

Always check for system binary modifications, otherwise known as installing 'backdoors' or 'trojans.' Backdoors and trojans for NT have only recently become popular. The most popular trojans are Back Orifice 2000[6] and Netbus[7]. Back Orifice 2000 has been specifically designed with NT in mind and includes mechanisms to hide itself very well. Periodically scan your system for any trojan applications. There are many free trojan detectors available online for download and included with most virus-detection packages[8].

Unfortunately for the administrator, occurrences of NT trojans and backdoors are on the rise. An organized effort at http://www.rootkit.com/ has risen to make a full-fledged NT rootkit—a package of trojaned system applications that are undetectable. This type of attack is virtually impossible to detect once it happens, unless you have taken appropriate precautions. One such precaution is the creation of a binary signature database using a program such as Tripwire[9]. Programs like Tripwire create a digital "signature" of every binary on the system and keep a database of the signatures. If your server is compromised, you can create a signature database of the current binaries and compare it to the original signature database. This will alert you to any changes. Note that faking timestamps, file sizes, etc is trivial; the only way to be positive of binary integrity is to compare it to a known signature produced by a non-compromised program that uses a strong hash algorithm (MD5 is common). For security reasons, it is best to keep the program or original signature database on a different server, since it is possible that the attacker could modify them—which compromises the whole procedure.

An attacker can also change the ACLs on files located on the system. Tripwire for NT will notify you of any changes in permissions. Make sure all executable applications, especially any found beneath the %systemroot% directory, are **not** writable by anyone other than the administrator and system. Strict permissions should be enforced on the %systemroot%\repair directory, because this contains backups of your password database. You should also monitor the file-creation dates of files in this directory, particularly sam._ (a backup of all of your account passwords). If the file date is newer than the last time you ran rdisk.exe, then someone may have purposely backed-up your password database to gain access to it (the SAM is not normally accessible, so attackers must back it up and use this copy).

NT, unlike Unix, has another database prone to attacks: the registry. It's massive size and complexity make changes difficult to detect. Luckily, Tripwire

for NT, as well as other applications available on the Internet will monitor and log changes to the registry. A particular key to watch is:

\HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurePipeServers
\winreg\AllowedPaths

(key is wrapped for clarity). This key defines what parts of the registry are remotely accessible. By default this is:

System\CurrentControlSet\Control\ProductOptions
System\CurrentControlSet\Control\Print\Printers
System\CurrentControlSet\Services\EventLog
Software\Microsoft\Windows NT\CurrentVersion
System\CurrentControlSet\Services\Replicator

Your security configuration may call for fewer of these (Software\Microsoft\windows NT\CurrentVersion in particular), but any addition to this list should be deemed suspicious. Also, verify the ACLs on this key, and any keys contained in the allowed paths mentioned above. None of the keys should be set to allow normal users to Set Value, Delete Key or the like. Another common registry attack adds entries to the 'Run' and 'RunOnce' keys, located in various places for each user and the system. Trojans are typically launched from these keys.

Attackers may modify the user-base. A common element to an attack is the addition of a new user account, therefore it is important to have proper accounting on valid accounts. Unfortunately this can be tough for ISPs or companies with large user-bases. Check for the addition of unauthorized accounts, as well as accounts that have no password. You should also periodically double-check who is in the Administrators, Domain Admins, Server Operators or Account Operators groups, and verify they have valid passwords. Some attackers will add a new user to the Administrators group, or add an existing user and change or remove the password. They may also delete the password of a current administrator. A good way to get a list of all accounts in the Administrators (or any) group is to use the 'net' command at the prompt:

```
C:\> net group "domain admins"
Group name        Domain Admins
Comment           Designated administrators of the domain

Members

*Administrator          *NewAdmin
The command completed successfully.
```

In this case you should check the validity of NewAdmin. Also verify that the guest account stays disabled, if your security configuration has it as such.

Attackers may modify or schedule new 'at' jobs. Monitor and verify all 'at' jobs, as well as any applications for scripts they call to perform their function.

Attackers can modify one of the programs/scripts to create a backdoor at a certain interval, etc. Since many administrators schedule tasks and then 'forget' about them, it is common to overlook any modifications in this respect. Also verify that all called components have a restrictive ACL that disallows anyone from modifying them (have write/modify permission).

The recent ODBC vulnerabilities have been a popular attack method. Monitor your ODBC DSNs, watching for any new creation. Unauthorized system DSNs, especially ones that use the Microsoft Access Driver, may be signs of an attack. Make sure to delete all DSNs that you are not using.

Use the services applet to periodically double-check what services are running on your system. Some services will add startup and shutdown messages to the event log; however, not all services have this feature. You can also use the 'net start' command to get a list of running services:

```
C:\> net start
      Server
      Net Logon
      NT LM Security Support Provider
      Plug and Play
      Protected Storage
      Remote Procedure Call (RPC) Locator
      Remote Procedure Call (RPC) Service
      Workstation
      Alerter
      Eventlog
      Computer Browser

The command completed successfully.
```

(Note that you may have a different combination of services than those listed above.) As with binary signatures, make an initial list of running services for comparison. Attackers may turn on a service known for its vulnerability or even install their own service. Since services usually run under the 'System' context—meaning they have full access to do anything on your server—you would be wise to make sure nothing unauthorized runs as a service.

---

[0] http://www.insecure.org/nmap/

[1] http://www.psionic.com/abacus/portsentry/

[2] http://www.wiretrip.net/rfp/

[3] http://www.tripwiresecurity.com/

[4] ftp://ftp.freelsd.net/ADM/

[5] http://packetstorm.securify.com/NT/scanners/

[6] http://www.bo2k.com/

[7] http://www.netbus.org/

[8] http://www.nai.com/, http://www.symantec.com/, http://www.datafellows.com/

[9] http://www.tripwiresecurity.com/