Global Information Assurance Certification Paper

Effective Log Reduction and Analysis Using Linux and Open Source Tools
GIAC Security Essentials (GSEC)
Practical Assignment, Version 1.4b, Option 1
August 19, 2003
Tim Kramer

Abstract:

This paper presents a "HOWTO", discussing a few methods for rapid log reduction and effective analysis. The tools discussed in this paper include grep, vi, Perl, nmap, amap, nbtscan, and smbclient.

The Problem:

On a daily basis, a network administrator can upward of 100MB of router logs to review. Without a consistent and effective method of reducing what has to be manually reviewed, the administrator would be wasting time on a daily basis. In the long run, log file review would fall by the wayside as other less time consuming tasks often appear to have a higher priority.

The Logs:

For this exercise, we will assume you are working with logs from a Cisco router ACL which logs traffic inbound from the Internet. Your router logs are most likely generated via a syslog daemon on a server which allows your other equipment to send log entries to it. The format for syslog entries [1] generally follows:

<date/timestamp> <reporting machine> <message>

For log entries from a Cisco router, the form followed is:

<local date/timestamp> <reporting machine> <process id> <remote date/timestamp> <Cisco Message Type> <message>

where "*<message>*" usually equates to:

<reporting access list> <action> <protocol> <source> <destination> <packet count>

An example:

Jul 4 02:00:10 [172.2.255.5.220.228] 21991638: Jul 4 02:05:42: %SEC-6-

IPACCESSLOGP: list FWI permitted tcp 218.5.91.217(4305) -> 192.168.95.186(110), 1 packet

Note: "SEC-6-IPACCESSLOGP" is the Syslog Message Type[2] most often associated with reporting TCP/UDP traffic. ICMP traffic is associated with the "SEC-6-IPACCESSLOGDP" Message Type.

The general rule of thumb is to log as much as possible without affecting network/system performance. Depending on your acceptable usage policy, you may want to log everything that isn't expressly allowed. For corporate, government, or military organizations, this might be "log everything except legitimate mail and web traffic". The idea is to explicitly allow specific traffic, implicitly deny (and log) everything else. If you cannot "deny by default", at least use "log".

Needless to say, this level of logging will generate megabytes of log entries.

The Tools:

grep [3] - A standard utility in most versions of Linux and Unix. Considered part of the GNU project. Allows user to search for patterns in files. Command line switches include inverse filtering (ignore matches, report everything else) and ignoring text case. The most common switch used for the purpose of this exercise is the "-v" switch. It basically "inverts" the filtering process, causing grep to output any lines that do not match a given expression.

telnet [11] - A standard utility in most versions of Linux, Unix, and Microsoft Windows, telnet allows the user to connect to remote TCP services on various ports.

nmap [4] - Self described as a "network exploration tool and security scanner". A very powerful port scanner which can also recognize some RPC services and make educated guesses about the operating system running on the target machine(s). Command-line switches allow for enabling specific types of scans, outputs, and other controls.

nbtscan [5] - A NetBIOS nameserver scanner which is capable of running on Linux. Allows you to view miscellaneous NetBIOS data available from a target system. For the purpose of this exercise, it is used to determine the SMB host and domain names of the target system.

amap [6] - An application/services scanner which attempts to identify the service listening on a port by making multiple connections to it, feeding it specific "triggers" and comparing the responses with known "fingerprints". Although a full description of this tool is beyond the scope of this paper, SANS does have a

paper describing it [7].

smbclient [8] - Part of the Samba tool suite, SMBClient is able to "talk" to a SMB/CIFS server. For the purposes of this exercise, we will use in in conjunction with NBTScan to test for available shares.

nmblookup [9] - Also part of the Samba tool suite. From the man page: "NMBLookup is used to query NetBIOS names and map them to IP addresses in a network using NetBIOS over TCP/IP queries." For our purposes, we will use this to test for NAT'd IP addresses and dual NIC cards.

perl [10] - A well-known and very powerful scripting language. For the purposes of this paper, it will be used to glue together a few of the above tools to automate log file reduction.

Log File Reduction (Cleaning out the chaff):

Believe or not, users are usually creatures of habit. On a day-to-day basis, they will usually perform the same actions via their workstations. Examples include: reading e-mail, surfing the web, playing online games, listening to streaming audio or video. These generate consistent log entries (source/destination addresses/ports). Once you identify the traffic as legitimate, you shouldn't have to review it again. Following is a method for removing "recognized" traffic.

I'll describe how to perform manual log reduction for now. Later in this document I'll describe how to automate the process.

First, you'll want to keep an original copy of the traffic. Well call this "main". Then, you want a working copy, which you will make smaller as you remove recognized traffic. Let's call this one "working". For our exercise, we'll only discuss working with TCP. (It usually will make up the majority of your log files.) You'll actually do this twice, once for TCP and once for UDP. You'll also want to take a look at ICMP traffic but as it's generated by errors or by ping sweeps, there's not much you can filter from it as legitimate traffic.

Generate your working file from the main file via:

```
grep tcp main > working
```

If the example Cisco log (page 1) is a legitimate connection to a POP3 server, we can remove it from the "working" copy via:

```
grep -v " -> 192.168.95.186(110)" working > temp
mv temp working
```

The "v" switch for grep means to filter (remove) the following string from the file read. The ">" redirects the output from STDOUT to a file called "temp". You then rename "temp" back to "working". The idea is to do this for each legitimate service that you run (SMTP, HTTP, IMAP, LDAP, etc.). This process will probably provide the greatest reduction in the size of the working copy.

After this, you want to look for known "bad" traffic and note (or report) the source IP's. "Bad" traffic includes anything that should be coming in through your premise routers: NetBIOS, DNS zone transfers, finger requests, SQL, etc. The use of grep differs slightly as you want to look at the log entries before you remove them from the "working" copy. The idea is that you note the source IP so that you can add it to your premise router's ACL to block any further traffic from that site.

A manual example of this:

```
grep "(13[579])," working|more
grep -v "tcp 172.23.23.48 " working > temp
mv temp working
```

The first line searches "working" for traffic to destination ports 135, 137 and 139. The second and third lines are an example of removing a specific source IP address from the "working" copy.

What remains in your working copy will be a mix of unknown "bad" traffic, unknown traffic generated by your users, and some background noise. Background noise is traffic that you'll probably never recognize. It can be backsplash from DoS attacks with spoofed source IP's which just happen to be your IP's, random manual errors generated by users testing services and fat fingering the destination IP's, etc. The idea is to minimize the "working" copy as much as possible before you "write off" the remaining file as background noise.

Automating the process:

For the purposes of this exercise, we'll work with Perl. Substitute your favorite scripting language if you cannot work with Perl.

First, you'll want to automate the "known good" removal process. You'll want some form of list of valid IP addresses and port numbers. I maintain a file for each port with IP addresses listed in it. As an example, I use "legit.25" to list my mail servers in the following format:

```
# mailservers
# hisdomain.com
192.168.95.30
```

```
# herdomain.com
192.168.95.40
```

Similar files will hold IP addresses for web servers, file servers, or anything that is visible to the outside world.

So to strip legitimate traffic from the "working" copy, we want to get a list of the names of the files that hold the legitimate IP's. Then, we'll want to open each of those files, determine the port number from the filename, load the IP's from the file and grep the legitimate traffic from the working copy:

```
#!/usr/bin/perl

# get the names of the files containing the legitimate traffic
@legits=glob("legit.*");

# load each file
foreach $legit (@legits) {
        chomp $legit;

        # determine the port # the file describes
        ($junk,$port)=split(/\./,$legit,2);

        # open the file and grep the IP's from the working copy
        open(INFILE,"$legit");
        while(<INFILE>) {
                $line=$_;
                chomp $line;

                # if the line starts with a #, it's a comment!
                if($line =~ /^#/) {
                        next;
                } else {
                        $phrase=" -> $line"."($port)";
                        system("grep $phrase working > temp");
                        system("mv temp working");
                }
        }
}
```

Analyzing What's Left:

After this is done, you'll have to manually review the remaining "working" file. The objective is to look for probes or attacks from outsiders and/or internal abuse. Once you identify IP addresses and port as suspicious, you'll have to

probe them to determine what they are.

You'll need a quick way to trim the file down to just IP addresses and ports. This can be done by opening working in "vi" and issue the following command in command-mode (example for the "tcp" file):

:s/.* tcp // (Note the spaces before and after "tcp".)

What's left will be a source IP, a source port, a destination IP, a destination port, and a packet count. You can then sort this file via:

```
sort tcp > temp
mv temp tcp
```

and then view it via "vi" or "more". It will be readily apparent if an external IP address is searching your network for specific services. An example output might look like:

```
172.38.59.7(3458) -> 192.168.95.1(110), 1 packet
172.38.59.7(3458) -> 192.168.95.2(110), 1 packet
172.38.59.7(3458) -> 192.168.95.3(110), 1 packet
172.38.59.7(3458) -> 192.168.95.4(110), 1 packet
172.38.59.7(3458) -> 192.168.95.5(110), 1 packet
172.38.59.7(3458) -> 192.168.95.6(110), 1 packet
172.38.59.7(3458) -> 192.168.95.7(110), 1 packet
172.38.59.7(3458) -> 192.168.95.8(110), 1 packet
172.38.59.7(3458) -> 192.168.95.9(110), 1 packet
172.38.59.7(3458) -> 192.168.95.10(110), 1 packet
172.38.59.7(3458) -> 192.168.95.11(110), 1 packet
172.38.59.7(3458) -> 192.168.95.12(110), 1 packet
```

You can then note (or report) the offending IP address, and remove it from your "working" copy. Do this for each of the remaining obviously "bad" source IP's and your working copy will be much smaller.

Some of your "tcp" file may look like:

```
172.49.58.9(1029) -> 192.168.95.110(1214), 5 packets
172.49.58.9(1030) -> 192.168.95.110(1214), 3 packets
172.49.58.9(1032) -> 192.168.95.110(1214), 16 packets
172.49.58.9(1035) -> 192.168.95.110(1214), 100 packets
172.49.58.9(1037) -> 192.168.95.110(1214), 5 packets
172.49.58.9(1038) -> 192.168.95.110(1214), 7 packets
```

Obviously, an external IP address is connecting to a service listening on port 1214 on one of your internal IP's and some sort of information transfer is taking

place. You can either visit the machine and run "fport" on it to determine what software is listening on that port or you can attempt to test the port directly.

Probing the System:

One of the first tools to use in identifying the service is Telnet (if it's a TCP service). I used an obvious port in the example above as it is one that can be tested by connecting to it and prodding it into responding with enough information to identify what it is:

```
telnet 192.168.95.110 1214
Trying 192.168.95.110...
Connected to 192.168.95.110.
Escape character is '^]'.
```

At this point, typing "GET /" will get a response:

```
GET /
HTTP/1.0 501 Not Implemented
X-Kazaa-Username: michaelellison73
X-Kazaa-Network: KaZaA
X-Kazaa-IP: 192.168.95.110:2776
X-Kazaa-SupernodeIP: 172.75.235.30:1120
```

Again, the resulting output is obvious. While this doesn't always work, it does allow you to identify more services running on your network and determine if they're allowed or should be removed. Examples of services that return text banners or can be prompted into responding with text information include: HTTP, SMTP, VNC, the Jeem worm, MySQL, and Bit Torrent. There are many more services that react this way.

If you cannot gather enough information from the service to determine what it is by telnet'ing to it, try amap. As stated above, amap will try to determine what the service is by making multiple connections to the service, sending various well-known "triggers" to the service, and examining any responses from the service. It is a very nice tool to have when identifying Microsoft RPC services.

As you gain experience with investigating services listening on your network, you will recognize more and more port numbers. Google is a very useful tool in determining what service is listening on a port. It make take a few extra minutes to find the proper description but it's well worth the effort if you keep notes on what you've looked at previously. A sample search of Google might look like "tcp port 1214".

Speeding Things Up:

To automate catching insider abuse, you want to do something similar to the method used for removing “known good“, though you only need a list of ports to watch for:

```
# Kazaa
1214
# NetBIOS
135
137
139
# Bit Torrent
6681
6882
6883
6884
6885
6886
```

To pull the insider abuse from the logs, you may want to do something like:

```
#!/usr/bin/perl

open(INFILE,"badportlist");
@ports=<INFILE>;
close INFILE;

foreach $port (@ports) {
        chomp $port;

        if($port =~ /^#/) {
                next;
        } else {
                $phrase="($port), ";
                system("grep \"$phrase\" working > badport.$port");
                system("grep -v \"$phrase\" working > temp");
                system("mv temp working");
        }
}
```

This will leave you with a collection of files with the prefix “badport”. Open each of them, review their contents, and take a test the systems supplying the services.

Looking Closely at the Systems:

In any case, one you've determined that you have insider abuse (or a possible compromised system), it's usually a good idea to gather as much information as possible before going to the security manager to report the system. The main tools you'll use for this are: nmap, nbtscan, smbclient, and nmblookup.

First, you'll want a full port scan. Nmap is perfect for this as it can not only scan for open ports, it can also make educated guesses as to what operating system is running on the suspicious system. A common use of nmap would look like:

nmap -sS -sU -O -r -v -n -p 1-65535 192.168.95.110

The "-sS" switch tells nmap to perform a half-open (SYN) TCP scan. The "-sU" switch tells nmap to also perform a UDP scan. The "-O" switch tells nmap to guess at what operating system is running on the remote system. The "-r" switch tells nmap to not randomize the order of the ports scanned. The "-v" switch tells nmap to be verbose while scanning. "-n" tells nmap to skip performing reverse DNS lookups. "-p 1-65535" tells nmap to scan all available ports. Sample output of the above looks like:

```
Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Interesting ports on (192.168.95.110):
(The 131055 ports scanned but not shown below are in state: closed)
Port State Service
135/tcp open loc-srv
137/udp open netbios-ns
138/udp open netbios-dgm
139/tcp open netbios-ssn
445/tcp open microsoft-ds
445/udp open microsoft-ds
500/udp open isakmp
1025/tcp open listen
1028/tcp open unknown
1029/tcp open unknown
1433/tcp open ms-sql-s
1503/tcp open imtc-mcs/Tijit
6101/tcp open unknown
38037/udp open unknown
38292/tcp open unknown
```

Note the Tijit entry. This is not standard for NMap output but can be easily added by editing /usr/share/nmap/nmap-services.

If port UDP port 139 is listening, it's a good indication that NetBIOS is running on the suspicious machine and more data can be obtained from it via the use of

NBTScan. A common use of nbtscan would look like:

```
nbtscan -v -h 192.168.95.110
```

The "-v" switch tells nbtscan to be verbose and the "-h" switch tells nbtscan to format its output so that it's readable by a human. Output from the above might look like:

```
[root@intmgmtws1 etc]# nbtscan -v -h 192.168.95.110
Doing NBT name scan for addresses from 192.168.95.110


NetBIOS Name Table for Host 192.168.95.110:

Name             Service Type
----------------------------------------
LAPTOP           Workstation Service
MSHOME           Domain Name
LAPTOP           Messenger Service
LAPTOP           File Server Service
MSHOME           Browser Service Elections
MSHOME           Master Browser
__MSBROWSE__     Master Browser

Adapter address: 00-40-f4-32-41-72
----------------------------------------
```

Note that nbtscan has displayed the local MAC address of the suspicious machine! This works from remote scans also.

If nbtscan reports any SMB host name, it's a good idea to perform further checks with smbclient and nmblookup. smbclient will display available shares and domains visible from the suspicious machine. A common use of smbclient looks like:

```
smbclient -I 192.168.95.110 -L ADRIANS -N
```

The "-I 192.168.95.110" tells smbclient what IP address to query. "-L ADRIANS" tells smbclient what SMB list the services/shares available via the ADRIANS hostname. "-N" indicates that smbclient should not ask you for a password. Sample output from the above looks like:

```
[kramert@intmgmtws1 kramert]$ smbclient -I 192.168.95.110 -L
DESKTOP -N
added interface ip=192.168.52.9 bcast=192.168.52.255
```

nmask=255.255.255.0
Domain=[HOME] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]

Sharename	Type	Comment
IPC$	IPC	Remote IPC
print$	Disk	Printer Drivers
SharedDocs	Disk	
C	Disk	
Deskjet	Printer	hp deskjet 990c
ADMIN$	Disk	Remote Admin
C$	Disk	Default share
Send_with_eF	Printer	Send with eFax Messenger Plus

Server	Comment
DESKTOP	Desktop

Workgroup	Master
ACCT	ACCT673
EPA	DEPT31BODO
HOME	DESKTOP
PERSONNEL	LAPTOP4A
DEPT101	DEPT101SERV1
DEPT103	DEPT103SERV1
DEPT11	DEPT11SERV1
DEPT143	DEPT143PDC01
DEPT211	DEPT211SERV1
DEPT213	DEPT213SERV1
DEPT31	DEPT31OAS
DEPT32	DEPT32OAS
DEPTC12	DEPTC12SERV2
WORKGROUP	SAME_KBD

Note the shares listed. You can try mounting them using smbmount but make sure you have permission from your organization to do so. This may be construed as an invasion of privacy, even if the share doesn't have a password, if your organization doesn't have the proper policies in place and the user has an expectation of privacy.

nmblookup also provides valuable information if you aim it at the suspicious system:

```
nmblookup -U 192.168.95.110 DESKTOP
```

The "-U 192.168.95.110" tells nmblookup that you only want to query the one IP address and "ADRIANS" is the SMB hostname that you want to use in the query.

```
[root@intmgmtws1 root]# nmblookup -U 192.168.95.110 DESKTOP
querying DESKTOP on 192.168.94.110
10.14.12.192 DESKTOP<00>
```

This information becomes valuable when systems are behind a NAT'ing router or firewall. Note that in this case, two IP addressees are displayed, neither looking like the IP address used in the command line. This is an indication that one of the IP's is NAT'd externally (via another device) and that the other IP address is attached to either the same NIC or a second NIC in the suspicious system. Normal, directly connected systems will report back with the same IP address that you queried with nmblookup.

Depending on your organization's policies, you should now have enough information to either block the suspicious system(s) or to turn over to your security manager for prosecution. You should also keep notes on the information gathered as it might show up again, either in the form of another compromise of the same system or a similar compromise of a different system.

As you probe each system or note each outside attacker, remember to remove them from your working copy. Once you've removed them from known_good, known_bad, and investigated the remaining, all you should have left is a small file containing what should amount to the background noise of the internet.

References:

[1] RFC 3164 BSD Syslog Protocol
URL: http://www.ietf.org/rfc/rfc3164.txt

[2] S4T68360 Error Messages
URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios112/ebook/ess4t.pdf

[3] Grep man page
URL: http://www.nevis.columbia.edu/cgi-bin/man.sh?man=grep
Grep is normally available as part of most Unix/Linux distributions.

[4] NMap man page
URL: http://lrp2.steinkuehler.net/Packages/man/nmap.1.man.htm
NMap is available via http://www.insecure.org/nmap/

[5] NBTScan home page (download available)
URL: http://www.unixwiz.net/tools/nbtscan.html

[6] Amap home page
URL: http://www.thc.org/

Download available via: http://www.thc.org/releases.php
[7] Antonia Rana - What is AMap and how does it fingerprint applications? (SANS paper explaining the use of AMAP)
URL: http://www.sans.org/resources/idfaq/amap.php
[8] SMBClient man page
URL: http://www.ast.cam.ac.uk/~rgm/cirsi/samba/docs/man/smbclient.html
SMBClient is available as part of the Samba tool suite.
[9] NMBLookup man page
URL: http://au1.samba.org/samba/docs/man/nmblookup.1.html
NMBLookup is available as part of the Samba tool suite.
[10] Perl home page: http://www.perl.org
Download available via: ftp://ftp.cpan.org/pub/CPAN/src/perl-5.8.0.tar.gz
[11] Telnet man page:
URL: http://www.nevis.columbia.edu/cgi-bin/man.sh?man=telnet

Additional Recommended Reading:

Article "As Easy As Falling Off A Syslog" by Mark Gibbs, Network World 6/10/02
URL: http://www.nwfusion.com/columnists/2002/0610gearhead.html

Article Various at URL: http://www.counterpane.com/log-analysis.html

Article "Log Analysis Resources on the World Wide Web", Maintained by Tina Bird. URL: http://www.sometips.com/tips/goodarticles/199.htm

Article "Application Specific Log Parsing", Maintained by Tina Bird
URL: http://www.loganalysis.org/sections/parsing/application-specific/

Book Network Intrusion Detection – An Analyst's Handbook
By Steven Northcutt
ISBN 0-7357-0868-1

Book Intrusion Signatures and Analysis
By Steven Northcutt, Mark Cooper, Matt Fearnow, and Karen Frederick
ISBN 0-7357-1063-5

Book Know Your Enemy – Revealing the Security Tools, Tactics, and Motives of the Blackhat Community (Compiled by The Honeynet Project)
ISBN 0-201-74613-1