Global Information Assurance Certification Paper

Bruce Auburn
GCIA Practical v.3.3
Intrusion Detection in Depth
July 8, 2003

Table of Contents

Abstract

This paper consists of three main parts. Part one is an explanation of honeypots. Types of honeypots, legal implications and building a honeypot are discussed. Some actual data from a honeypot is shown with an explanation of the data.

Part two of the paper details three network detects. The detects were based on data from daily logs at intrusions.org. The first detect is a scan Socks proxy attempt. Socks servers are often targets for spammers and other attackers trying to hide behind a legitimate site. The second detect concerns the TCP "more fragments" and "don't fragment" flags being set at the same time. The third detect investigates malformed advertisements associated with Microsoft's Universal Plug and Play protocol.

Part three of the paper analyzes five days' worth of data logs from incidents.org. Alert files, scan files and out of specification files are examined. Some of the most dangerous and interesting anomalies are examined in depth.

Part 1 – Describe the State of Intrusion Detection

Honeypots Explained

What is a Honeypot?

A Honeypot is a device, hardware or software based, that detects and tracks intruders and attackers. Unlike a bank, the owner of a honeypot wants people to break in. A honeypot can be virtual, existing only in software, or it can be a computer, switch or router. The goal of a honeypot is to draw attackers so their methods can be studied. In this way, security weaknesses can be identified and future hacking attempts can sometimes be prevented. Honeypots can also act as sacrificial bait in a system, drawing attackers away from the production servers. Groups of honeypots can be combined to form a honeynet, or network of honeypots. This provides a more realistic environment for hackers, who can be quick to spot a single honeypot. Honeypots should be implemented only by companies who already have a fairly secure system. The honeypot will do no good if the other computers in the network can routinely be broken into.

A honeypot is a research tool. It can be deliberately misconfigured to draw attackers, or it can be set to the highest possible security to see how secure it really is. There are some disadvantages that come with honeypots. If a honeypot is compromised, a hacker might gain entrance to the rest of a network and wreak havoc. Honeypots require manpower to maintain the equipment and analyze the logs. It is difficult to build a system that can log all events without an attacker knowing he is being watched. Not all intrusions will be stopped, because new attack methods are being invented all the time. The honeypot should not be able to communicate or share files with the other machines on the network, unless it is part of a honeynet. Kernels can be made non-rewriteable to avoid damage, and special hardened versions of some operating systems are available that resist changes. Computers can be set to automatically reboot if attempts to change the kernel are made. (Spitzner)

Honeypots have their advantages, though. Since they are not performing any useful functions, all activity into the honeypot can be considered illicit. The volume of regular data seen with production servers is not present in a honeypot. Therefore, the data that is logged is smaller in volume and more informative. There is much less for data for the analyst to wade through. This makes it easier to discern new exploits that might get lost in the noise of a busy production system. Honeypots also require limited resources and can run on otherwise obsolete machines. Log data can be encrypted to thwart defacement. Honeypots can slow down attacks to gain time for the rest of the network to be

protected, especially in the case of worm attacks. One technique is to use a tcp window size of zero to put an attacker in a holding pattern. Honeypot systems can be taken offline to analyze what an attacker has done, unlike many production systems. The data gathered in a honeypot system can be used to prosecute an attacker.

Honeypot and Honeynet Examples

A group of security professionals has set up a network of computers, configured like a typical production environment, and has studied the attacks on these computers for years, amassing a large volume of useful data on hackers. The results are posted to the www.honeynet.org web site. This site has been helpful in getting the message out that attacks on computers are a real danger and are quite common. A lot of system administrators have no idea of the volume of hacking attempts. (ZDnet)

An example of a honeypot in action: A group of hackers in Pakistan broke into a network in the United States and used it to attack computers in India. For over a month, they were not aware that they had actually targeted a honeypot. Every keystroke was recorded, as were the techniques they used to attack computers and hide their tracks. (Olsen)

Another example of a honeypot is one that looks like an open proxy. This attracts email spammers, who are always looking for an open proxy so they can send millions of spam messages anonymously. A program called Jackpot simulates an open proxy to attract spammers. The email flows in, but it does not go out to its intended victims.

An example of an imaginary honeypot: A large Internet security firm with employees in many countries had their web site compromised and defaced. Instead of admitting their shortcomings, they claimed that they used their site as a honeypot and expected such an attack. This has happened more than once. It turns out that poor security is being called a honeypot by some administrators looking for an excuse.

Low-Interaction and High-Interaction Honeypots

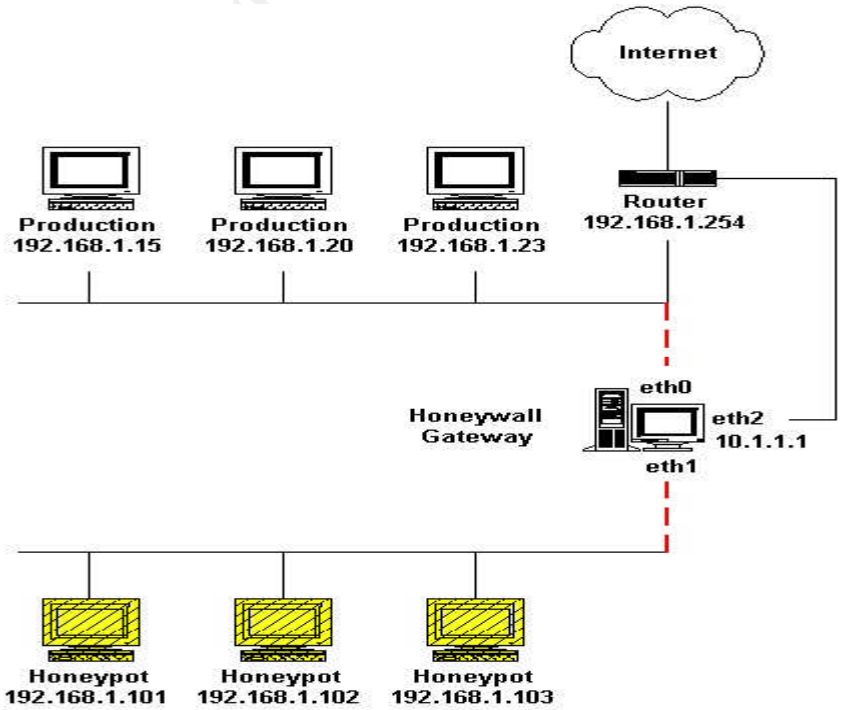
Honeypots can be characterized as low-interaction or high-interaction. A low-interaction honeypot limits the options available to the attacker. For instance, the telnet service might be enabled but be limited to login only, or it may allow a limited number of telnet commands. This type of honeypot is simple to configure and capture logs from. Low-interaction honeypots are usually software emulations. These types of honeypots are easier to set up and operate, and they run less risk because the hacker does not have access to the actual operating system. However, an experienced hacker can usually detect that he is on an emulated system. Some examples of low-interaction honeypots are Specter,

Honeyd, Deception Tool Kit and KFSensor.

Honeyd is an open source, low-interaction honeypot designed for Unix systems but ported to Windows. Honeyd logs any traffic to UPD or TCP ports. All incoming traffic to a port can be logged, including usernames and passwords. The drawback to Honeyd is that certain behavior is expected of the attacker, and if the hacker does something unanticipated (as any computer user is guaranteed to do), the emulator will probably not know how to respond. Honeyd can emulate services, complete operating systems or even a Cisco router. Even if a hacker uses a fingerprinting tool like nmap, the emulator responds as an actual system would.

A high-interaction honeypot gives the attacker a real operating system to work with. Since the system is the same as an actual production computer, the hacker is not limited in his scope of operations, nor is he able to figure out as quickly that he is being watched. Being actual systems, high-interaction honeypots are harder to set up and maintain than low-interaction ones. Examples are Symantec Decoy Server and Honeynets. (Tracking-hackers.com)

Honeynets are simply a collection of honeypots configured into a network. Kernel modules are placed within the honeynet which enable the system to capture all of the attackers' activities. A honeywall gateway controls the attackers' actions by allowing traffic into the honeynet, but limited traffic out, which is the opposite of a typical firewall. An example of a honeynet is shown below.



http://www.honeynet.org/papers/gen2/figureA.jpg

Counterattacking

Some organizations that have been attacked try to turn the tables and counterattack with some of the same techniques they have fallen prey to. This is sometimes done illegally, because in many countries it is illegal to intentionally attack computer systems. Some organizations have set their firewalls to automatically strike back at what they perceive to be an attack. One drawback to this aggressive stance is that source addresses are often spoofed, so an innocent network could find itself being attacked. In a more legal vein, a victim can use a lookup site such as www.dshield.org to find information about an attacker (if the source address is not spoofed) and send an email to the system administrator, who may disable the account. However, most sophisticated hackers can hide their actual IP address. (Landergren)

Legal Issues

As stated, Counterattacking is illegal in the United States, and it may be a bad idea because it would only promote further attacks.

Some people think that because honeypots lure attackers, the ability to prosecute the attackers is reduced. This is not true, because honeypots are passive and do not advertise themselves. It is a good idea, however, to put a standard banner page about unauthorized use on any honeypot system. In all probability, a hacker would not see this banner unless he had cracked a password, but it is good to have all of the legal angles covered, since a judge or jury might tend to focus on something that seems pertinent to a case but in reality has nothing to do with the malfeasance that has taken place.

If you operate a honeypot and an attacker compromises your system, then uses it in attacks against other systems, there is a chance you will be held liable. Legal opinion has not yet solidified on this issue. (Windowsecurity.com)

Building a Honeypot

First, decide what information you want to get out of the honeypot. If you want to test a particular vulnerability, a software honeypot like Honeyd might be the answer. If you want to see how vulnerable a particular computer or operating system is, it might be best to build a system typical of the machines within your network. You must decide what kind of information you want to log and where to store the logs. The logs must be made inaccessible because the experienced attacker will want to hide his activity. Somebody must monitor and maintain the honeypot. After an attack, if a computer is used as a honeypot, it will have to be

fixed or rebuilt. This paper will assume a Unix computer is being used as a honeypot. The computer itself must be interesting enough to attack. If it obviously being used as a honeypot, the attacker may ignore it, or do his best to destroy it without leaving you any useful information.

Collecting data is the most important part of a honeypot. The first line of defense is the firewall, and most firewalls have the ability to log data. This data should be stored and analyzed. As stated before, the firewall rules should allow traffic in, but limited traffic out. This will prevent the attacker from using your computer for other ill deeds. It is probably enough to allow only FTP, ICMP and DNS out. It may be best to use a dedicated server to store system logs, since the attacker will most likely alter or destroy the logs on the honeypot. The syslogd program, which logs system activity, can be recompiled to look at a different configuration file. The system log will then be written both locally and to the dedicated server. After an attack the remote and system logs can be compared to see if the attacker has altered the system log. A sophisticated attacker can sniff packets, detect the remote logging, and kill the syslogd process. In that case, you might want to encrypt data or use an entirely different protocol such as IPX to send the log data to the remote server.

One other way to log hacker activity is by using a sniffer at the firewall. Since all data has to come through the firewall, all activity down to keystrokes can be captured. Snort can be used for this and has the advantage of being free. Having a variety of data logs is important, because there is always a chance that the attacker will find a way to destroy your logs.

Logs from an actual honeypot

Following are some traces from a Honeypot designed by Toby Miller. Some of the data makes it quite apparent what the attacker is doing. This particular attacker exploited vulnerabilities in OpenSSH, which allow an attacker to gain root control. SSH is a program that encrypts all Internet traffic and provides tunneling and authentication. The honeypot had been running about six weeks before it was compromised. (Miller)

```
02:49:56.493164 xxx.xxx.xxx.xxx.1153 > 10.10.10.40.ssh: P [tcp sum ok]
1:29(28) ack 22 win 32120 <nop,nop,timestamp 54447663 655960541> (DF)
(ttl 45, id 6952, len 80)
0x0000    4500 0050 1b28 4000 2d06 bb3c xxxx xxxx        E..P.(@.-..<...=
0x0010    0a0a 0a28 0481 0016 6618 d792 d4ec 634d        ...(....f.....cM
0x0020    8018 7d78 7e82 0000 0101 080a 033e ce2f        ..}x~.......>./
0x0030    2719 29dd 5353 482d 312e 302d 5353 485f        '.).SSH-1.0-SSH_
0x0040    5665 7273 696f 6e5f 4d61 7070 6572 0a00        Version_Mapper..
```

Above, the attacker is using a scanner program called ScanSSH to scan for SSH in order to exploit the known vulnerability.

```
02:50:56.203164 10.10.10.40.syslog > 10.10.10.70.syslog:  [udp sum ok]
```

```
udp 85 (ttl 64, id 10463, len 113)
0x0000    4500 0071 28df 0000 4011 291c 0a0a 0a28    E..q(...@.)....(
0x0010    0a0a 0a46 0202 0202 005d ae8b 3c33 343e    ...F.....]..<34>
0x0020    4175 6720 3139 2030 333a 3137 3a30 3720    Aug.19.03:17:07.
0x0030    7373 6864 5b32 3432 3636 5d3a 2066 6174    sshd[24266]:.fat
0x0040    616c 3a20 5265 6365 6976 6564 2070 6163    al:.Received.pac
0x0050    6b65 7420 7769 7468 2062 6164 2073 7472    ket.with.bad.str
0x0060    696e 6720 6c65 6e67 7468 2032 3633 3136    ing.length.26316
0x0070    38
```

Next, the attacker has used a program called SSH-2.0 GOBBLES (above) exploit to enable him to take over the machine.

```
02:50:02.953164 10.10.10.40.ssh > xxx.xxx.xxx.xxx.1154: P [tcp sum ok]
2099:2140(41) ack 20846 win 17376 <nop,nop,timestamp 655960554
54448309> (DF) (ttl 64, id 12215, len 93)
0x0000    4500 005d 2fb7 4000 4006 93a0 0a0a 0a28    E..]/.@.@......(
0x0010    xxxx xxxx 0016 0482 b9bd 2672 667b df00    ...=......&rf{..
0x0020    8018 43e0 13f0 0000 0101 080a 2719 29ea    ..C.........'.).
0x0030    033e d0b5 7569 643d 3028 726f 6f74 2920    .>..uid=0(root).
0x0040    6769 643d 3028 7768 6565 6c29 2067 726f    gid=0(wheel).gro
0x0050    7570 733d 3028 7768 6565 6c29 0a           ups=0(wheel).
```

Above, it can be seen that the attacker has obtained root privileges (uid=0[root[). After this, the attacker changed the root password, covered his tracks and used the computer for IRC.

Conclusion

Honeypots are a valuable tool to examine how attackers break into and attack computer systems. Honeypots must be used with care, or you may find your entire network damaged. Experience in computer security is necessary, both to recognize exploits and to discover new exploits as they take place.

Part 2 – Network Detects

Detect 1: 6/29/03 Scan socks proxy attempt

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Priority: 0]
10/31-20:00:53.696507 216.77.209.189:39627 -> 207.166.90.205:1080
TCP TTL:49 TOS:0x0 ID:32161 IpLen:20 DgmLen:40 ******S* Seq: 0x7B9DE6B
Ack: 0x7B9DE6B Win: 0x400 TcpLen: 20 [Xref => url
help.undernet.org/proxyscan/]

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Priority: 0]
10/31-20:03:07.846507 216.77.209.189:4440 -> 207.166.93.234:1080
TCP TTL:49 TOS:0x0 ID:9325 IpLen:20 DgmLen:40 ******S* Seq: 0x392A32FF
Ack: 0x392A32FF Win: 0x400 TcpLen: 20 [Xref => url
help.undernet.org/proxyscan/]

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Priority: 0]
10/31-20:05:22.036507 216.77.209.189:45276 -> 207.166.246.33:1080
TCP TTL:49 TOS:0x0 ID:34774 IpLen:20 DgmLen:40 ******S* Seq: 0x56B02723
Ack: 0x56B02723 Win: 0x400 TcpLen: 20 [Xref => url
help.undernet.org/proxyscan/]

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Priority: 0]
10/31-20:07:36.256507 216.77.209.189:37100 -> 207.166.140.161:1080
TCP TTL:49 TOS:0x0 ID:6333 IpLen:20 DgmLen:40 ******S* Seq: 0x5E5F5DB1
Ack: 0x5E5F5DB1 Win: 0x400 TcpLen: 20 [Xref => url
help.undernet.org/proxyscan/]

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Priority: 0]
10/31-20:09:50.436507 216.77.209.189:23552 -> 207.166.219.188:1080
TCP TTL:49 TOS:0x0 ID:9259 IpLen:20 DgmLen:40 ******S* Seq: 0x676E539A
Ack: 0x676E539A Win: 0x400 TcpLen: 20 [Xref => url
help.undernet.org/proxyscan/]

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Priority: 0]
10/31-20:12:04.596507 216.77.209.189:53816 -> 207.166.25.146:1080
TCP TTL:49 TOS:0x0 ID:41615 IpLen:20 DgmLen:40 ******S* Seq: 0x1F101DD0
Ack: 0x1F101DD0 Win: 0x400 TcpLen: 20 [Xref => url
help.undernet.org/proxyscan/]

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Priority: 0]
10/31-20.756507 216.77.209.189:60558 -> 207.166.37.159:1080
TCP TTL:49 TOS:0x0 ID:6982 IpLen:20 DgmLen:40 ******S* Seq: 0x29169515
Ack: 0x29169515 Win: 0x400 TcpLen: 20 [Xref => url
help.undernet.org/proxyscan/]

Return traffic from scanned host:

[**] [1:0:0] Return traffic from Socks port 1080 scan
[**]
[Priority: 0]
10/31-21:52:29.426507 207.166.87.157:63861 -> 216.170.250.223:9284
TCP TTL:124 TOS:0x0 ID:23437 IpLen:20 DgmLen:367 DF ***AP*** Seq:
0xE8FF3803 Ack: 0x1BBB2BD1 Win: 0x4248 TcpLen: 20

[**] [1:0:0] Return traffic from Socks port 1080 scan
[**]
[Priority: 0]
10/31-22:01:21.026507 207.166.87.157:64419 -> 216.209.145.189:6347
TCP TTL:124 TOS:0x0 ID:35459 IpLen:20 DgmLen:360 DF ***AP*** Seq:
0xF1936A65 Ack: 0x2CB9797A Win: 0x40E8 TcpLen: 20

[**] [1:0:0] Return traffic from Socks port 1080 scan
[**]
[Priority: 0]
10/31-22:07:33.436507 207.166.87.157:64665 -> 216.196.129.234:9174
TCP TTL:124 TOS:0x0 ID:46062 IpLen:20 DgmLen:371 DF ***AP*** Seq:
0xF7498BE8 Ack: 0x292284F7 Win: 0x4470 TcpLen: 20

[**] [1:0:0] Return traffic from Socks port 1080 scan
[**]
[Priority: 0]
10/31-00:09:54.406507 207.166.87.157:61797 -> 216.162.111.229:6791
TCP TTL:124 TOS:0x0 ID:3775 IpLen:20 DgmLen:364 DF ***AP*** Seq:
0x6539EAD6 Ack: 0xC8246143 Win: 0x40E8 TcpLen: 20

1. Source of trace:

This trace was obtained from www.incidents.org/logs/Raw/20021001 . The altered source IP range of 216.77.209.xxx and 216.77.216.xxx are class C. The destination IP range of 207.166.xxx.xxx is class B. The MAC addresses indicate a Cisco router. I speculate that the sensor is located close to the Internet router because of the wide range of potentially harmful traffic seen in the logs.

2. Detect Generated by:

The detect was generated by Snort, version 2.0.0 running on Windows 2000. The rule used was: alert tcp any any -> any 1080 (msg:"SCAN SOCKS Proxy attempt"; flags:S; sid:615; rev:3;). This rule was modified from a proxy rule obtained from ww.potatosoft.com. 1080 is the signature port for a Socks proxy attempt. The rule used to examine return traffic was: alert tcp 207.166.219.0/8 any -> 216.77.209.0/8 any (msg: "Return traffic from Socks port 1080 scan";).

3. Probability the source address was spoofed:

The source IP address and TTL do not change in the sample. The ID number jumps around instead of consistently increasing over time, but with fast scanning and only 16 bits to work with, the ID number may be wrapping around. This address is probably legitimate.

A summary of part of the trace is shown in the table below.

Time	Source IP	Port	Dest IP	Dest Port	ID	TTL
20:00:53	216.77.209.189	39627	207.166.90.205	1080	32161	49
20:03:07	216.77.209.190	4440	207.166.90.206	1080	9325	49
20:05:22	216.77.209.191	45276	207.166.90.207	1080	34774	49
20:07:36	216.77.209.192	37100	207.166.90.208	1080	6333	49
20:09:50	216.77.209.193	23552	207.166.90.209	1080	9259	49
20:12:04	216.77.209.194	53816	207.166.90.210	1080	41615	49
20:14:18	216.77.209.195	60558	207.166.90.211	1080	6982	49

The SYN scan occurs every two minutes and fourteen seconds. The random nature of the ports and the source IP incrementing by one for each scan suggest spoofed IPs. The fact that the IP ID numbers jump around so much also point toward spoofing.

4. Description of Attack

A scanning tool is used to probe networks for open port 1080. If an ACK is returned from port 1080, a session can be established. There are many programs freely available (see attack mechanism, below) on the Internet to gain use of a proxy server. It is likely that the source IP addresses will be spoofed during the scan, since the object of connecting to a proxy server is usually to hide one's identity. Actually, scanning is not always needed since there are web sites that publish open proxy servers.

5. Attack mechanism

Socks is an approved Internet standard, described in RFC 1928. Socks is a protocol that allows a server to connect to a client without a direct connection, the socks server acting as an intermediary. When a client computer needs to connect to an application or data server, it connect to a Socks proxy server, which then connects to the application or data server. Socks servers also can provide security and firewall protection.

A proxy server that is not configured correctly is an inviting target for hackers, since the proxy server can hide the identity of the hacker and be used for scans, attacks, exploits, etc. An open or insecure server is one that will accept requests from any computer. Spammers often use insecure proxy servers to distribute their junk email. Socks servers should only allow connections from their own local networks. (Socks.permeo.com)

An attacker sends SYN scans until he finds a server with an open port 1080. Nmap or any other scanning tool can be used. He can then connect to the proxy server with a self-written program or tool, or programs can be downloaded. A program called ProxyKing will find open proxy servers (http://www.proxyking.com/) Below is a partial example of a c program available on the internet that allows a user to exploit a Socks proxy server: (http://www.imasy.or.jp/~gotoh/ssh/connect.c). The comments give the novice hacker concise instructions on how to use the program. (Goto)

```
/************************************************************************
 * connect.c -- Make socket connection using SOCKS4/5 and HTTP tunnel.
 *
 * Copyright (c) 2000, 2001 Shun-ichi Goto
 * Copyright (c) 2002, J. Grant (English Corrections)
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
Foundation, Inc., 59 Temple Place - Suite 330, Boston,
MA  02111-1307, USA.
 *
 *
-------------------------------------------------------
 * PROJECT:  My Test Program
 * AUTHOR:   Shun-ichi GOTO <gotoh@taiyo.co.jp>
 * CREATE:   Wed Jun 21, 2000
 * REVISION: $Revision: 1.68 $
 *
-------------------------------------------------------
 *
```

```
* Getting Source
* ==============
*   Recent version of 'connect.c' is available from
*   Pre-compiled biniary for Win32 is also available:
*
*
* How To Use
*   You can specify proxy method in an environment
*   variable or in a command line option.
*   usage:  connect [-dnhs45] [-R resolve] [-p local-port] [-w sec]
*                   [-H [user@]proxy-server[:port]]
*                   [-S [user@]socks-server[:port]]
*                   host port
*
*   "host" and "port" is for the target hostname and
*    port-number to connect to.
*
*   The -H option specifies a hostname and port number of the http proxy
*   server to relay. If port is omitted, 80 is used.
*   You can specify this
*   value in the environment variable HTTP_PROXY and pass the -h option
*   to use it.
```

6. Correlations:

Port 1080 is shown as one of the most frequently scanned ports at www.incidents.org. In the raw log from the same website that this detect is based on (www.incidents.org/logs/Raw/20021001), about 3/4 of the activity consisted of scans to port 1080, so it is very common.

Microsoft.com's support site has directions on how to secure a socks server. Www.technerd.net has an Insecure Proxy Scanning Information Center page.

7. Evidence of active targeting:

The attacker did target all of the IP addresses in a class B range, but it is hard to say if this was not just part of a larger automated scan that included this network. Scanning for open proxy servers is more of a search than an attack, so malice towards a particular network is probably not the goal. A given network having a history of leaving their proxy servers open would certainly get targeted more often.

8. Severity:

Since the data was taken from incidents.org, the following factors will be estimated based on probable use.

Criticality = 4. Socks servers provide applications, data and security functions to clients. Losing the server could cause anything from inconvenience to severe disruption.

Lethality = 1. Lethality is limited, because the general idea is to use the proxy server for hiding identity. Unauthorized use of a proxy server is like a flea on a dog – the flea may be irritating but it does not generally kill the dog. Of course, enough unauthorized activity on a proxy server could cause its normal functions to slow down considerably or even grind to a halt. A computer known to have an open proxy port could be probed for other weaknesses, however. These proxy servers are easily found, since web sites routinely publish lists of open proxy servers. I couldn't find any data on attacks on open proxy servers compared with other servers. It would be interesting to see if proxy servers were compromised at a greater rate than other servers. You could wind up with the odd situation of one hacker shutting down a server used by a group of other hackers, spammers, etc.

System Countermeasures = 2. One server did answer a port 1080 scan with the ACK flag set, so it did respond to outside traffic. If this server was a socks server, it was configured wrong – it should not have responded.

Network Countermeasures = 4. In the sample I looked at, I could find no established sessions using port 1080 (I did not find the final ack of the three-way handshake), so the traffic may have been blocked by a firewall or router.

(4 + 1) – (2 + 4) = -1.

9. Defensive recommendation

It is easy to stop the problem of hijacked proxy servers – limit traffic to the local network. This can be done within the socks setup or via a firewall. If it is not feasible to stop non-local traffic entirely, it could at least be limited to an approved range of addresses.

10. Multiple choice question:

What is the primary reason proxy servers are such inviting targets?

A. You can hack into them and steal information.
B. They are commonly used for Distributed Denial of Service attacks.
C. They hide the identity of spammers and hackers.

D. You can download free music from them.

Answer: C.

Detect 2/3, 7/1/03 - More Fragments and Don't Fragment Set On.

[**] [1:0:0] BAD TRAFFIC bad frag bits [**]
[Priority: 0]
10/10-05:13:56.246507 62.252.8.218 -> 32.245.26.39
TCP TTL:111 TOS:0x0 ID:15144 IpLen:20 DgmLen:1468 DF
MF
Frag Offset: 0x0000 Frag Size: 0x0014

[**] [1:0:0] BAD TRAFFIC bad frag bits [**]
[Priority: 0]
10/10-05:14:21.216507 62.252.8.218 -> 32.245.26.39
TCP TTL:111 TOS:0x0 ID:17231 IpLen:20 DgmLen:1468 DF
MF
Frag Offset: 0x0000 Frag Size: 0x0014

[**] [1:0:0] BAD TRAFFIC bad frag bits [**]
[Priority: 0]
10/10-05:15:14.116507 62.252.8.218 -> 32.245.26.39
TCP TTL:111 TOS:0x0 ID:20560 IpLen:20 DgmLen:1468 DF
MF
Frag Offset: 0x0000 Frag Size: 0x0014

[**] [1:0:0] BAD TRAFFIC bad frag bits [**]
[Priority: 0]
10/10-05:16:55.566507 62.252.8.218 -> 32.245.26.39
TCP TTL:111 TOS:0x0 ID:26680 IpLen:20 DgmLen:1468 DF
MF
Frag Offset: 0x0000 Frag Size: 0x0014

[**] [1:0:0] BAD TRAFFIC bad frag bits [**]
[Priority: 0]
10/10-06:26:36.226507 80.4.24.124 -> 32.245.236.186
TCP TTL:111 TOS:0x0 ID:2141 IpLen:20 DgmLen:1468 DF MF
Frag Offset: 0x0000 Frag Size: 0x0014

[**] [1:0:0] BAD TRAFFIC bad frag bits [**]
[Priority: 0]
10/10-06:27:06.686507 80.4.24.124 -> 32.245.236.186
TCP TTL:111 TOS:0x0 ID:3161 IpLen:20 DgmLen:1468 DF MF
Frag Offset: 0x0000 Frag Size: 0x0014

[**] [1:0:0] BAD TRAFFIC bad frag bits [**]
[Priority: 0]
10/10-06:30:01.636507 80.4.24.124 -> 32.245.236.186
TCP TTL:111 TOS:0x0 ID:10904 IpLen:20 DgmLen:1468 DF
MF
Frag Offset: 0x0000   Frag Size: 0x0014

[**] [1:0:0] BAD TRAFFIC bad frag bits [**]
[Priority: 0]
10/10-06:33:56.346507 80.4.24.124 -> 32.245.236.186
TCP TTL:111 TOS:0x0 ID:22089 IpLen:20 DgmLen:1468 DF
MF
Frag Offset: 0x0000   Frag Size: 0x0014

[**] [116:46:1] (snort_decoder) WARNING: TCP Data
Offset is less than 5! [**]
10/10-09:53:55.446507 62.13.27.29:0 -> 32.245.113.30:0
TCP TTL:234 TOS:0x0 ID:0 IpLen:20 DgmLen:40
*****R** Seq: 0x1EC00EE Ack: 0x1EC00EE Win: 0x0
TcpLen: 0

1. Source of trace:

This trace was obtained from www.incidents.org/logs/Raw/20020910 . The altered source IP range are most likely spoofed and not significant. The destination IP range of 32.245.xxx.xxx is class B. The MAC addresses indicate a Cisco router, probably located close to the Internet router because of the wide range of potentially harmful traffic seen in the logs.

2. Detect Generated by:

The detect was generated by Snort, version 2.0.0 running on Windows 2000. The rule used was: alert ip any any -> any any (msg:"BAD TRAFFIC bad frag bits"; fragbits:MD;). An attempt was made to find any return tcp or icmp traffic, but none was found.

3. Probability the source address was spoofed:

The source address was probably spoofed. The headers are nearly identical except for the source address. The fragment sizes are all 20 bytes, the flags set are always MF and DF, the offset is always 0 and the ID number is constantly changing. Real fragments would not have these properties. In a real set of fragments, the more fragments flag would be set on for all but the last fragment, but the don't fragment flag would never be set on. The ID number would be the

same for the entire set of fragments. The fragment offset would start at 0 and increase by the size of the fragment for each packet. These fragments were obviously crafted, and they came from a variety of source addresses, so it is likely that the addresses were spoofed.

4. Description of Attack

Multiple groups of four crafted fragments were sent from one destination address to the 32.245 network, to the same port. The time between packets increased for each of the four packets. A typical series of time intervals for the second, third and fourth packets was about 30 seconds, three minutes and four minutes after the first packet. It is not known why the same packet was sent four times at wide intervals. The frag2 preprocessor was tried but no complete fragments were assembled for the destination IPs in the above sample. So this was not a Denial of Service attack such as ssping, where highly fragmented traffic is used to clog the routers. There was no attempt at using overlapping or missing fragments to avoid the IDS or as an insertion attack.

5. Attack mechanism

This appears to be a scan of some sort. One could speculate that by setting both the more fragments and the don't fragment flags, there was an attempt at a Denial of Service attack as the router waited for more fragments, but the packets were sent at such a slow rate that this would not have happened. It looks like the attacker wanted to get an ICMP message back for reconnaissance purposes. However, no ICMP (or TCP) data was sent back to the source IPs in the sample above.

6. Correlations:

None found, except in references back to GIAC pages. It is perhaps not surprising, since this scan did not appear to elicit any response from the targeted systems. This could have been some sort of experimentation just to see what would happen.

7. Evidence of active targeting:

The consistency of this data points to active targeting. About ten groups of four records were aimed at the 32.245 address range over a 24 hour period. It is probably not accidental or garbage data because of the strangeness of the information. The more fragments and don't fragment fields should never be set at the same time.

8. Severity:

Criticality – 3. The data used was from incidents.org, but it could have been from any server.

Lethality – 2. This traffic did not even elicit an ICMP response. What it would do on other systems is not known.

System Countermeasures – 3. This is unknown since it looks like the router dropped the packets before they got to the host.

Network Countermeasures – 4. It appeared that the MF DF scan was totally ineffective in doing any damage or gaining information.

Severity = (3 + 2) – (3 + 4) = -2

9. Defensive recommendation

It looks like the firewall dropped these packets before they reached the host. Since fragmentation attacks such as Teardrop have been around for a while, routers and firewalls have been configured to stop most of these types of attacks. Since Ethernet is so predominant, MTUs tend to be pretty consistent nowadays, fragmentation is relatively rare, and any fragment is worth scrutinizing.

10. Multiple choice question:

A datagram has become fragmented on its travels through the internet. Comparing the second fragment to the first, you will see:

A. A higher fragment offset number
B. A higher fragment ID number
C. The More Fragments flag set for both fragments
D. A and C
E. A, B and C

Answer: D.

Reply 1 to detect:

1. A lack of correlations seems odd. I can recall seeing several discussions on similar detects, and when I do a search on Google using the snort alert "BAD TRAFFIC bad frag bits" I get a couple thousand hits. Did you look at those hits and for some reason feel that none correlated with your detect?

2. You mention a MAC address, but do not include any trace output to validate you clain, could you include it in a trace? It would also be helpful to see the IP

headers in a raw format so that we can see the bits have been set rather than just an interpretation by Snort. (I have a reason for asking that based on what I saw in the search by Google).

Comments on Reply 1 to detect:

I did restrict my search too much. I looked for the string "MF DF", and the "BAD TRAFFIC bad frag bits" string does generate more replies. As far as the missing MAC addresses, they seem to have been lost in the fog of battle. I am reasonably sure I saw the Cisco code.

Reply 2 to detect

Interesting detect. I think that you can do a little more correlation, as pointed out by others. My comments are inline below.

Yes, one could speculate that this could be a DoS against a router, but Is there anything to back you up there? Any known vulnerabilities in anything that would back you up on this?

Comments on Reply 1 to detect:
In my searches for similar flag settings, I found many instances of the MF DF flags, but not any good explanations as to why they were the way they were.

Detect 3, 7/6/03 – UPNP malformed advertisement

[**] [1:1384:2] MISC UPNP malformed advertisement [**]
[Classification: Misc Attack] [Priority: 2]
04/26-13:34:55.198004 2:A0:24:BB:7E:EB ->
1:0:5E:7F:FF:FA type:0x800 len:0x1CB
192.168.0.1:1900 -> 239.255.255.250:1900 UDP TTL:4
TOS:0x0 ID:28749 IpLen:20 DgmLen:445
Len: 417
[Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0877][Xref
=>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0876]

[**] [1:1384:2] MISC UPNP malformed advertisement [**]
[Classification: Misc Attack] [Priority: 2]
04/26-13:34:55.227231 2:A0:24:BB:7E:EB ->
1:0:5E:7F:FF:FA type:0x800 len:0x183
192.168.0.1:1900 -> 239.255.255.250:1900 UDP TTL:4
TOS:0x0 ID:28751 IpLen:20 DgmLen:373

Len: 345
[Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0877][Xref
=>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0876]

[**] [1:1384:2] MISC UPNP malformed advertisement [**]
[Classification: Misc Attack] [Priority: 2]
04/26-13:34:55.277475 2:A0:24:BB:7E:EB ->
1:0:5E:7F:FF:FA type:0x800 len:0x1B9
192.168.0.1:1900 -> 239.255.255.250:1900 UDP TTL:4
TOS:0x0 ID:28753 IpLen:20 DgmLen:427
Len: 399
[Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0877][Xref
=>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0876]

[**] [1:1384:2] MISC UPNP malformed advertisement [**]
[Classification: Misc Attack] [Priority: 2]
04/26-13:34:55.317915 2:A0:24:BB:7E:EB ->
1:0:5E:7F:FF:FA type:0x800 len:0x1C7
192.168.0.1:1900 -> 239.255.255.250:1900 UDP TTL:4
TOS:0x0 ID:28755 IpLen:20 DgmLen:441
Len: 413
[Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0877][Xref
=>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0876]

The following entries are from my home computer:

[**] [1:1384:2] MISC UPNP malformed advertisement [**]
[Classification: Misc Attack] [Priority: 2]
07/05-20:53:26.686638 192.168.1.1:1900 ->
239.255.255.250:1900
UDP TTL:4 TOS:0x0 ID:3355 IpLen:20 DgmLen:306
Len: 278
[Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0877][Xref
=>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0876]

[**] [1:1384:2] MISC UPNP malformed advertisement [**]
[Classification: Misc Attack] [Priority: 2]
07/05-20:53:26.690060 192.168.1.1:1900 ->

239.255.255.250:1900
UDP TTL:4 TOS:0x0 ID:3356 IpLen:20 DgmLen:362
Len: 334
[Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0877][Xref
=>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0876]

1. Source of trace

I initially saw this alert on my home computer. Since my computer triggered the alert, I wasn't too alarmed, but in researching the alert I found some information on a vulnerability on Windows systems. The first four log entries above were found from the site
http://roninsoftwaregroup.com.hosting.domaindirect.com/mysql/alert.ids.txt
and the last two entries are from my home computer.

2. Detect Generated by:

My detect was generated by Snort, version 2.0.0 running on Windows XP. The standard snort.conf file was used to generate the alerts. The detects from the Internet look like they also came from Snort, using the same rule set. Henceforth, I will be referring only to the trace that did not come from my home computer.

3. Probability the source address was spoofed:

The issue of spoofing is moot here because the source address is private, probably generated behind a router. The contents of the packet could have been crafted. The datagram lengths are all different, which could be normal or a sign of crafting.

4. Description of Attack:

Universal Plug and Play (UPnP) is a set of protocols designed to allow computers and devices to work together with little or no configuration. The NOTIFY directives have a vulnerability to buffer overflows, the result being the ability of a hacker to run commands on a target computer with administrative privileges.

5. Attack mechanism:

A couple of attack mechanisms have been discovered. The first method involves sending malformed advertisements at various speeds in order to cause access violations on the target machine, which caused pointers to be overwritten. This

allows an attacker to insert commands and gain control of the target computer. The second attack depends on Windows XP searching for an "Internet Gateway Device", which Microsoft invented to encourage manufacturers to make UPnP-compatible devices. An attacker can force the XP client to connect back to a specified IP address.
http://www.eeye.com/html/Research/Advisories/AD20011220.html

6. Correlations:

CERT has a page with links about this problem: http://www.kb.cert.org/vuls/id/951555. Microsoft has patches available at: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-059.asp
.

7. Evidence of active targeting:

The two sample traces do not indicate active targeting. The datagrams sent from my home computer match the other sample. Although this can be a dangerous attack on an unpatched machine, XP routinely spits out these packets that trigger Snort. It may be possible to reconfigure the rule so that the normal discovery process is ignored and malicious UPnP traffic is detected. The sample data I have represent false positives.

8. Severity:

Criticality - 3. This is an estimate, since my sample destination IPs are multicast addresses.

Lethality – 5. If successful, the attacker can gain control of the target computer.

System Countermeasures – 3. Assuming the computer is patched, this exploit will not be successful. I turned off the Universal Plug and Play service on my computer, the average user probably does not. Patched or not, the furor over this bug seems to have died
down in recent months. It may be that the exploit is difficult to actually carry out even though the potential is there.

Network Countermeasures – 3. Snort can detect these malformed packets, and resumably other IDS systems can also.

Severity = (3 + 5) – (3 + 3) = 2.

9. Defensive Recommendation:

Patches are available from Microsoft. Universal Plug and Play services can be turned off. If UPnP catches on, there will probably be other attempts at attacking this protocol.

10. Multiple Choice Question

What is the greatest weakness of Universal Plug and Play?

A. It can be used in a DDoS attack.
B. It is a relatively new protocol with bugs and vulnerabilities.
C. It can give administrative control to an attacker.
D. All of the above
E. A and B
F. B and C

Answer: F

Part 3 – Analyze This

1. Executive Summary

The analysis performed is based on data collected during the June 13, 2003 to June 18, 2003 period. Alert, scan and out of spec files were combined and summarized with Perl scripts. High volume and dangerous exploits observed in the logs were analyzed.

The alert logs show a high volume of potentially malicious activity. There is also a large amount of data resulting from file sharing and other student activity. There is a lot of traffic outbound from the network that needs to be examined, such as port scanning.

The sheer volume of alerts points out the need to stay current with upgrades, to limit services, to have a coherent security policy, and to be aware of the latest trends in malicious activity.

2. Files Analyzed:

Scan Files	Alert Files	OOS Files
scans.030613	alert.030613	oos_rpt_030613
scans.030614	alert.030614	oos_rpt_030614
scans.030615	alert.030615	oos_rpt_030615
scans.030616	alert.030616	oos_rpt_030616
scans.030617	alert.030617	oos_rpt_030617

3. Lists of Defects

Number of Alerts by date

91331 June 13, 2003
108305 June 14, 2003
59640 June 15, 2003
96016 June 16, 2003
118426 June 17, 2003

Alerts by message type

146758 CS WEBSERVER - external web traffic

68442 [UMBC NIDS IRC Alert] IRC user /kill detected
57130 SMB Name Wildcard
45351 spp_http_decode: IIS Unicode attack detected
25960 EXPLOIT x86 NOOP
22752 MY.NET.30.4 activity
18320 MY.NET.30.3 activity
15623 Queso fingerprint
10661 spp_http_decode: CGI Null Byte attack detected
7435 High port 65535 udp - possible Red Worm - traffic
6361 High port 65535 tcp - possible Red Worm - traffic
5648 TCP SRC and DST outside network
1635 IDS552/web-iis_IIS ISAPI Overflow ida nosize
1342 Null scan!
995 FTP DoS ftpd globbing
800 NMAP TCP ping!
663 External RPC call
645 IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
534 [UMBC NIDS IRC Alert] Possible Incoming XDCC Send Reque...
410 connect to 515 from outside
392 SUNRPC highport access!
329 SMB C access
326 Possible trojan server activity
303 NIMDA - Attempt to execute cmd from campus host
255 DDOS mstream handler to client
254 scan (Externally-based)
214 Incomplete Packet Fragments Discarded
195 SNMP public access
73 Notify Brian B. 3.54 tcp
71 Notify Brian B. 3.56 tcp
71 EXPLOIT x86 setuid 0
69 EXPLOIT x86 setgid 0
68 [UMBC NIDS IRC Alert] Possible sdbot floodnet detected ...
68 TFTP - Internal UDP connection to external tftp server
68 FTP passwd attempt
61 ICMP SRC and DST outside network
36 EXPLOIT x86 stealth noop
30 RFB - Possible WinVNC - 010708-1
28 NIMDA - Attempt to execute root from campus host
28 TFTP - Internal TCP connection to external tftp server
23 NETBIOS NT NULL session
23 DDOS shaft client to handler
22 Attempted Sun RPC high port access
19 Probable NMAP fingerprint attempt
18 EXPLOIT NTPDX buffer overflow
12 [UMBC NIDS IRC Alert] K\:line'd user detected
11 TCP SMTP Source Port traffic

9 IRC evil - running XDCC
7 TFTP - External UDP connection to internal tftp server
5 Tiny Fragments - Possible Hostile Activity
4 External FTP to HelpDesk MY.NET.70.50
3 Traffic from port 53 to port 123
2 SYN-FIN scan!
2 External FTP to HelpDesk MY.NET.83.197
2 External FTP to HelpDesk MY.NET.53.29
2 External FTP to HelpDesk MY.NET.70.49
2 [UMBC NIDS IRC Alert] User joining XDCC channel detecte...
1 Fragmentation Overflow Attack
1 Bugbear@MM virus in SMTP
1 DDOS mstream client to handler

Number of Scans by Type

6971342 UDP scan (Externally-based)
2158978 SYN scan (Externally-based)
525 UNKNOWN scan (Externally-based)
416 FIN scan (Externally-based)
387 NULL scan (Externally-based)
350 INVALIDACK scan (Externally-based)
155 NOACK scan (Externally-based)
66 VECNA scan (Externally-based)
22 XMAS scan (Externally-based)
16 NMAPID scan (Externally-based)
7 FULLXMAS scan (Externally-based)
7 scan (Externally-based)
3 SPAU scan (Externally-based)
3 SYNFIN scan (Externally-based)
2 14 scan (Externally-based)
1 ******S* scan (Externally-based)
1 15 scan (Externally-based)
1 130.85.141.21:4039 scan (Externally-based)
1 13 scan (Externally-based)
1 -> scan (Externally-based)
1 130.85.70.207:12203 scan (Externally-based)
1 130.85.153.190:6257 scan (Externally-based)
1 218.98.64.33:57052 scan (Externally-based)
1 130.85.1.3:32814 scan (Externally-based)
1 130.85.18.36:1357 scan (Externally-based)

Top 10 Alerts – Description

Following is a brief description of the top ten alerts by frequency of occurrence.

1. CS Webserver – This traffic is directed at port 80, the http port. Looking in the out of spec reports for the five day period, it can be seen that certain IPs are making multiple scans of port 80, perhaps for reconnaissance. These scans tend to increment the source port by 1 each time, with each scan spaced about ten seconds apart.

2. UMBC NIDS IRC Alert - IRC user /kill detected. IRC stands for internet relay chat, invented in 1988 by Jarkko Oikarinen. It is used for connecting groups of people together via the Internet for discussions. A /kill signal will remove a user from an IRC session. These signals can be used legitimately to remove abusive users from a chat room, or they can be used maliciously. Many of these packets are coming from the 195.159.000.000 network, port 6667. This can be part of a Trojan attack. The Trojan is spread via email, which leads to a distributed denial of service attack against IRC servers. Many thousands of computers can be involved in an attack, halting the IRC service. (Olavsrud)

3. SMB Name Wildcard – These probes are directed at the MY.NET subnet, the destination port always being 137, the netbios-ns port. SMB is used to share information over networks. Currently, port 137 is the most attacked port according to incidents.org, so it is obviously a good target for hackers. When coming from an external network, a port 137 probe is used to gain information about workstation names, domains and logged-in users. (Finchaven.com)

4. spp_http_decode: IIS Unicode attack detected – This attack is directed at port 80 of a server running Microsoft Internet Information Server. There is a divergence of opinion on the severity of this attack. Some analysts say that it is the result of errant search engines while others say it can be used to gain access to cmd.exe on a Windows server. There are large numbers of these packets both coming into the MY.NET network and leaving the network. Unless MY.NET was being used as a proxy by attackers, I believe the packets originating from MY.NET are false positives. The incoming packets could be part of a Nimda attack. (Mcabee)

5. EXPLOIT x86 NOOP – No-ops are often involved in a buffer overflow attack, and this is no exception. This attack utilizes a buffer overflow to perform an attack on Internet Information Server. A user can get a directory listing with this exploit and later gain administrative control. There are patches available for this problem. Most, but not all of these are packets are directed at port 119, the nntp, or news transfer protocol port. Some of the destination addresses resolve to news servers (using nslookup), while others are from .gov addresses, so it looks like there are some false positives here. (Richard)

6. MY.NET.30.4 activity – This activity is directed at MY.NET.30.4 port 80. It turns out to be harmless search and indexing activity from Inktomi and Google. If

the destination computer is not used as a web server, it might be advisable to block traffic to port 80 entirely or rewrite the IDS rules to avoid false positives.

7. MY.NET.30.3 activity – This activity is directed at MY.NET.30.3, a variety of ports. Typically, traffic is sent from one destination IP address/port to a specific port at MY.NET.30.3, such as 524 and 8080. The source records are sent many times a second, so it looks almost like some kind of denial of service attempt. The source IP numbers do resolve to actual names using nslookup.

8. Queso fingerprint – Queso is a program like nmap which attempts to discover a host's operating system and version by sending a variety of datagrams to a destination and analyzing the return packets. Queso uses a variety of techniques to fingerprint operating systems, from sending FIN probes to setting TCP options. In fact, some of the alerts can be found in the out of spec logs, and they have a variety of TCP options set, five options in all of the cases I examined. Curiously enough, the source and destination IPs and ports did not change for all of the scans, nor did anything in the IP header except the ID number. It looked the source computer was stuck in a loop, because a working fingerprint program would be trying different things to map the operating system of the destination computer. Or perhaps the hacker had misconfigured Queso. (Fyodor)

9. spp_http_decode: CGI Null Byte attack detected - The majority of these alerts were outgoing traffic from MY.NET to popular .com sites and were probably false positives. A '%00' in an http request will trigger this warning. Sites using urlencoded binary data for cookies can be the cause of this false positive. The incoming packets triggering this alarm that I examined came from www.shockwave.com, a legitimate site. Null bytes can be injected into commands to pass malicious code. Null bytes can also fool a web site into thinking a different file type had been requested. So it is possible for a null byte alarm to be legitimate. (Zenomorph)

10. High port 65535 udp - possible Red Worm – traffic The Red Worm exploit is another attack on the vulnerabilities of the Microsoft IIS. The worm selects random IP addresses and sees if port 80 is open. If it is, it send a copy of the buffer overflow attack to that machine. Computers that are low on the list of random IP addresses are likely to be probed again and again, resulting in a denial of service attack. Computers can be infected multiple times, with hundreds of worm threads running at one time. (CIAC)

Other Statistics Generated from Alert and Scan Logs

Alerts by source IP

8979	68.49.35.0
7709	207.151.67.140

6397	205.160.101.121
4801	68.81.2.19
4464	193.225.219.29
3770	217.35.117.209
3686	213.156.45.224
3540	169.254.45.176
2518	61.22.152.64
2330	192.168.2.21
2007	216.231.171.27
1991	62.70.32.143
1964	216.39.48.127
1866	80.205.39.114
1504	64.124.5.10
1499	217.88.127.125
1431	66.168.226.143
1397	68.33.11.236
1392	67.74.230.212
1325	210.101.152.2
1304	80.15.10.217
1207	138.88.165.81
1029	216.87.56.44
1000	24.35.42.249
995	81.49.150.21
980	68.55.52.234
888	134.192.42.4
719	195.243.41.154
710	211.126.198.22
663	130.227.86.130
631	12.65.37.251
629	66.196.73.45
596	68.65.111.226
580	66.196.72.94
579	62.103.247.147
561	66.196.72.90
554	66.196.72.54
551	217.21.39.18
547	66.196.72.40
529	66.196.72.60
528	213.140.8.170
526	66.196.72.78
525	66.196.72.57
522	66.196.72.35
520	66.196.72.25
509	66.196.65.24
504	217.128.169.110
504	66.196.72.51

499	66.196.72.109
489	66.196.72.16
486	66.196.72.23
482	66.196.72.59
481	66.196.72.12
478	66.196.72.19
478	66.196.72.15
475	66.196.72.31
474	66.196.72.53
470	203.200.106.39
468	66.196.72.17
467	66.196.72.14
465	66.196.72.52
460	66.196.72.33

Alerts by relationship - external

8677	68.49.35.0->MY.NET.30.3
6395	205.160.101.121->MY.NET.83.100
4688	68.81.2.19->MY.NET.30.3
4464	193.225.219.29->MY.NET.100.165
2864	207.151.67.140->MY.NET.100.158
2817	207.151.67.140->MY.NET.130.157
2518	61.22.152.64->MY.NET.15.216
2007	216.231.171.27->MY.NET.70.207
1963	216.39.48.127->MY.NET.100.165
1431	66.168.226.143->MY.NET.30.4
1401	213.156.45.224->MY.NET.86.19
1398	217.35.117.209->MY.NET.110.224
1397	68.33.11.236->MY.NET.30.4
1393	217.35.117.209->MY.NET.86.19
1392	67.74.230.212->MY.NET.30.4
1207	138.88.165.81->MY.NET.30.3
1128	64.124.5.10->MY.NET.100.165
1123	217.88.127.125->MY.NET.130.122
995	81.49.150.21->MY.NET.114.116
896	24.35.42.249->MY.NET.30.3
888	134.192.42.4->MY.NET.30.4
884	68.55.52.234->MY.NET.30.3
709	211.126.198.22->MY.NET.153.145
626	66.196.73.45->MY.NET.100.165
606	80.205.39.114->MY.NET.110.224
600	213.156.45.224->MY.NET.110.224
596	68.65.111.226->MY.NET.30.4
591	80.205.39.114->MY.NET.86.19
551	217.21.39.18->MY.NET.100.165

528 213.140.8.170->MY.NET.100.165
470 203.200.106.39->MY.NET.100.165
455 80.15.10.217->MY.NET.86.19
454 66.196.72.60->MY.NET.30.4
451 66.196.72.23->MY.NET.100.165
450 66.196.72.17->MY.NET.100.165
450 66.196.72.54->MY.NET.100.165
447 66.196.72.14->MY.NET.100.165
447 66.196.72.52->MY.NET.100.165
447 66.196.72.18->MY.NET.100.165
444 66.196.72.15->MY.NET.100.165
442 80.15.10.217->MY.NET.110.224
442 66.196.72.16->MY.NET.100.165
440 66.196.72.12->MY.NET.100.165
439 66.196.72.28->MY.NET.100.165

Alerts by relationship / ports - internal

19379 MY.NET.83.100
4940 MY.NET.153.185
2852 MY.NET.97.71
2702 MY.NET.97.20
2519 MY.NET.15.216
1567 MY.NET.70.207
1435 MY.NET.153.190
1280 MY.NET.97.41
1277 MY.NET.97.98
1251 MY.NET.163.76
1153 MY.NET.168.98
1144 MY.NET.91.151
1144 MY.NET.97.194
1129 MY.NET.97.248
1020 MY.NET.75.107
915 MY.NET.97.219
858 MY.NET.97.204
771 MY.NET.83.171
730 MY.NET.84.216
665 MY.NET.97.40
661 MY.NET.168.229
650 MY.NET.150.121
649 MY.NET.97.84
645 MY.NET.97.80
632 MY.NET.153.201
628 MY.NET.81.58
589 MY.NET.17.54
566 MY.NET.153.116

554	MY.NET.97.37
541	MY.NET.97.212
530	MY.NET.53.41
512	MY.NET.97.172
511	MY.NET.153.198
495	MY.NET.98.14
465	MY.NET.97.52
444	MY.NET.153.115
439	MY.NET.97.198
419	MY.NET.153.187
408	MY.NET.97.63
390	MY.NET.88.150
369	MY.NET.97.96

Alerts by source ports - internal

2527	1074
1953	6257
1567	12203
1322	1249
475	3633
460	4029
386	4026
371	4069
332	4027
330	1690
269	80
268	4054
258	3331
256	4321
255	12754
239	3627
234	1240
233	4330
227	2060
224	2177
205	2059
196	2190
194	4059
182	4630
181	2436
179	2189
165	2435
162	2366
160	4111
160	4074

157 4110
155 1239
153 1031
153 1689
150 1144
144 2176
142 65535

Alerts by relationship - internal to external

12630 MY.NET.83.100->64.235.110.34
5374 MY.NET.83.100->208.194.163.37
2505 MY.NET.15.216->61.22.152.64
2476 MY.NET.97.20->203.161.233.132
1503 MY.NET.97.71->207.200.86.97
1479 MY.NET.70.207->216.231.171.27
1251 MY.NET.163.76->209.116.81.5
1210 MY.NET.153.185->218.153.6.33
1206 MY.NET.97.71->207.200.86.66
1144 MY.NET.91.151->212.161.35.251
1115 MY.NET.97.194->216.241.219.14
1102 MY.NET.97.248->203.161.233.132
915 MY.NET.97.219->211.58.254.253
895 MY.NET.83.100->155.207.19.204
858 MY.NET.153.185->218.153.6.197
672 MY.NET.153.185->211.233.29.52
667 MY.NET.168.98->211.233.29.5
650 MY.NET.150.121->199.104.95.15
636 MY.NET.168.229->192.151.53.10
632 MY.NET.153.201->216.26.171.19
628 MY.NET.97.40->202.103.69.100
604 MY.NET.153.185->218.153.6.244
511 MY.NET.153.198->216.241.219.22
479 MY.NET.83.100->205.160.101.121
361 MY.NET.153.187->216.26.171.19
327 MY.NET.97.41->61.78.53.27
295 MY.NET.97.172->211.239.164.250
293 MY.NET.153.190->220.55.212.137
289 MY.NET.97.52->211.117.63.214
287 MY.NET.151.99->192.151.53.10
282 MY.NET.98.14->211.233.29.54
270 MY.NET.83.171->220.90.215.249
262 MY.NET.97.212->211.234.108.26
256 MY.NET.153.190->219.160.152.151
255 MY.NET.84.235->80.100.101.176
252 MY.NET.84.216->211.239.164.248

241 MY.NET.97.41->211.176.60.147
239 MY.NET.97.217->211.233.53.235
232 MY.NET.153.185->211.233.79.236
223 MY.NET.98.36->211.239.164.250
218 MY.NET.88.150->211.233.29.54
218 MY.NET.97.198->64.12.54.24
216 MY.NET.97.204->211.39.133.122
210 MY.NET.7.26->192.151.53.10

Alerts by destination port - internal

192275 80
57113 137
14829 524
13923 21
6498 113
6451 25
6038 51443
3517 3019
2519 1074
2024 12203
1741 6257
692 8009
663 111
467 2200
414 32771
410 515
396 2736
352 139
261 53
226 4662
220 8088
195 161
181 119
149 143
127 443
122 1624
98 58115
97 65535
90 6346
65 1034
62 8080
59 3347
55 2335
52 81
48 8888

47 8000

Alerts by destination port - external

12630 64.235.110.34
5374 208.194.163.37
3578 203.161.233.132
2727 67.80.77.94
2505 61.22.152.64
1592 192.151.53.10
1566 207.200.86.97
1479 216.231.171.27
1394 202.103.69.100
1345 218.153.6.33
1266 207.200.86.66
1251 209.116.81.5
1208 216.26.171.19
1144 212.161.35.251
1115 216.241.219.14
996 211.239.164.250
916 211.58.254.253
895 155.207.19.204
858 218.153.6.197
789 211.233.29.5
709 199.244.218.42
696 211.233.29.52
650 199.104.95.15
649 64.12.54.25
643 220.90.215.249
623 64.12.54.24
621 64.12.54.217
479 205.160.101.121
444 211.39.133.122

Scan by source IP - external

2526454 130.85.1.3
420331 130.85.1.4
388692 130.85.83.170
381910 130.85.153.190
250322 130.85.97.16
245000 130.85.100.230
191746 130.85.88.198
191248 130.85.97.37
160003 130.85.97.129
156294 130.85.97.71

150580	130.85.97.81
147491	130.85.18.36
141519	130.85.98.50
131747	130.85.97.231
128445	218.98.64.33
107725	130.85.97.90
103197	130.85.141.21
103146	130.85.70.207
102503	130.85.132.24
95956	130.85.98.69
94608	130.85.97.96
93174	130.85.153.223
92625	130.85.84.178
85230	130.85.97.80
78909	130.85.97.88
76351	130.85.91.252
69421	130.85.98.49
62180	130.85.97.27

Scan by destination IP - external

82174	192.26.92.30
70416	205.231.29.244
57962	205.231.29.243
53864	192.148.252.171
52040	130.94.6.10
35719	192.5.6.30
34739	216.109.116.17
30076	192.52.178.30
29043	192.12.94.30
28182	213.130.63.233
27893	66.33.98.17
26023	209.208.92.254
25269	207.191.192.130
23099	147.91.242.1
22682	208.185.43.73
20756	192.84.159.20
19393	63.87.242.172
19211	131.118.254.33
19132	209.73.205.123
16642	131.118.254.34
16608	205.236.189.10
16199	131.118.254.35
16070	198.102.86.4
14995	209.227.18.5
14209	208.48.34.135

13524	165.230.209.227
13460	128.8.10.90
13364	209.208.0.104
12970	202.96.64.68
12723	128.121.26.10
12694	206.183.198.240
12511	209.208.0.97
12354	206.183.198.241
12058	209.208.0.96

Scan by destination port - external

2958412	53
1715341	137
738163	80
548234	6257
263738	25
184535	17300
184397	22321
174658	445
168629	7674
136862	139
91822	4000
90229	41170
57603	21
49483	8000
49109	8888
46634	8080
46058	6346
42224	3128

4. Top Talkers

Quantity	Source IP	Destination IP	Type
8979	68.49.35.0	MY.NET.30.3:524	Scan?
7709	207.151.67.140	MY.NET.000.000:80	Exploit x86 NOOP
6397	205.160.101.121	MY.NET.83.100:113	Queso fingerprint
4464	193.225.219.29	MY.NET.100.165:21	External ftp traffic
6395	205.160.101.121	MY.NET.83.100	SYN scan
19379	MY.NET.83.100	208.194.163.37:6667	IRC Alert
2526454	130.85.1.3:32814	Various	UDP scan
6971342	130.85.97.000	Various	SYN scan
3770	217.35.117.209	Not Listed	Portscan Detected
3540	169.254.45.176	MY.NET.000.000:137	SMB Name Wildcard

Alerts by source IP

8979 68.49.35.0
7709 207.151.67.140
6397 205.160.101.121
4464 193.225.219.29
3770 217.35.117.209
3540 169.254.45.176

Alerts by relationship - external

6395 205.160.101.121->MY.NET.83.100 syn 12****s* reserved bits

Alerts by relationship / ports - internal

19379 MY.NET.83.100

Alerts by relationship - internal to external

2476 MY.NET.97.20->203.161.233.132

Number of Scans by Type

6971342 UDP scan (Externally-based)
2158978 SYN scan (Externally-based)

Scan by source IP – external

2526454 130.85.1.3

The top talkers list was selected from the lists of alerts and scans above. I tried to choose a few from each list, especially where there are high volumes of alerts or scans. With greater experience in intrusion detection, I'm sure my selection criteria would be different. In actual practice, the very latest exploits that a given network is not prepared for represent the greatest threat. Since I am using logs from sans.org, I do not know what patches have been applied to the system. I thought the external traffic causing alerts was the most important and potentially dangerous traffic, so I gave six of the ten slots to that kind of traffic. The other four slots were based on the relationship lists shown above.

Top Talker 1- Port 524 scan: I couldn't find much information on port 524 scans. Port 524 is used by Netware 5.x file servers and was once used to exploit Linux systems. The source address resolves to a md.comcast.net address. This looks like a denial of service attack in that many packets are sent per second to the same address.

Top Talker 2 – Exploit x86 NOOP: This alert is explained in the top ten alerts previously (number 5/10) as a buffer overflow attack against an Internet Information Server or possibly as a false positive.

Top Talker 3 – Queso: This is also explained in the top 10 alert list (8/10) as a fingerprinting technique. However, like Top Talker 1, you see the same source and destination address and port over and over, as if a poor hacking attempt was made or a router got stuck in a loop.

Top Talker 4 – External FTP traffic: This traffic is coming from an IP address in Hungary (if the address has not been spoofed) every few seconds. Looking in the alert file, there are often port 80 scans from another IP address (202.3.71.26) which directly precede these port 21 scans, suggesting spoofed addresses. Port 21 probes are commonly used to gain FTP services information as a prelude to an attack, such as a Ramen worm.

Top Talker 5 – SYN Scan: Port 113 is the Ident port and is often left open by firewalls. This port is used when a user connects to a mail server or IRC server to find out who is using the service. In this case, the attacker is manipulating TCP options, Sequence numbers and TCP IDs in a reconnaissance attempt.

(From the out of spec files)

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
06/16-18:07:11.584641 205.160.101.121:56979 -> MY.NET.83.100:113
TCP TTL:52 TOS:0x0 ID:61251 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x4B2106C0  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 298510723 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

```
06/16-18:07:11.703390 205.160.101.121:56980 -> MY.NET.83.100:113
TCP TTL:52 TOS:0x0 ID:49702 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x4ABC2BA9  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 298510735 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Top Talker 6 – IRC Alert: The host on the home network is attempting to join an IRC session. Using nslookup, the name returned is Irctoo.net, which is a chat network. There is nothing stealthy about this, as the destination IP address and port are the same each time. The scans are typically minutes apart. The home address of MY.NET.83.100 stays the same, but the ports are constantly changing. It looks like an application on the source computer may be trying to establish an IRC session.

Top Talker 7 – UDP scan to port 53: These records come from the scan logs. Most of the scans are from 130.85.1.3, port 32814. The destination IPs seem to be random, so it does not look like a flood. The rate is as high as 50 per second. It looks like this host is being used by someone else, given the random nature of the destination addresses. DNS queries can provide "information gain" in that they can return ten times as many bytes as are received. The logs do not show information coming back from the targeted servers, but it is possible that a denial of service was being launched against the source host by making it receive back large amounts of DNS information. The source, if not spoofed, is from the network at the University of Maryland, Baltimore County.

Top Talker 8 – UDP scan to port 137: This is very similar to Top Talker 7. Dozens of UDP packets per second are sent from the 138.85 network, this time from ports in the 1024 – 1028 range, to random IP addresses, port 137. In the late 1990s, floods to port 137 could bring down WINS service within seconds. This traffic could be using a spoofed source address in order to flood the (source) network with UDP data. Sending NetBIOS name requests to insecure networks makes all Windows computers with NetBIOS enabled and Unix computers running Samba reply to it. (Vigo)

Top Talker 9 – Portscan detected from 217.35.117.209. These alerts are triggered by too many connections occurring in too short a time frame. Little else can be determined because the alert log in this case provides only a source address.

Top Talker 10 – SMB name wildcard: See Top Ten alerts, number three. The source address cannot be found with nslookup, nor does a Google search yield any information, so the address was most likely spoofed. This is a slow scan – one every minute or so. The destination is to selected subnets in MY.NET, always to port 137.

5. Selected Addresses – Registration Information

I selected addresses based on the potential danger of the activity and frequency. Any address can be spoofed. I looked at incoming attacks only. I assumed random addresses attacking one IP address or a subnet were either spoofed or a distributed attack. I looked for one IP address consistently targeting a small range of addresses. I also gave greater weight to foreign source IPs and sources that showed other attack attempts according to dshield.org.

Source 1: Queso fingerprinting - 212.106.150.180. These attacks originated from the Selesian Institute of Technology in Gliwice, Poland. As seen below, this network has been responsible for other attacks, according to Dsheild.org. Universities are prime locations for hackers because of the concentration of talent and availability of computer facilities. Never trust a student. (Dshield.org)

IP Address:212.106.150.180
HostName:212.106.150.180
DShield Profile:Country:PL
Contact E-mail:gucio@polsl.gliwice.pl
Total Records against IP: 831
Number of targets: 4
Date Range:2003-06-12 to 2003-06-13
Update Summary

Top 10 Ports hit by this source:

Port	Attacks	Start	End
7541	731	2003-06-11	2003-06-13
4662	8	2003-06-09	2003-07-03
6881	8	2003-06-11	2003-06-11
53796	5	2003-06-11	2003-06-11
52153	3	2003-06-11	2003-06-11
5632	2	2003-06-11	2003-06-11
1281	2	2003-06-11	2003-06-11
63296	2	2003-06-11	2003-06-11
54609	2	2003-06-11	2003-06-11
2178	2	2003-06-11	2003-06-11

Last Fightback Sent: not sent
Whois:% This is the RIPE Whois server.

% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

```
inetnum:      212.106.128.0 - 212.106.191.255
netname:      PL-CKPOLSL-981207
descr:        Silesian University of Technology
descr:       PROVIDER LOCAL REGISTRY
country:      PL
admin-c:      SASK1-RIPE
tech-c:       SASK1-RIPE
status:       ALLOCATED PA
notify:       hostmaster@silweb.pl
mnt-by:       RIPE-NCC-HM-MNT
mnt-lower:    PL-CKPOLSL-MNT
mnt-routes:   PL-CKPOLSL-MNT
changed:      hostmaster@ripe.net 19981207
changed:      hostmaster@ripe.net 19981209
changed:      hostmaster@ripe.net 20010307
changed:      lir-help@ripe.net 20011217
changed:      hostmaster@ripe.net 20020222
changed:      hostmaster@ripe.net 20020227
source:       RIPE

route:        212.106.128.0/19
descr:        PL-CKPOLSL-NET-19981207-COM
origin:       AS15744
mnt-by:       AS8508-MNT
changed:      marpros@polsl.gliwice.pl 20001212
source:       RIPE

role:         PL-CKPOLSL Hostmaster Team
address:      Silesian University of Technology
address:      Computer Center
address:      ul. Akademicka 16
address:      44-100 Gliwice
address:      Poland
phone:        +48 32 2307686
fax-no:       +48 32 2372175
e-mail:       hostmaster@silweb.pl
mnt-by:       PL-CKPOLSL-MNT
changed:      hostmaster@silweb.pl 20020124
source:       RIPE
```

Source 2: Possible Red Worm – 217.209.142.239. This traffic originated in Stockholm, Sweden. Dshield.org did not show any other attacks for this source. The fact that these datagrams were all aimed at port 65535 makes this very suspicious. I did find this source address listed on a “Dogs of War” gaming page, so it is possible this could be the result of a multiplayer game, although why it

would use such a high port is unknown. On the other hand, there are probably a lot of hackers who are avid game players. (Dogs of War Sweden)

IP Address: 217.209.142.239
HostName: h239n2fls31o1008.telia.com
DShield Profile: Country: SE
Contact E-mail: mntripe@telia.net
Total Records against IP: not processed
Number of targets: select update below
Date Range: to
Summary was recently updated.

Top 10 Ports hit by this source:
Port Attacks Start End

Last Fightback Sent: not sent
Whois: % This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum: 217.209.0.0 - 217.209.255.255
netname: TELIANET
descr: Telia Network Services
descr: ISP
country: SE
admin-c: TR889-RIPE
tech-c: TR889-RIPE
status: ASSIGNED PA
notify: backbone@telia.net
mnt-by: TELIANET-LIR
changed: fia@telia.net 20011204
changed: aca@telia.net 20020109
source: RIPE

route: 217.208.0.0/13
descr: TELIANET-BLK
origin: AS3301
mnt-by: TELIANET-RR
changed: rr@telia.net 20010508
source: RIPE

role: TeliaNet Registry

address: Telia Network Services
address: Carrier & Networks
address: Box 10707
address: SE-121 29 Stockholm
address: Sweden
nic-hdl: TR889-RIPE
notify: mntripe@telia.net
mnt-by: TELIANET-LIR
changed: fia@telia.net 20020319
changed: eva@telia.net 20020821

Source 3: Null scan – 63.251.52.75. I include this as an example of a false positive. It comes from Shockwave.com, a legitimate commercial site. A user may have been downloading some graphics or games from this site to trigger the IDS. Dshield.org does not show any attacks coming from this site.

IP Address: 63.251.52.75
HostName: www.shockwave.com
DShield Profile: Country: US
Contact E-mail: abuse@internap.com
Total Records against IP: 20
Number of targets: 3
Date Range: 2003-05-20 to 2003-05-20
Summary was recently updated.

Top 10 Ports hit by this source:
Port Attacks Start End

Last Fightback Sent: sent to noc@INTERNAP.COM on 2003-02-17 18:45:49

Whois:
CustName: Shockwave.com
Address: 650 Townsend Street, #450
City: San Francisco
StateProv: CA
PostalCode: 94103
Country: US
RegDate: 2000-08-24
Updated: 2000-08-24

NetRange: 63.251.52.0 - 63.251.52.255
CIDR: 63.251.52.0/24
NetName: PNAP-SFO-SWAVE-RM-01
NetHandle: NET-63-251-52-0-1
Parent: NET-63-251-0-0-1

NetType: Reassigned
Comment:
RegDate: 2000-08-24
Updated: 2000-08-24

ARIN WHOIS database, last updated 2003-04-15 20:10
Enter ? for additional hints on searching ARIN's WHOIS database.

OrgName: Internap Network Services
OrgID: PNAP
Address: 250 Williams Street
Address: Suite E100
City: Atlanta
StateProv: GA
PostalCode: 30303
Country: US

NetRange: 63.251.0.0 - 63.251.255.255
CIDR: 63.251.0.0/16
NetName: NETBLK-PNAP-11-99
NetHandle: NET-63-251-0-0-1
Parent: NET-63-0-0-0-0
NetType: Direct Allocation
NameServer: NS1.PNAP.NET
NameServer: NS2.PNAP.NET
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate: 1999-12-02
Updated: 2002-06-17

TechHandle: INO3-ARIN
TechName: Network Operations Center, InterNap Network
TechPhone: +1-206-256-9500
TechEmail: noc@internap.com

OrgTechHandle: INO3-ARIN
OrgTechName: Network Operations Center, InterNap Network
OrgTechPhone: +1-206-256-9500
OrgTechEmail: noc@internap.com

OrgAbuseHandle: IAC3-ARIN
OrgAbuseName: Internap Abuse Contact
OrgAbusePhone: +1-206-256-9500
OrgAbuseEmail: abuse@internap.com

ARIN WHOIS database, last updated 2003-04-15 20:10

Enter ? for additional hints on searching ARIN's WHOIS database.

OrgName: Internap Network Services
OrgID: PNAP
Address: 250 Williams Street
Address: Suite E100
City: Atlanta
StateProv: GA
PostalCode: 30303
Country: US
Comment:
RegDate: 1996-07-18
Updated: 2003-02-11

AdminHandle: INO3-ARIN
AdminName: Network Operations Center, InterNap Network
AdminPhone: +1-206-256-9500
AdminEmail: noc@internap.com

TechHandle: INO3-ARIN
TechName: Network Operations Center, InterNap Network
TechPhone: +1-206-256-9500
TechEmail: noc@internap.com

AbuseHandle: IAC3-ARIN
AbuseName: Internap Abuse Contact
AbusePhone: +1-206-256-9500
AbuseEmail: abuse@internap.com

Source 4: FTP DoS ftpd globbing from 81.49.150.21. Ftpd is an ftp daemon used in Linux and Unix systems. An attacker can execute code with root privileges if the attack is done correctly. This appears to come from an ISP in France called Wannado.

IP Address: 81.49.150.21
HostName: APoitiers-101-1-5-21.w81-49.abo.wanadoo.fr
DShield Profile: Country:
Contact E-mail:
Total Records against IP: not processed
Number of targets: select update below
Date Range: to
Summary was recently updated.

Top 10 Ports hit by this source:
Port Attacks Start End

Last Fightback Sent: not sent
Whois: % This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum: 81.49.150.0 - 81.49.150.255
netname: IP2000-ADSL-BAS
descr: BSPOI101 Poitiers Bloc2
country: FR
admin-c: WITR1-RIPE
tech-c: WITR1-RIPE
status: ASSIGNED PA
remarks: for hacking, spamming or security problems send mail to
remarks: postmaster@wanadoo.fr AND abuse@wanadoo.fr
mnt-by: FT-BRX
changed: gestionip.ft@francetelecom.com 20021014
changed: gestionip.ft@francetelecom.com 20030318
source: RIPE

route: 81.49.0.0/16
descr: France Telecom
descr: Wanadoo Interactive
remarks: ----------
remarks: For Hacking, Spamming or Security problems
remarks: send mail to abuse@wanadoo.fr
remarks: ----------
origin: AS3215
mnt-by: RAIN-TRANSPAC
changed: karim@rain.fr 20020916
source: RIPE

role: Wanadoo Interactive Technical Role
address: WANADOO INTERACTIVE
address: 48 rue Camille Desmoulins
address: 92791 ISSY LES MOULINEAUX CEDEX 9
address: FR
phone: +33 1 58 88 50 00
e-mail: abuse@wanadoo.fr
e-mail: technical.contact@wanadoo.com
admin-c: WITR1-RIPE
tech-c: WITR1-RIPE
nic-hdl: WITR1-RIPE

mnt-by: FT-BRX
changed: gestionip.ft@francetelecom.com 20010504
changed: gestionip.ft@francetelecom.com 20010912
changed: gestionip.ft@francetelecom.com 20011204
changed: gestionip.ft@francetelecom.com 20030428
source: RIPE

Source 5: Exploit NTPDX buffer overflow – 63.250.195.10. NTP is a time synchronization service and is vulnerable to a buffer overflow attack. If successful, the attacker can gain root privileges and execute commands. Below, the query to Dshield.org shows that this address has been used in other attacks.

IP Address: 63.250.195.10
HostName: l8.cache.vip.dal.yahoo.com
DShield Profile: Country: US
Contact E-mail: netops@broadcast.com
Total Records against IP: 2445
Number of targets: 195
Date Range: 2003-05-20 to 2003-06-28
Update Summary

Top 10 Ports hit by this source:

Port	Attacks	Start	End
1235	52	2003-06-19	2003-07-02
0	24	2003-06-06	2003-06-13
4014	23	2003-06-19	2003-06-19
1776	18	2003-06-20	2003-06-20
3720	15	2003-06-11	2003-06-11
1350	15	2003-06-10	2003-06-11
2382	14	2003-06-20	2003-06-23
1267	14	2003-06-12	2003-06-12
4207	13	2003-06-27	2003-07-02
3533	13	2003-06-11	2003-06-11

Last Fightback Sent: sent to netops@broadcast.com on 2003-06-27 02:06:28

Whois:
OrgName: Yahoo! Broadcast Services, Inc.
OrgID: YAHO
Address: 701 First Avenue
City: Sunnyvale
StateProv: CA
PostalCode: 94089
Country: US

NetRange: 63.250.192.0 - 63.250.223.255
CIDR: 63.250.192.0/19
NetName: NETBLK2-YAHOOBS
NetHandle: NET-63-250-192-0-1
Parent: NET-63-0-0-0-0
NetType: Direct Allocation
NameServer: NS1.YAHOO.COM
NameServer: NS2.YAHOO.COM
NameServer: NS3.YAHOO.COM
NameServer: NS4.YAHOO.COM
NameServer: NS5.YAHOO.COM
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate: 1999-11-24
Updated: 2002-03-27

TechHandle: NA258-ARIN
TechName: Netblock Admin, Netblock
TechPhone: +1-408-349-7183
TechEmail: netblockadmin@yahoo-inc.com

ARIN WHOIS database, last updated 2003-04-20 20:10
Enter ? for additional hints on searching ARIN's WHOIS database.

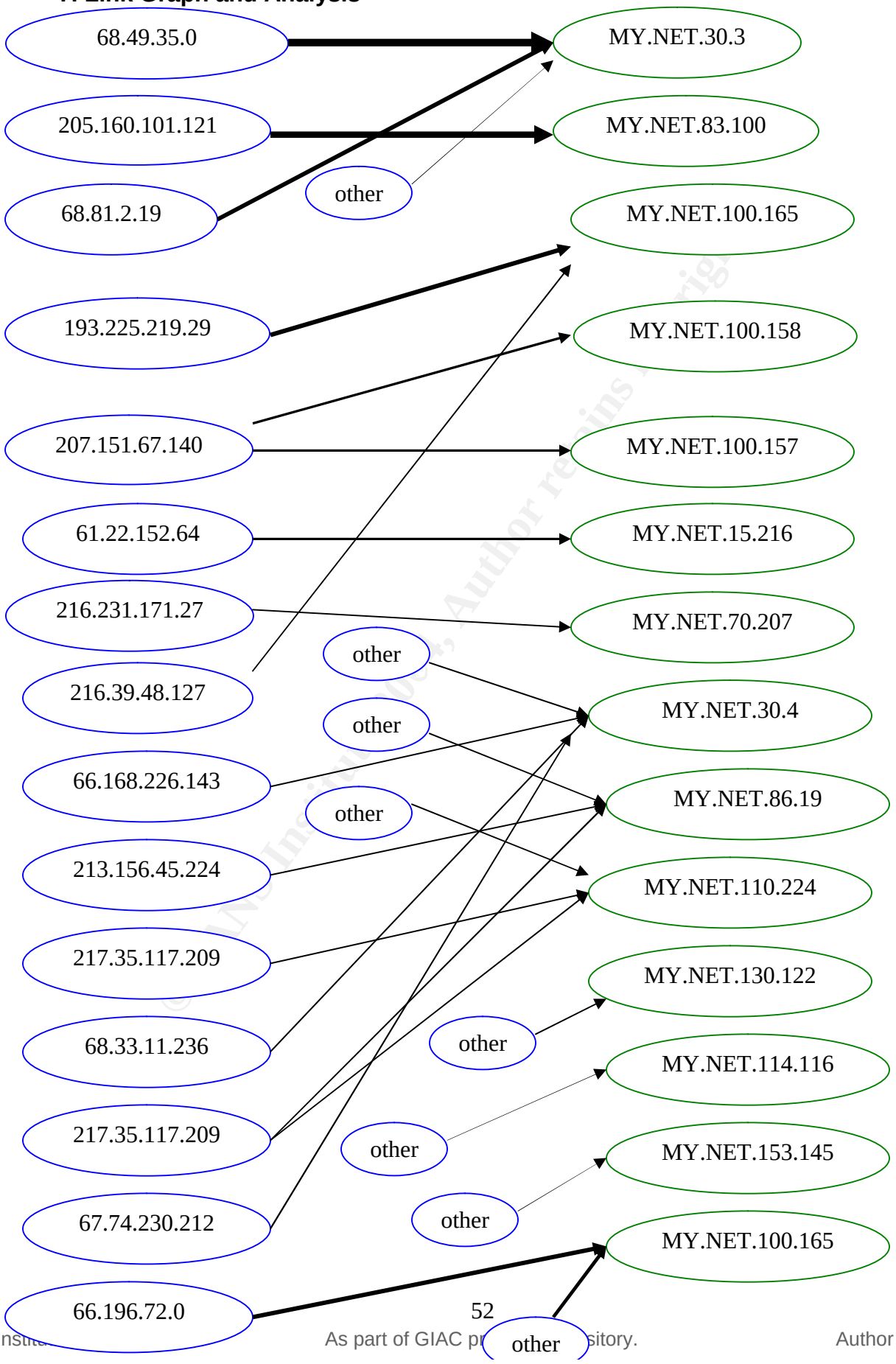
OrgName: American Registry for Internet Numbers
OrgID: ARIN
Address: 3635 Concorde Parkway, Suite 200
City: Chantilly
StateProv: VA
PostalCode: 20151
Country: US

Comment:
RegDate: 1997-04-25
Updated: 2002-08-23

OrgNOCHandle: ARINN-ARIN
OrgNOCName: ARIN NOC
OrgNOCPhone: +1-703-227-9840
OrgNOCEmail: noc@arin.net

OrgTechHandle: IP-FIX-ARIN
OrgTechName: ARIN IP Team
OrgTechPhone: +1-703-227-0660
OrgTechEmail: hostmaster@arin.net

7. Link Graph and Analysis



As the link graph demonstrates, there are concentrations of attackers and victims. Heavy traffic should be investigated because it implies successful hacking attempts, items of interest at a particular address or merely a high volume of unacceptable activity. It may be feasible to block the addresses of the high volume attackers, although the addresses may be spoofed.

9. Defensive Recommendations

1. The snort rules could be tightened up to avoid false positives.
2. Subnets sending large amounts of malicious traffic could be blocked, as long care is taken not to eliminate legitimate traffic.
3. Shut down unessential services and ports, such as FTP, telnet, etc.
4. Use ingress and egress filtering to stop file sharing (Kazaa) and music downloads.
5. Use stateful filtering to help catch insertion attempts and other exploits that require the NIDS to make conclusions about a series of packets.
6. Remove any compromised machines on the network.
7. Make sure security policy is up to date, is known and understood, and have designated people responsible for it.
8. Make sure patches are up to date. The Snort filter caught a lot of Internet Information Server exploits that would harm an unpatched server.
9. Run vulnerability scans on a regular basis. Nessus Security Scanner, NetRecon and ISS Internet Scanner are examples of scanner products.
10. Either allow IRC and file sharing by students, block it with a firewall or IDS, or move it to other servers.
11. There was a lot of outbound scanning activity. Make students aware of policy and discipline inside hackers.
12. Monitor the SANS top 20 list and keep abreast of the latest trends.
13. Make sure servers do not have example and test programs loaded that are vulnerable to hackers.
14. Design automated scripts that perform the types of analysis done in this assignment. If scans, alerts and out-of-spec files were contained in one database, the tables could be joined in meaningful ways. You could have other tables with useful information, such as known bad addresses, the most dangerous traffic, etc. There are analysis tools available, but it is nice to be able to run any type of custom query, as hackers are ingenious and always working to defeat the latest technology.

10. Description of Analysis Process

The five days' worth of files were downloaded to a Linux machine. The 'cat' command was used to join all of the alert, scan and out-of-spec files into three large files. After many attempts to use the 'awk' and 'sed' commands, I concluded my Linux expertise was wanting. Snortsnarf brought my machine to a halt (even after adding more memory) so I could not use it. I located some excellent Perl scripts from a previous GIAC student paper written by Tod

Beardsley. (Beardsley) A script called csv.pl converted the Snort data in to comma delimited files and a script called summarize.pl grouped the data into summary files. These summary files were then used for my analysis.

http://www.giac.org/practical/Tod_Beardsley_GCIA.pdf

References

Spitzner, Lance. “Building a Honeypot.” URL: http://rootprompt.org/article.php3?article=210 (10 May 2003).

Johnson, Keith. “Hackers Caught in Security ‘Honeypot’” Wall Street Journal Online. 4 Oct 2000. URL: http://zdnet.com.com/2100-11-526520.html?legacy=zdnn (12 May 2003).

Tracking-hackers.com. “Honeypots Solutions” URL: http://www.tracking-hackers.com/solutions/ (1 Jul 2003.)

Olsen, Jen. “Hackers caught in security honeypot.” 19 Dec 2000. http://www.linuxsecurity.com/articles/hackscracks_article-2151.html (7 May 2003).

Landergren, Pia. “Hacker Vigilantes Strike Back” IDG News Service 18 Jun 2001. URL: www.security.itworld.com/4362/IDG010618hacks/pfindex.html (20 May 2003).

Windowsecurity.com. “Intrusion Detection Systems FAQ.” URL: www.windowsecurity.com/faqs/Intrusion_Detection/ (12 May 2003).

Miller, Toby. “Intelligence Gathering: Watching a Honeypot at Work.” URL: http://www.securityfocus.com/infocus/1656 (1 Jun 2003).

Socks.permeo.com. “Socks Overview.” http://www.socks.permeo.com/AboutSOCKS/SOCKSOverview.asp (8 Jun 2003

Goto, Shun-ichi. “Make Socket Connection Using Socks4/5 and HTTP Tunnel.” URL: http://www.imasy.or.jp/~gotoh/ssh/connect.c (24 Apr 2003).

Roninsoftwaregroup.com. “Alert Logs.” URL: http://roninsoftwaregroup.com.hosting.domaindirect.com/mysql/alert.ids.txt (12 Jun 2003).

eEye Digital Security. “UPNP – Multiple Remote Windows XP/ME/98 Vulnerabilities.” URL: http://www.eeye.com/html/Research/Advisories/AD20011220.html (19 Jun 2003).

Olavsrud, Thor. “Could Attack on DALnet Kill IRC?” Internetnews. 27 Jan 2003. URL: http://www.isp-planet.com/news/2003/irc_030127.html (25 Jun 2003).

Finchhaven.com. "Two Examples of udp:137 netBIOS name table probes." URL: http://www.finchhaven.com/pages/incidents/030102_udp_137.html (2 Jul 2003).

Mcabee.org. "RE: [Snort-users] spp_http_decode: IIS Unicode attack detected." URL: http://www.mcabee.org/lists/snort-users/Aug-01/msg01261.html (16 May 2003).

Richard, Matthew. "SANS Intrusion Detection Practical v2.7." 27 Feb 2001. URL: http://www.giac.org/practical/matthew_richard_gcia.doc (3 Jun 2003).

Fyodor. "Remote OS detection via TCP/IP Stack Fingerprinting." Phrack Magazine. 18 Oct 1998. URL: http://kaizo.org/mirrors/phrack/phrack54/p54-09 (20 Jun 2003).

Zenomorph. "Fingerprinting Port 80 attacks." URL: http://www.cgisecurity.com/papers/fingerprint-port80.txt (19 Jun 2003).

CIAC. "L-117 The Code Red Worm." 19 Jul 2001. URL: http://www.ciac.org/ciac/bulletins/l-117.shtml (25 Jun 2003).

Vigo, Francesco. "NetBIOS could be used as network flood amplifier." URL: http://www.securityfocus.com/archive/1/317365/2003-03-31/2003-04-06/0 (27 Jun 2003.)

Dshield.org. "IP Info." URL: http://www.dshield.org/ipinfo.php?ip=212.106.150.180 (25 Jun 2003).

Dogs of War Sweden. "Player Profile for TDC." URL: www.dow-swe.com/dow/stats/full/player_Mjk4NDUyMg.php (18 May 2003).

Beardsley, Tod. "Intrusion Detection and Analysis: Theory, Techniques, and Tools." 11 Mar 2002. URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.pdf (2 Jul 2003).