Global Information Assurance Certification Paper

SANS Intrusion Detection in Depth
GCIA Practical Assignment
Version 3.2

Submitted By:
Tim Smoljanovic
September 18, 2002

TABLE OF CONTENTS

1. Assignment 1 – Describe The State Of Intrusion Detection

Intelligence Gathering

1.1 Introduction

History tells us that in many situations, before an adversary attacks, some kind of intelligence or reconnaissance is performed to find and exploit any weakness or vulnerability the target may have. The same holds true for networks. The attacker is looking for holes in the perimeter of our networks, on our servers and on our workstations, so they may find and exploit our vulnerabilities. We must close the holes and be aware of the attacker's methods and what it is he may be trying to accomplish. The intent of this paper is to familiarize the reader with the different intelligence gathering techniques and processes used to gather information on our systems. We will discuss some of the different types of scanning techniques available and different methods used to collect intelligence. Finally, we will discuss the role human psychology plays in information gathering.

There are two common phases used in intelligence gathering; "passive reconnaissance" and "active reconnaissance". Passive reconnaissance is the phase in which the attacker chooses a target. This could stem from a grudge with the company, media attention concerning a recent event, or a previous attack on an organization, and in some cases, for no reason at all. The attacker will try to gain as much information as possible about the company, including who they are, what they do, how big they are, and where they are. This information is quickly and easily obtained in many ways such as the Internet, phone book, Whois search, or advertising. Once the attacker has this information, the passive phase completes and the attacker moves into the active phase and begins to look for vulnerabilities. The attacker will attempt to map the network, find out what services are running, and what operating systems are in use. He will be looking for open ports, un-patched servers, weak passwords, and rogue modems. The avenues of attack are numerous, and just when you think you have closed them all, a new one opens up. The most we can do is keep aware of new attacks and exploits, keep our systems patched and be aware of the tools the attacker is using. Knowledge is power; knowing and understanding the tools of our adversary is a must. Using these tools against our own networks allows us to see what the attacker sees, it gives us insight into our own vulnerabilities, and where we need to tighten our defenses.

1.2 ICMP Refresher

Before we begin, let's briefly review the Internet Control Message Protocol (ICMP) first, so that we may better understand what we are seeing. ICMP is a protocol used for error notification, control messages, and diagnostic functions. ICMP is a connectionless protocol, and although it uses the services of IP, it is not a higher layer protocol, but is, in fact, an integral part of the IP protocol. ICMP will provide us with error notification for events such as; lost datagram fragments, unreachable protocols, services, hosts or networks, network congestion source quenches and many others.

ICMP uses a type and field to identify the type of message being generated. For example, type 8 code 0 would indicate an ICMP echo request. Type 0 code 0 would indicate an ICMP echo reply. More information on ICMP can be found in RFC 792.

1.3 Ping

The ping command makes use of the ICMP echo request. With this ping we can determine if a host on a network is live or not. When we initiate a ping, ICMP will send an echo request (ICMP type 8 code 0), to the remote destination. In return the remote station will send back an echo reply (ICMP type 0 code 0) for every request sent. Figure 1.0 shows a simple ICMP echo reply to a ping sending 4 echo requests. This ping was generated by a Linux workstation running Red Hat 7.2.

```
PING Remote.Host.56.29 (Remote.Host.56.29) 56 bytes of data.
64 bytes from Remote.Host.56.29: icmp_seq=0 ttl=116 time=43.524 msec
64 bytes from Remote.Host.56.29: icmp_seq=1 ttl=116 time=39.952 msec
64 bytes from Remote.Host.56.29: icmp_seq=2 ttl=116 time=69.965 msec
64 bytes from Remote.Host.56.29: icmp_seq=3 ttl=116 time=49.954 msec

--- Remote.Host.56.29 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/mdev = 39.952/50.848/69.965/11.607 ms
```

Figure 1.0

Using the ping command, we have now determined that Remote.Host.56.29 is live. We have also determined that the host is 12 hops away. This is determined by the TTL value (time to live). ICMP will set the TTL to the default value (which is determined by the OS), in this case, the echo reply was issued from a Windows 2000 workstation and, therefore, the initial TTL was set for 128. As the echo reply passes through the routers, the TTL is decremented by 1. If the TTL reaches 0, the packet is discarded. This way, lost packets will eventually be terminated. The TTL was decremented 12 times before reaching us, thereby giving us a TTL of 116. To determine the TTL of your own OS, ping the local loop back address (127.0.0.1). This will return the default TTL of your system. The following is a list of some default TTL values for some operating systems;

OS	TTL Value
Windows 2000	128
Windows NT 4.0	128
Cisco IOS 12.x	255
OpenBSD 3.1	255
Linux RedHat 7.2	255

When a packet is sent to a broadcast address, all live hosts on the broadcast domain[1] will receive and process the packet. If a response is required from the destination host a reply will be issued. Not all live hosts on the network will reply to the echo requests sent to a broadcast address. Some operating systems will deny a response to a broadcast ping thus giving the attacker an in-complete list of all live hosts on that

[1] The broadcast domain is the portion of a network that receives all broadcasts. With the use of subnetting and Virtual Lans (RFC 3069), the entire network may not be on the same broadcast domain (RFC 950).

network. What the attacker will gain is a partial list of live hosts and a good indication of their possible operating systems. A well-known attack using the ping on a broadcast address is the Smurf attack. The attacker will ping a broadcast address (possibly more than one if he is able) and direct the replies to a specific host. Enough traffic could be generated to bog down the host or network, causing a denial of service. Figure 1.1 shows a ping to the broadcast address of a network.

```
PING My.Remote.Host.255 (My.Remote.Host.255): 56 data bytes
64 bytes from My.Remote.Host.159: icmp_seq=0 ttl=255 time=0.836 ms
64 bytes from My.Remote.Host.253: icmp_seq=0 ttl=255 time=1.371 ms (DUP!)
64 bytes from My.Remote.Host.252: icmp_seq=0 ttl=255 time=2.699 ms (DUP!)
64 bytes from My.Remote.Host.216: icmp_seq=0 ttl=64 time=4.489 ms (DUP!)
64 bytes from My.Remote.Host.214: icmp_seq=0 ttl=64 time=4.877 ms (DUP!)
64 bytes from My.Remote.Host.237: icmp_seq=0 ttl=64 time=5.363 ms (DUP!)
64 bytes from My.Remote.Host.246: icmp_seq=0 ttl=255 time=5.751 ms (DUP!)
64 bytes from My.Remote.Host.247: icmp_seq=0 ttl=255 time=7.701 ms (DUP!)
64 bytes from My.Remote.Host.78: icmp_seq=0 ttl=60 time=8.269 ms (DUP!)
64 bytes from My.Remote.Host.56: icmp_seq=0 ttl=60 time=8.645 ms (DUP!)
64 bytes from My.Remote.Host.234: icmp_seq=0 ttl=60 time=9.019 ms (DUP!)
64 bytes from My.Remote.Host.225: icmp_seq=0 ttl=60 time=10.846 ms (DUP!)
64 bytes from My.Remote.Host.229: icmp_seq=0 ttl=60 time=11.237 ms (DUP!)
64 bytes from My.Remote.Host.230: icmp_seq=0 ttl=60 time=11.621 ms (DUP!)
64 bytes from My.Remote.Host.228: icmp_seq=0 ttl=60 time=12.005 ms (DUP!)
64 bytes from My.Remote.Host.232: icmp_seq=0 ttl=60 time=13.832 ms (DUP!)
64 bytes from My.Remote.Host.233: icmp_seq=0 ttl=60 time=14.362 ms (DUP!)
64 bytes from My.Remote.Host.218: icmp_seq=0 ttl=60 time=14.745 ms (DUP!)
64 bytes from My.Remote.Host.238: icmp_seq=0 ttl=60 time=15.127 ms (DUP!)
64 bytes from My.Remote.Host.235: icmp_seq=0 ttl=60 time=18.421 ms (DUP!)
64 bytes from My.Remote.Host.220: icmp_seq=0 ttl=60 time=20.257 ms (DUP!)
64 bytes from My.Remote.Host.32: icmp_seq=0 ttl=60 time=20.791 ms (DUP!)
64 bytes from My.Remote.Host.245: icmp_seq=0 ttl=255 time=21.172 ms (DUP!)
64 bytes from My.Remote.Host.8: icmp_seq=0 ttl=60 time=23.384 ms (DUP!
64 bytes from My.Remote.Host.236: icmp_seq=0 ttl=60 time=23.877 ms (DUP!)
64 bytes from My.Remote.Host.227: icmp_seq=0 ttl=60 time=24.271 ms (DUP!)
64 bytes from My.Remote.Host.222: icmp_seq=0 ttl=60 time=24.656 ms (DUP!)
64 bytes from My.Remote.Host.226: icmp_seq=0 ttl=60 time=25.376 ms (DUP!)
64 bytes from My.Remote.Host.231: icmp_seq=0 ttl=64 time=25.620 ms (DUP!)
64 bytes from My.Remote.Host.249: icmp_seq=0 ttl=255 time=25.846 ms (DUP!)
64 bytes from My.Remote.Host.159: icmp_seq=1 ttl=255 time=0.470 ms
--- My.Remote.Host.255 ping statistics ---
2 packets transmitted, 2 packets received, 33 duplicates, 0% packet loss
round-trip min/avg/max/std-dev = 0.470/14.041/25.846/7.890 ms
```

Figure 1.1

There are a few points to note here. The TTL changes with different systems. As mentioned earlier, this is because each OS sets its own default TTL. So, at this point, not only have we determined a portion of hosts that are live on the network, but we also have a good idea of what operating systems are being used. Knowing what hosts respond to a broadcast echo request and the default TTL they use will help aid the attacker in directing his attack to a specific operating system exploit. The “DUP!” indicates that there are duplicate replies from that host. Since we can specify the number of echo requests to send, and the size of the packet, you can see how ping can be a very useful and nefarious tool. We are also able to resolve the IP address

from a host name or URL. If we did a ping to the URL www.foobar.com, it would return the IP address of the system hosting that URL. The following is a list of some operating systems and their response to a broadcast ping

Operating System	Response Sent
Windows NT 4.0 SP6	No
Windows 2000	No
Windows XP	No
Linux RedHat 7.2	Yes
Cisco IOS 12.x	Yes
OBSD 3.1	No

More information regarding ICMP and it's use in scanning can be found at http://www.sys-security.com/html/projects/icmp.html

1.4 Advanced Scanning

The use of scanning for intelligence gathering has steadily become more advanced. Methods are being developed that will allow stealth scanning of networks, with the intention of evading intrusion detection systems and firewalls. These types of scans are intended to probe your network, find your weakness and avoid detection. In this section we will focus on techniques that attempt to accomplish these goals. We will look at scans that attempt to evade the intrusion detection systems and firewalls. We will look at how some scans will probe and map your network, tell us what operating system is running, what ports are open, and detect other vulnerabilities. There are many programs available on the Internet free for downloading that will give an attacker the ability to perform this type of activity. A powerful and feature-rich scanner may be the ultimate choice for one individual, but because the program is command-line based and complicated to run, may make it an unwanted choice for another. For the purpose of this paper we will be using Nmap which is available from Insecure.org to generate the scans. We will also take a brief look at GFI Languard Network Scanner so we may compare the results of two different scanners. Before we get started, let's review a couple of key concepts of the Transport Control Protocol (TCP) as a refresher.

1.5 TCP Refresher

To get a better understanding of what is transpiring with the different styles of scans, a basic understanding of key TCP concepts is needed. TCP is a connection-oriented protocol. This means that the packets TCP transmits are acknowledged, or they must be retransmitted. TCP relies on the Internet Protocol (IP) to transport the packets to the receiving station, yet IP is not connection-oriented and is not concerned with reliable delivery. IP will leave TCP to figure out if a packet did not make it to its final destination. The connectionless counterpart to TCP is the User Datagram Protocol (UDP). UDP is also a transport layer protocol, but is not concerned with the

retransmission of packets in the event of an error. TCP and UDP make use of ports to determine what application service a packet should be processed by. The ports tell TCP and UDP the end destination of a packet. You may find a list of the common port assignments for both UDP and TCP at http://www.iana.org/assignments/port-numbers.

When we want to make a TCP connection, say to an HTTP web server (port 80), TCP must complete a three-way handshake to establish a session. We would define a session as the time from when you connect to the web site; to the time you leave that web site. This is where we start to see the TCP flags. There are 6 flags in total, urgent (URG), acknowledge (ACK), push (PSH), reset (RST), synchronize (SYN), and finish (FIN). RFC 793 will describe the use of each of these flags in more detail. The three-way handshake must complete for a session to take place. Figure 1.2 shows a three-way handshake taking place.

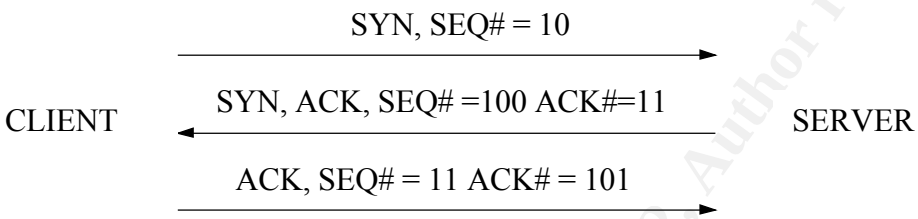

Figure 1.2

Let's take a closer look at what is happening. The client wants to make a connection to the server. The client will first send a packet with the SYN flag set, and a sequence (SEQ) number also set. The server will then issue a packet, also with the SYN flag set to say, "Yes, let's establish a session". The server will set his initial sequence number and also set the ACK flag, to acknowledge the initial packet from the client. The Server also sets the ACK number to what is expected to be the next packet's sequence number he will receive from the client. At this point the client responds with an ACK and sets the next sequence number. The client also sets the ACK# to what is expected as the sequence number of the next packet to be received from the server. At this point, a session is established and they may now begin to exchange data.

1.6 Nmap

Nmap is an open source stealth scanner available for download from http://www.insecure.org/nmap/index.html. It is one of the more widely-used scanners and is available for Linux, OpenBSD, FreeBSD, Win32 and others. Nmap has a variety of possible techniques including, but not limited to, TCP Connect(), TCP SYN, Xmas tree, stealth FIN, UDP, and ACK scanning. Nmap also allows for Remote OS Fingerprinting, FTP bounce, and Zombie scanning. Other types of techniques and an explanation as to what each one does, can be found in the Nmap manual pages located at http://www.insecure.org/nmap/nmap_manpage.html. The intent of this paper is to review intelligence gathering techniques. Nmap is simply the tool used to produce the scans we will be reviewing. For the purpose of this paper, we will use Nmap version 2.54BETA36, running on Red Hat Linux version 7.2.

Although Nmap is capable of scanning an entire network, we will only be looking at the results from a single host for clarity. Below the output of each scan, I have included the port scan alerts generated from a Snort version 1.8.6 Intrusion Detection System so that we may get a better understanding of what is happening.

a. TCP Connect() scan

This first scan is a TCP Connect() scan. TCP Connect will attempt to complete the three-way handshake and open a session with the probed port. If a successful connection to the port is made, we can assume it is open. If no connection is made, the port is closed. Although the TCP Connect() scan is fast, it is easily detected. System logs would show a bunch of connections and the errors produced, resulting from immediately shutting down the connections. The –sT switch is used to perform this scan.

```
Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Remote.Host.56.28 (XXX.XXX.56.28):
(The 1551 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
80/tcp     open        http
111/tcp    open        sunrpc
1024/tcp   open        kdm
6346/tcp   filtered    gnutella
6699/tcp   filtered    napster
8888/tcp   filtered    sun-answerbook

Nmap run completed -- 1 IP address (1 host up) scanned in 2 seconds
```

Snort Detect

```
Jul 14 09:16:40 Local.Host.201.159:3306 -> Remote.Host.56.28:451 SYN ******S*
Jul 14 09:16:40 Local.Host.201.159:3307 -> Remote.Host.56.28:700 SYN ******S*
Jul 14 09:16:40 Local.Host.201.159:3308 -> Remote.Host.56.28:1522 SYN ******S*
Jul 14 09:16:40 Local.Host.201.159:3309 -> Remote.Host.56.28:406 SYN ******S*
Jul 14 09:16:40 Local.Host.201.159:3310 -> Remote.Host.56.28:1515 SYN ******S*
.
.
Jul 14 09:16:41 Local.Host.201.159:4858 -> Remote.Host.56.28:162 SYN ******S*
Jul 14 09:16:41 Local.Host.201.159:4859 -> Remote.Host.56.28:290 SYN ******S*
Jul 14 09:16:41 Local.Host.201.159:4861 -> Remote.Host.56.28:2028 SYN ******S*
Jul 14 09:16:41 Local.Host.201.159:4862 -> Remote.Host.56.28:2501 SYN ******S*
Jul 14 09:16:41 Local.Host.201.159:4863 -> Remote.Host.56.28:674 SYN *******S*
Total = 1542
```

Figure 1.3

Let's examine the results. The first capture is the Nmap scan, and it tells us a total of 1,558 ports were scanned (this is derived by adding the total detected ports to the reported undetected ports). Of those 1,558 ports it found 4 ports open and 3 ports filtered. In this case, an access control list on the router has filtered the three ports. Looking at the Snort detection, we notice a couple of things. First, we know this is an automated scan by the time it took to complete the scan. Most port scans will be automated so this should not come as a big surprise. We also notice that the source

ports are listed sequentially, yet the destination ports are random. This is an attempt to confuse some intrusion detection systems. Last, you may notice that Snort only detected 1,542 scans. In this case, it is caused by the hardware snort is running on. It's an older computer and does not have the power to keep up with the speed of the scans and therefore drops some of the packets.

b. Xmas Tree

This next scan is called a stealth Xmas Tree scan. The purpose of the Xmas Tree scan is to attempt to elude intrusion detection systems. A stealth scanner works on the premise that the system will record multiple TCP Connect()s with an immediate tear down. The stealth Xmas Tree scan will set the URG, PSH, and FIN flags, thus hopefully avoiding detection by receiving a RST from closed ports, and therefore never completing the connection. Any open ports should simply drop the packet. This, of course, is how it should be done according to RFC 793. Some operating systems do not follow this standard, including Microsoft Windows, and therefore will return RST even for an open port. The setting of the three flags is where this scan gets its name. It lights up all three flags like a Xmas tree. This type of scan would be of no use when scanning a Windows system, as we will not be able to detect any open ports. On the other hand we now have a crude method of OS Fingerprinting. To perform the Xmas Tree scan we use the –sX switch.

```
Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Remot.Host.56.28(xxx.xxx.56.28):
(The 1551 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
80/tcp     open        http
111/tcp    open        sunrpc
1024/tcp   open        kdm
6346/tcp   filtered    gnutella
6699/tcp   filtered    napster
8888/tcp   filtered    sun-answerbook
Nmap run completed -- 1 IP address (1 host up) scanned in 5 seconds
```

Snort Detect

```
Jul 14 09:27:46 Local.Host.201.159:49108 -> Remote.Host.56.28:870 XMAS **U*P**F
Jul 14 09:27:46 Local.Host.201.159:49108 -> Remote.Host.56.28:373 XMAS **U*P**F
Jul 14 09:27:46 Local.Host.201.159:49108 -> Remote.Host.56.28:608 XMAS **U*P**F
Jul 14 09:27:46 Local.Host.201.159:49108 -> Remote.Host.56.28:394 XMAS **U*P**F
Jul 14 09:27:46 Local.Host.201.159:49108 -> Remote.Host.56.28:338 XMAS **U*P**F
.
.
Jul 14 09:27:48 Local.Host.201.159:49108 -> Remote.Host.56.28:929 XMAS **U*P**F
Jul 14 09:27:48 Local.Host.201.159:49108 -> Remote.Host.56.28:7008 XMAS **U*P**F
Jul 14 09:27:49 Local.Host.201.159:49108 -> Remote.Host.56.28:1432 XMAS **U*P**F
Jul 14 09:27:49 Local.Host.201.159:49108 -> Remote.Host.56.28:65 XMAS **U*P**F
Jul 14 09:27:49 Local.Host.201.159:49108 -> Remote.Host.56.28:553 XMAS **U*P**F
Total = 1038
```

Figure 1.4

I mentioned that some operating systems will not respond correctly to this type of scan,

yet we are showing open ports. So, we now know we are not scanning a Windows system. The Xmas Tree scan returned the exact same thing as the TCP Connect() scan. We could still find a benefit in this type of scan; if we were scanning the entire network, we would be able to determine running Windows systems. When we look at the Snort detects, we do see a bit of a pattern change. Again, the time gives away the fact that this is automated, but notice now that the source ports do not change. Our trace does not show it, but they will change. Very seldom they will increase by one for about 2 packets, then return to their original number. Our destination ports are still random and we are showing the URG, PSH, and FIN bits set, giving away that this is a Xmas Tree scan.

c. SYN scan

The next scan is called the SYN scan, also referred to as the "half-open" scan. We know the TCP Connect() scan will complete the three-way handshake, thus making a complete connection to the port. The premise of the SYN scan is to send only the SYN packet, the first part of a three-way handshake. If we then receive a SYN/ACK back from the remote host, we know that port is open. If we receive a RST, we know it's closed. If we do receive the SYN/ACK, we then send a RST to tear down and terminate the connection. The purpose of this is that some intrusion detection systems know that a SYN packet is legitimate traffic and, therefore, will not log it. The switch used for the Syn scan is –sS.

```
Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Remote.Host.56.28 (xxx.xxx.56.28):
(The 4995 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
80/tcp     open        http
111/tcp    open        sunrpc
1024/tcp   open        kdm
1214/tcp   filtered    unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 4 seconds
```

Snort Detect

```
Jul 14 09:24:29 Local.Host.201.159:51553 -> Remote.Host.56.28:2565 SYN ******S*
Jul 14 09:24:29 Local.Host.201.159:51553 -> Remote.Host.56.28:1475 SYN ******S*
Jul 14 09:24:29 Local.Host.201.159:51553 -> Remote.Host.56.28:3867 SYN ******S*
Jul 14 09:24:29 Local.Host.201.159:51553 -> Remote.Host.56.28:3925 SYN ******S*
Jul 14 09:24:29 Local.Host.201.159:51553 -> Remote.Host.56.28:4545 SYN ******S*
.
.
Jul 14 09:24:33 Local.Host.201.159:51553 -> Remote.Host.56.28:3195 SYN ******S*
Jul 14 09:24:33 Local.Host.201.159:51553 -> Remote.Host.56.28:4925 SYN ******S*
Jul 14 09:24:33 Local.Host.201.159:51553 -> Remote.Host.56.28:3150 SYN ******S*
Jul 14 09:24:33 Local.Host.201.159:51553 -> Remote.Host.56.28:1990 SYN ******S*
Jul 14 09:24:33 Local.Host.201.159:51553 -> Remote.Host.56.28:2477 SYN ******S*
Total = 3084
```

Figure 1.5

We are now only showing one filtered port and it is not one of the original filtered ports. Using the –p option allows us to specify what ports we wanted scanned, as opposed to using the Nmap defaults. In this case, we specified ports 1 through 5,000. Keep in mind that there are over 65,500 ports available, so being able to specify what ports you want scanned is a helpful feature. Looking at the snort logs we notice things have not changed much from the Xmas Tree scan. We still have our source port remaining constant and our destination port is still random. The only real difference we see, is we only have the SYN flag set. We were able to detect this scan because some detection systems will watch for a large number of SYN packets destined for many ports in a very short period of time.

d. Frag scan

This next scan is the frag scan. The purpose here is to avoid detection by hiding our intentions in fragments. When a packet exceeds the Maximum Transmission Unit (MTU) of a network, the router will break the packet up into fragments. Once the fragments reach their destination, the receiving host will rebuild the packet and send back a reply. This scan was done in conjunction with a Xmas scan. This scan used the –f to perform the frag scan and the –sX for the Xmas scan.

```
Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Host   (Remote.Host.61.0) seems to be a subnet broadcast address (returned
1 extra pings). Skipping host.
Interesting ports on Remote.Host.61.2 (xxx.xxx.61.2):
Port       State       Service
1/tcp      open        tcpmux
2/tcp      open        compressnet
3/tcp      open        compressnet
4/tcp      open        unknown
5/tcp      open        rje
6/tcp      open        unknown
7/tcp      open        echo
8/tcp      open        unknown
9/tcp      open        discard
10/tcp     open        unknown
11/tcp     open        systat
12/tcp     open        unknown
13/tcp     open        daytime
14/tcp     open        unknown
15/tcp     open        netstat
16/tcp     open        unknown
17/tcp     open        qotd
```

Snort Detect

```
Aug  6 22:09:21 My.Host.201.159:36427 -> Remote.Host.61.2:1026 XMAS
**U*P**F
Aug  6 22:09:21 My.Host.201.159:36427 -> Remote.Host.61.2:835 XMAS **U*P**F
Aug  6 22:09:21 My.Host.201.159:36427 -> Remote.Host.61.2:705 XMAS **U*P**F
Aug  6 22:09:22 My.Host.201.159:36428 -> Remote.Host.61.2:752 XMAS **U*P**F
Aug  6 22:09:22 My.Host.201.159:36428 -> Remote.Host.61.2:470 XMAS **U*P**F
```

```
Aug  6 22:09:22 My.Host.201.159:36428 -> Remote.Host.61.2:100 XMAS **U*P**F
.
.
Aug  6 22:09:29 My.Host.201.159:36427 -> Remote.Host.61.2:840 XMAS **U*P**F
Aug  6 22:09:29 My.Host.201.159:36427 -> Remote.Host.61.2:300 XMAS **U*P**F
Aug  6 22:09:29 My.Host.201.159:36427 -> Remote.Host.61.2:916 XMAS **U*P**F
Aug  6 22:09:30 My.Host.201.159:36428 -> Remote.Host.61.2:840 XMAS **U*P**F
Aug  6 22:09:30 My.Host.201.159:36428 -> Remote.Host.61.2:300 XMAS **U*P**F
Total = 1723
```

Snort Alert

```
[**] [1:522:1] MISC Tiny Fragments [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
08/06-22:09:30.247167 My.Host.201.159 -> Remote.Host.61.2
TCP TTL:59 TOS:0x0 ID:37687 IpLen:20 DgmLen:36 MF
Frag Offset: 0x0000   Frag Size: 0x0010
```

Figure 1.6

We have a bit more information this time. This scan resulted in all ports being scanned. I have only included a small portion of the response, due to it's size. We will notice with the Snort scan log, we see a typical Xmas scan taking place. We also see the changing pattern in the source ports with this trace. The pattern is a little different than the original Xmas scan in that the ports change more frequently, and for a longer period of time, but maintain the same two port numbers throughout the scan. Below the port scan, we see the alert snort has displayed in the alert log to inform us that tiny fragment activity is taking place.

e. OS Fingerprint

For this next scan, we are again going to do the SYN scan, but we will also attempt some basic operating system fingerprinting. Not all operating system vendors implement TCP/IP into their operating systems the same way. Each operating system will respond differently to illegal flag settings in the TCP header. Some operating systems generate easy to predict sequence and ID numbers. The way in which the systems respond to these characteristics allows us to determine what operating system is running. We will also do a TCP sequence number prediction, telling us how hard it would be to forge a TCP session against the remote system. This scan uses the –O to do the OS fingerprint and sequence number prediction, and the –sS for the Syn scan.

```
Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Host Remote.Host.56.28 (xxx.xxx.56.28) appears to be up ... good.
Initiating SYN Stealth Scan against Remote.Host.56.28 (xxx.xxx.56.28)
Adding open port 22/tcp
Adding open port 111/tcp
Adding open port 1024/tcp
Adding open port 80/tcp
The SYN Stealth Scan took 2 seconds to scan 1558 ports.
For OSScan assuming that port 22 is open and port 1 is closed and neither
are firewalled
Interesting ports on Remote.Host.56.28 (xxx.xxx.56.28):
(The 1551 ports scanned but not shown below are in state: closed)
```

```
Port       State       Service
22/tcp     open        ssh
80/tcp     open        http
111/tcp    open        sunrpc
1024/tcp   open        kdm
6346/tcp   filtered    gnutella
6699/tcp   filtered    napster
8888/tcp   filtered    sun-answerbook
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 5.827 days (since Mon Jul  8 14:33:30 2002)
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=1948084 (Good luck!)
IPID Sequence Generation: All zeros

Nmap run completed -- 1 IP address (1 host up) scanned in 7 seconds
```

Snort Alert,

```
Jul 14 10:20:56 Local.Host.201.159:33179 -> Remote.Host.56.28:995 SYN ******S*
Jul 14 10:20:56 Local.Host.201.159:33179 -> Remote.Host.56.28:2431 SYN ******S*
Jul 14 10:20:56 Local.Host.201.159:33179 -> Remote.Host.56.28:5977 SYN ******S*
Jul 14 10:20:56 Local.Host.201.159:33179 -> Remote.Host.56.28:273 SYN ******S*
Jul 14 10:20:56 Local.Host.201.159:33179 -> Remote.Host.56.28:1429 SYN ******S*
.
.
Jul 14 10:20:58 Local.Host.201.159:33179 -> Remote.Host.56.28:531 SYN ******S*
Jul 14 10:21:00 Local.Host.201.159:33187 -> Remote.Host.56.28:22 NULL ********
Jul 14 10:20:58 Local.Host.201.159:33188 -> Remote.Host.56.28:22 NMAPID **U*P*SF
Jul 14 10:20:58 Local.Host.201.159:33192 -> Remote.Host.56.28:1 XMAS **U*P**F
Jul 14 10:21:02 Local.Host.201.159:33185 -> Remote.Host.56.28:22 SYN ******S*
Total = 1559
```

Figure 1.7

This scan has yielded some interesting information. Nmap has made a guess that the operating system is Linux. In fact, it is Red Hat Linux 7.2. Also, we are aware of the last time this system was re-booted. The TCP sequence prediction is telling us that predicting the TCP sequence on this host would be rather difficult. Look at the alert generated by Snort in the beginning, we see basic SYN scan. The characteristics are the same until we get closer to the end. We notice that we have a Xmas Tree detection and also a packet with the URG, PSH, SYN, and FIN bits set. This is a part of the process used to determine the OS running on the remote host.

f. Decoy Scan

When the attacker gathers intelligence against a network, a response is required or the attacker would not gain any information. This would make it fairly certain the attacker is not spoofing his IP address. Using a Decoy scan, the attacker is able to conceal his address among addresses he is spoofing. This type of scan would create confusion as to where the actual scan came from, making it much harder for the analyst to track the attacker. To perform this scan we use the –D <decoy1,decoy2> switch.

```
Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Remote.Host.56.28 (xxx.xxx.56.28):
(The 1551 ports scanned but not shown below are in state: closed)
```

```
Port       State       Service
22/tcp     open        ssh
80/tcp     open        http
111/tcp    open        sunrpc
1024/tcp   open        kdm
6346/tcp   filtered    gnutella
6699/tcp   filtered    napster
8888/tcp   filtered    sun-answerbook
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 5.815 days (since Mon Jul  8 14:33:30 2002)

Nmap run completed -- 1 IP address (1 host up) scanned in 28 seconds
```

Snort Alert,

```
Jul 14 10:03:20 Decoy.Host.201.122:48147 -> Remote.Host.56.28:654 SYN ******S*
Jul 14 10:03:20 Decoy.Host.201.122:48147 -> Remote.Host.56.28:137 SYN ******S*
Jul 14 10:03:20 Local.Host.201.159:48147 -> Remote.Host.56.28:654 SYN ******S*
Jul 14 10:03:20 Local.Host.201.159:48147 -> Remote.Host.56.28:137 SYN ******S*
Jul 14 10:03:20 Decoy.Host.200.6:48147 -> Remote.Host.56.28:654 SYN ******S*
Jul 14 10:03:20 Decoy.Host.200.6:48147 -> Remote.Host.56.28:137 SYN ******S*
Jul 14 10:03:20 Decoy.Host.56.29:48147 -> Remote.Host.56.28:654 SYN ******S*
Jul 14 10:03:20 Decoy.Host.56.29:48147 -> Remote.Host.56.28:137 SYN ******S*
Jul 14 10:03:20 Decoy.Host.61.122:48147 -> Remote.Host.56.28:254 SYN ******S*
Jul 14 10:03:20 Decoy.Host.61.122:48147 -> Remote.Host.56.28:1013 SYN ******S*
Jul 14 10:03:38 Decoy.Host.61.122:48147 -> Remote.Host.56.28:356 SYN ******S*
Jul 14 10:03:38 Decoy.Host.61.122:48147 -> Remote.Host.56.28:909 SYN ******S*
Jul 14 10:03:21 Decoy.Host.207.210:48147 -> Remote.Host.56.28:154 SYN ******S*
Jul 14 10:03:21 Decoy.Host.207.210:48147 -> Remote.Host.56.28:1475 SYN ******S*
.
.
Jul 14 10:03:44 Decoy.Host.201.122:48155 -> Remote.Host.56.28:22 NULL ********
Jul 14 10:03:42 Decoy.Host.201.122:48156 -> Remote.Host.56.28:22 NMAPID **U*P*SF
Jul 14 10:03:40 Decoy.Host.201.122:48160 -> Remote.Host.56.28:1 XMAS **U*P**F
Jul 14 10:03:42 Local.Host.201.159:48155 -> Remote.Host.56.28:22 NULL ********
Jul 14 10:03:42 Local.Host.201.159:48156 -> Remote.Host.56.28:22 NMAPID **U*P*SF
Jul 14 10:03:40 Local.Host.201.159:48160 -> Remote.Host.56.28:1 XMAS **U*P**F
Jul 14 10:03:42 Decoy.Host.200.6:48155 -> Remote.Host.56.28:22 NULL ********
Jul 14 10:03:42 Decoy.Host.200.6:48156 -> Remote.Host.56.28:22 NMAPID **U*P*SF
Jul 14 10:03:40 Decoy.Host.200.6:48160 -> Remote.Host.56.28:1 XMAS **U*P**F
Jul 14 10:03:42 Decoy.Host.56.29:48155 -> Remote.Host.56.28:22 NULL ********
Jul 14 10:03:42 Decoy.Host.56.29:48156 -> Remote.Host.56.28:22 NMAPID **U*P*SF
Jul 14 10:03:40 Decoy.Host.56.29:48160 -> Remote.Host.56.28:1 XMAS **U*P**F
Jul 14 10:03:40 Decoy.Host.61.122:48154 -> Remote.Host.56.28:22 SYN *2****S*
Jul 14 10:03:42 Decoy.Host.61.122:48155 -> Remote.Host.56.28:22 NULL ********
Jul 14 10:03:42 Decoy.Host.61.122:48156 -> Remote.Host.56.28:22 NMAPID **U*P*SF
Jul 14 10:03:40 Decoy.Host.61.122:48160 -> Remote.Host.56.28:1 XMAS **U*P**F
Jul 14 10:03:42 Decoy.Host.207.210:48155 -> Remote.Host.56.28:22 NULL ********
Jul 14 10:03:42 Decoy.Host.207.210:48156 -> Remote.Host.56.28:22 NMAPID **U*P*SF
Jul 14 10:03:40 Decoy.Host.207.210:48160 -> Remote.Host.56.28:1 XMAS **U*P**F
```

Figure 1.8

Again we are showing nothing new here in terms of packets from a single host. We have obtained the results we would expect from the SYN scan and OS fingerprinting. The real benefit here is what the Intrusion Analyst is going to see. Take a look at what Snort has picked up. As you can see, it would be pretty difficult for the analyst to

determine where the scan came from. The analyst would be able to determine that most likely all the scans are related, simply by close examination of the captures. Notice the time; all the scans are within less than a second from each other. We see in the first part of the scan that the source port remains constant through most of the scan. Not shown here, but very seldom and randomly throughout the scan, the source port will change to 48148 for one or two packets, then back to 48147. The destination ports are again random, but the sequence in which they are scanned remains the same for each address. The Last thing we see is the OS fingerprint taking place at the end of the scan from all address, again less than a second apart.

g. Zombie Scan

Our previous scan made it very difficult for the Analyst to discover who was doing the scan. The only problem here is that although he is well hidden, our attacker's IP is still among the scans. If the Intrusion Analyst decided that the spoofed addresses were of no value to his organization, and if he were able to block all of them, our attacker would also be blocked. The attacker needs to come up with some way of scanning without his address being discovered. As we mentioned earlier, spoofing the address would not work, as we require a response. The next scan is called a Zombie scan and will allow us to spoof our address, yet still receive a response. A Zombie scan requires a few conditions to be met. First, the Zombie address used for spoofing must be an active system. There must also be an exploitable, observable condition that is also predictable on the Zombie's system. One such condition would be predictable IP identification numbers (IP ID). Not all systems will be vulnerable to this condition, so a suitable host must first be found. Once a qualified host is found, the attacker will send a simple ping to the Zombie host to determine the current IP ID. The attacker will then use a TCP Connect() scan and spoof the IP of our Zombie host. When the target system receives the spoofed SYN packet on a closed port, a RST will be sent in return to the Zombie host. When the Zombie receives the RST, he will not reply; therefore the IP ID sequence on the Zombie will not change. This tells the attacker that the port is closed, as there was no response from the Zombie. If the Zombie receives a SYN/ACK, a RST will be returned to the victim by the Zombie, and the Zombie's IP ID will increase telling the attacker the port is open. This scan uses the –sI <zombie host> for the Zombie scan and the –p to indicate which port to scan.

```
Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Host   (Remote.Net.61.0) seems to be a subnet broadcast address (returned 1 extra pings). Skipping
host.
Idlescan using zombie Zombie.Host.203.122 (xxx.xxx.203.122:80); Class: Incremental
Interesting ports on Remote.Host.61.4 (xxx.xxx.61.4):
Port       State       Service
21/tcp     open        ftp

Idlescan using zombie Zombie.Host.203.122 (xxx.xxx.203.122:80); Class: Incremental
Interesting ports on Remote.Host.61.5 (xxx.xxx.61.5):
Port       State       Service
21/tcp     open        ftp

Idlescan using zombie Zombie.Host.203.122 (xxx.xxx.203.122:80); Class: Incremental
```

```
Interesting ports on 061-006.camosun.bc.ca (xxx.xxx.61.6):
Port       State      Service
21/tcp     open       ftp

Idlescan using zombie Zombie.Host.203.122 (xxx.xxx.203.122:80); Class: Incremental
The 1 scanned port on Remote.Host.61.11 (xxx.xxx.61.11) is: closed

Idlescan using zombie Zombie.Host.203.122 (xxx.xxx.203.122:80); Class: Incremental
The 1 scanned port on Remote.Host.61.12 (xxx.xxx.61.12) is: closed

Idlescan using zombie Zombie.Host.203.122 (xxx.xxx.203.122:80); Class: Incremental
The 1 scanned port on Remote.Host.61.13 (xxx.xxx.61.13) is: closed

Snort Alert,
Jul 16 13:07:45 Zombie.Host.203.122:80 -> Remote.Host.61.18:21 SYN ******S*
Jul 16 13:07:46 Zombie.Host.203.122:80 -> Remote.Host.61.20:21 SYN ******S*
Jul 16 13:07:47 Zombie.Host.203.122:80 -> Remote.Host.61.21:21 SYN ******S*
Jul 16 13:07:48 Zombie.Host.203.122:80 -> Remote.Host.61.24:21 SYN ******S*
Jul 16 13:07:49 Zombie.Host.203.122:80 -> Remote.Host.61.25:21 SYN ******S*
.
.
Jul 16 13:08:24 Zombie.Host.203.122:80 -> Remote.Host.61.70:21 SYN ******S*
Jul 16 13:08:25 Zombie.Host.203.122:80 -> Remote.Host.61.71:21 SYN ******S*
Jul 16 13:08:26 Zombie.Host.203.122:80 -> Remote.Host.61.73:21 SYN ******S*
Jul 16 13:08:27 Zombie.Host.203.122:80 -> Remote.Host.61.74:21 SYN ******S*
Jul 16 13:08:28 Zombie.Host.203.122:80 -> Remote.Host.61.76:21 SYN ******S*
```

Figure 1.9

As you may have noticed this time, we scanned the entire network. We also changed our parameters to only scan for TCP port 21. Local.Host.201.128 did this scan, yet we get the address zombie.host.203.122. You will also notice that the source port has been set to TCP port 80 for the entire scan. Now if the Analyst decides to block the address, the attacker will be able to continue scanning and just pick a new Zombie address.

1.7 GFI Languard Network Scanner

Nmap will determine information such as open ports and operating systems and is a great program for general information gathering regarding a network or host. In some cases the attacker may want to tailor the scan to a specific application or exploit. For instance, if the attacker wants to execute a known exploit on an Apache web server, he would first, determine what systems have port 80 open, of those systems which ones have Apache running and, possibly the version number. In this section we will take a brief look at GFI Languard Network Scanner so we may be a little more familiar with different types of scanners available. With Languard the attacker is able to determine from a single scan what systems have port 80 open, and the web server and version

number. He is also able to determine much more detailed information about the remote system and other known exploits on the remote system. Based on personal preferences, Languard has some advantages and disadvantages compared to Nmap;

Advantages

Easy to use Graphical user interface
Much more detailed information produced by each scan
Simple to install and use

Disadvantages

Scanning is long and "noisy"
Overwhelming amount of information obtain from a subnet scan
Full-featured version costs money

Languard is offered as a freeware download for non-commercial use but will have some of the functionality disabled. For a fee, a full-featured version is available. Languard may be downloaded from http://www.gfi.com/lannetscan/index.htm. Languard is written for the Windows operating system. We will look at the output from both a Linux scan and a Windows 2000 scan so we may compare the results returned from two separate operating systems. Figure 1.10 shows the standard output of a single Linux host scan.

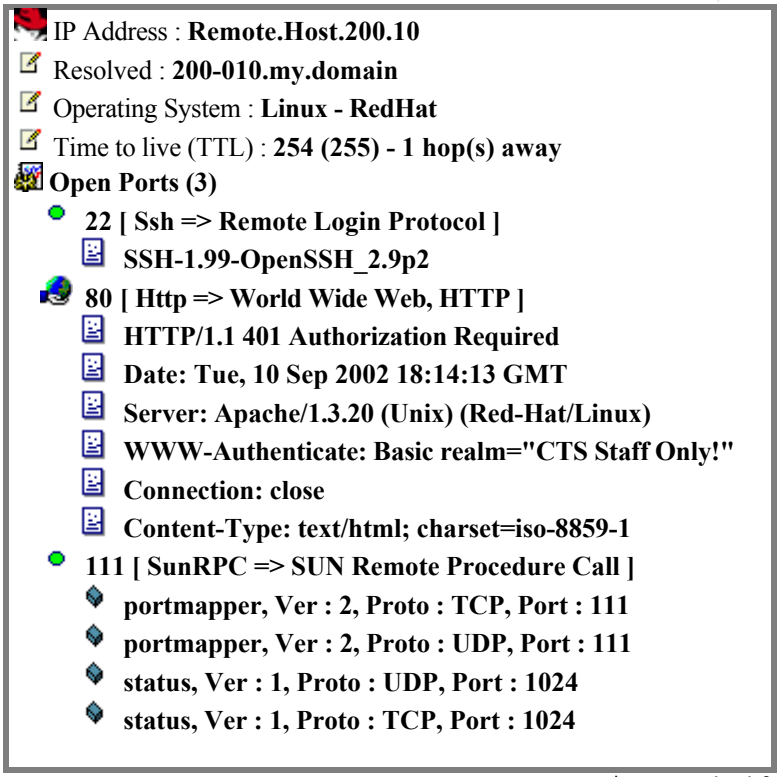

Figure 1.10

The output from the scan gives a lot of information. First, we are given the address scanned and the domain name it resolves to. We are also given the results of an OS fingerprint attempt and the time to live. Next, we are given the open ports found on the system. Unlike Nmap, we are also given detailed information on each port. In the

above scan for port 80, we are made aware of the version of HTTP running, the date, web server version and OS it is running on. Other information such as realm, character set and connection status are also made available. This type of information from a single scan allows the attacker to use an exploit directed at a specific application and version. Due to a situation beyond my control, I was unable to obtain the TCPDump or Snort logs for the output of this scanner.

Figure 1.11 is a scan of a single Windows 2000 host;

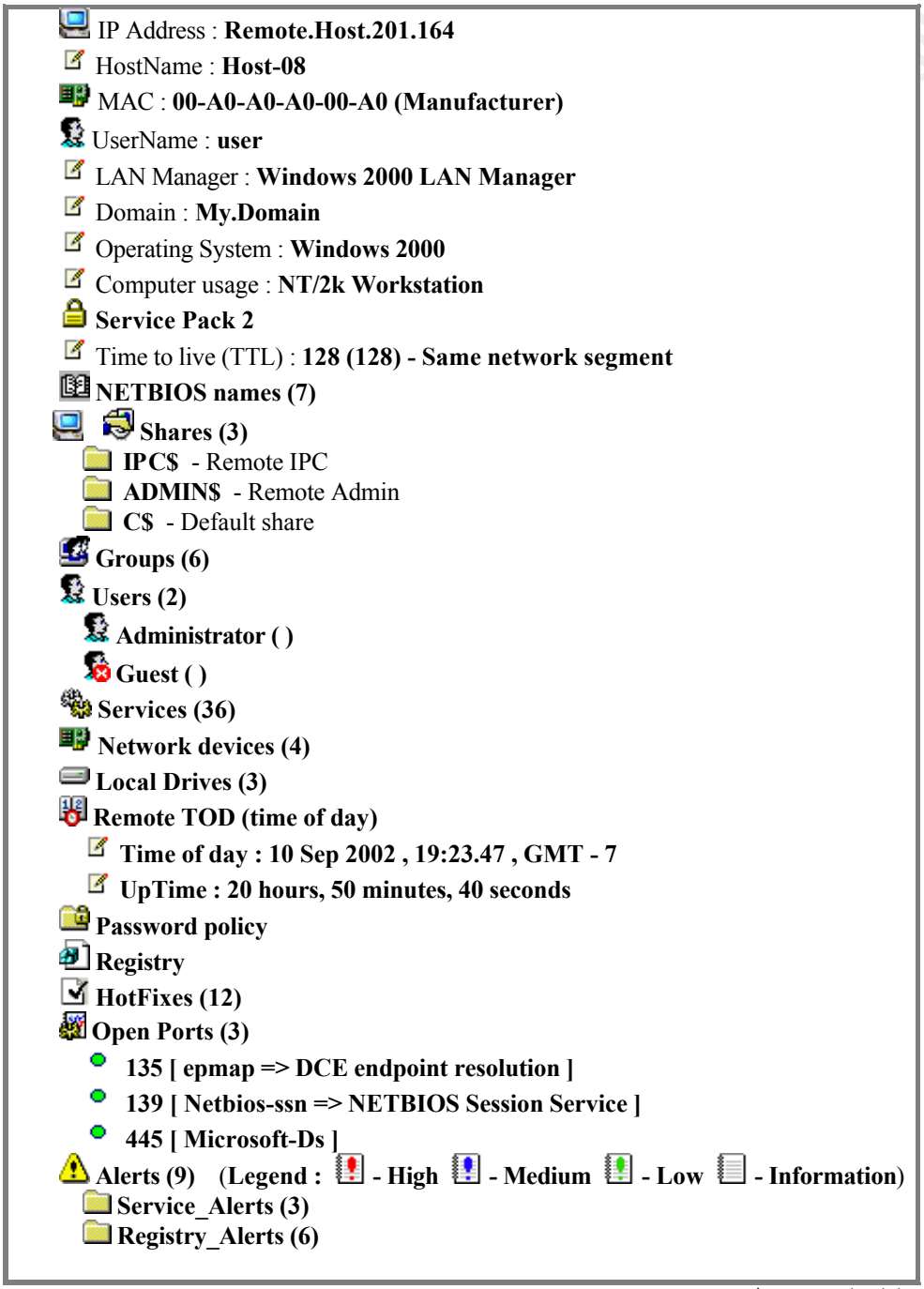

Figure 1.11

The results of the Windows operating system scan has returned much more detailed information than we received with the Linux scan. This includes Registry settings, installed hot fixes, local drives, services, NetBIOS information, file share information, and much more. There was so much information returned, I have had to omit some of the results due to limitations on the length of this paper. Some of the information included but not shown here is specific hot fixes installed, service and registry alerts, registry information, NetBIOS and detailed service information. Also not shown are the password policy, local drives, and user and group information.

To obtain the results shown, Languard will perform many different functions. This includes os fingerprinting, NetBIOS, ICMP, and SNMP queries. To determine alerts, Languard will run scripts that will attempt to begin executing an exploit to the point that the remote system will return information informing Languard of any possible vulnerability. In other words, the exploit is executed but not in it's entirety. The results returned are also susceptible to other variables. If the attacker is scanning from a system that he does not have administrative rights on, file shares will not be shown. Firewall rules or router ACLs may have an impact on the type of information returned, therefore, a Network Administrator scanning internally may obtain very different results than the attacker scanning externally. The administrator should also perform the scan from outside the network and also as a null user to vary what the attack may be seeing.

By simply clicking on a file share displayed in the results, Languard will attempt a connection to the share. If it is not password protected the attacker will have full access to the files. Languard also offers the ability to perform MS SQL Server audits and SNMP audits. Trace routes, DNS lookups and other utilities are also available. If a licensed copy of Languard is running, the ability to remotely install patches and the reporting features are also made available.

Languard can be used by the attacker for fast access to known exploits on the system, type of applications and versions running, direct access to unprotected NetBIOS shares, and detailed information about the system and users. Unlike Nmap, Languard will allow the attacker to not only determine what systems have specific ports open, but also necessary information such as applications and version, allowing the attacker to run a specific exploit against a specific host.

1.8 Social Engineering

I have heard many terms to describe social engineering, but I think the best has to be "psychological hacking". Playing on our emotions, social engineering offers a quick and simple solution to our problems. Social engineering is the art of deception, manipulating a situation's outcome to serve our own purpose. In this section we will look at ways the attacker will gather information about a potential victim. We will see how the attacker is able to take everyday events and use them to their advantage, while most of the time we are none the wiser. The intrusion analyst must be aware that social engineering offers another avenue into our networks. This avenue must be kept in mind when determining how an attack may have originated.

a) Dumpster Diving
Dumpster diving involves searching through garbage to obtain information. Many people will throw away items that may reveal a lot of information. Many people will pay a bill, then throw the bill in the garbage. Look at your phone bill one day and see the type of information that can be obtained. Information such as your full name, address, phone number, and long distance numbers you have called. A business is always a great source for information. We can find out the names of people who have recently left the company, and new people just coming in. We can find out about new promotions or even new clients. Much of the information gained through dumpster diving will give the attacker enough information to make him sound credible when attempting to impersonate an employee.

b) Shoulder Surfing
We all know what shoulder surfing is; when someone peaks over your shoulder to obtain your password. This is the definition most people associate with shoulder surfing but it is a very broad topic. Some con artists make a living hanging out in airports and bus terminals watching over people's shoulders while they use their calling cards, or enter their pin number for their bankcard. Watching you enter your password is another common form of shoulder surfing. Some individuals will even mouth the spelling of their password as they type it in. The attacker will stand behind you reading your email while you discuss the weather or read the numbers off your credit card receipt while you sign it? We think that if there is no one around us we are safe, yet shoulder surfers have been known to use binoculars from a mezzanine or from across a large room. They are fast and, most of the time, you have no idea that they were ever even there, until you receive a large bill for services or products you never purchased.

c) Eavesdropping
Listening in on a conversation from across the room is not as innocent as it may seem, it can reveal a lot of personal information. The attacker can learn about defenses used on a network, or when the best time to attack would be. He may gain personal knowledge about individuals in your organization or even information considered classified. Many attackers will use scanners to intercept cell phone and cordless phone conversations, or use a laptop to try to discover unprotected wireless access points. The attacker may be able to gain enough information on your organization so that any requests made by him are deemed legitimate.

d) Impersonation
Impersonation is probably one of the most recognized forms of social engineering. To rob a home, the attacker will pose as a utility repairman to gain access to the premises. Entering an office as the computer repair company representative is also common.
Internet Relay Chat (IRC) and Instant Messaging (IM) services were recently the host targets for impersonation attacks. While using an IRC or IM service, a user would receive a message from someone claiming to be the system administrator; informing

them a security hole was found on their system. The victim is given a file, told that it's a patch, and if they don't install it, they would be kicked off the system. Of course the patch was really a Trojan giving the attacker access to the user's system.

1.9 Defensive Recommendations

There are many different varieties of scanning techniques available. Some will try to evade your firewall, while others will hide their true source. There are options available to attempt to reduce or slow network scanning. Blocking ICMP echo requests and using network address translation (NAT, RFC 1631) will add a layer of defense but should not be solely relied upon. There are products available, such as LaBrea, used to attempt to slow or even halt a scan. LaBrea is available from http://www.hackbusters.org/. Some firewalls will automatically begin to deny access to any address that scans or attempts to attack them; this will, in effect, prevent that address from accessing any resources on your network. If this is done automatically, the attacker could spoof the addresses of your biggest client, resulting in cutting off any access they may require to your network. For this reason, it is best to manually update your firewall's deny list. Host based firewalls can be used to prevent the hosts from responding to scan attempts adding an additional layer of security. Policies need to be put into place and enforced to help prevent users from leaking confidential information or installing undesirable software. Users should be educated and be aware of techniques used for social engineering, and ways of dealing with any situations that may arise.

1.10 Conclusion

For as long as we have had networks and the Internet, we have had someone who wanted to break in. This was true in the past; it's true now, and will be for the future. This paper took a broad look at different ways information is gathered from our systems, yet we only scratched the surface. Defense in depth has taught us that multiple layers of protection are needed. If the attacker is able to circumvent one set of defenses, another must block them. Many companies are now just becoming aware of the importance of security, yet most still do not take action simply based on cost. Many will claim we have never been hacked so why put money into it? My response to this statement is, "my house has never caught fire yet I still purchase insurance". We need to be aware that the Internet is not the only way into our organization. Our phone lines, trash, and co-workers are just as vulnerable. If you are not monitoring your networks, there is a good chance you have been compromised, you just may not be aware of it. Our systems are being probed every day, we are being watched, and it's just a matter of time before someone attacks. We need to understand the different methods used to probe our networks and to get a better understanding of the tools being used. We must communicate with each other so we may stay informed of new attacks and exploits. We must help to educate one another; we must remain ever vigilant.

1.11 References

Northcutt, Stephen. Network Intrusion Detection, An Analyst's Handbook. USA
New Riders Publishing, 1999

Cole, Eric. Hackers Beware. USA
New Riders Publishing, 2002

Siyan, Karanjit S. Inside TCP/IP third Edition. USA
New Riders Publishing, 1997

S.M. Bellovin. Security Problems in the TCP/IP Protocol Suite. Murray Hill, NJ
http://www.insecure.org/stf/tcpip_smb.txt

Gaudin, Sharon. Social Engineering: The Human Side Of Hacking. 2002
http://itmanagement.earthweb.com/secu/article/0,,11953_1040881,00.html

Arkin, Ofir. ICMP Usage In Scanning: The Complete Know-How Version 3.0. 2001
http://www.sys-security.com/html/projects/icmp.html

Websites:
http://www.insecure.org/nmap/index.html
http://www.snort.org/
http://www.ietf.org/rfc.html

2. Assignment 2 – Network Detects

2.1 Introduction

In this portion of the assignment we will look at three different possible attacks. Figure 2.0 shows a general overview of the network where most of the detects were collected.

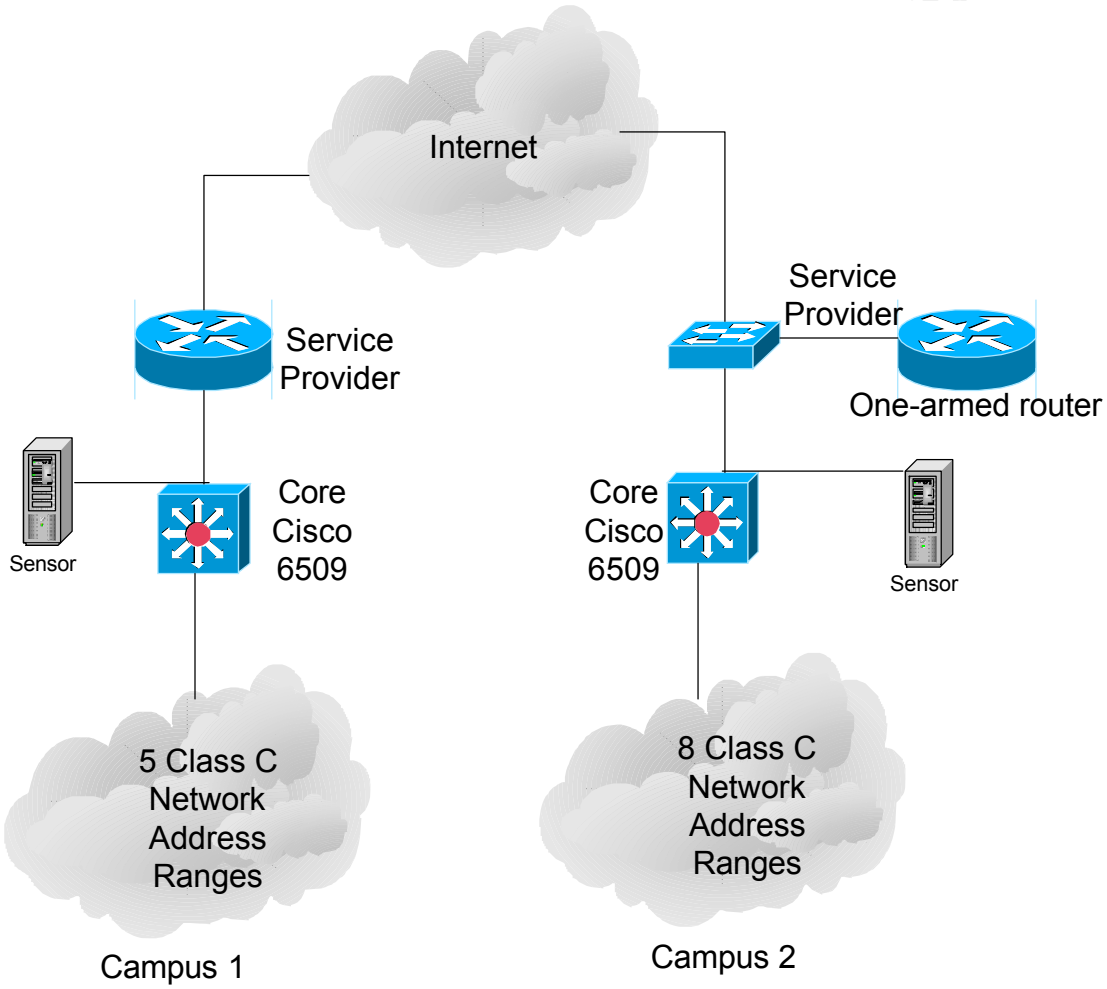

Figure 2.0

Most captures were detected on a Snort 1.8.6 sensor running on a Pentium II 350 with RedHat Linux 7.2. Both sensors are monitoring the ingress/egress port of a multi-layer switch in the core of the autonomous system.

Table 2.0 shows a Snort alert and the breakdown of the information contained.

[**] INFO FTP anonymous login attempt [**] 04/27-12:29:02.710865 66.218.231.51:4084 -> My.Host.63.4:21 TCP TTL:43 TOS:0x0 ID:15757 IpLen:20 DgmLen:56 DF ***AP*** Seq: 0xB0257ABB Ack: 0x31444272 Win: 0xFAD9 TcpLen: 20	
INFO FTP anonymous login attempt	Snort signature
06/23-02:05:04.474488	Date and Time indicator
66.218.231.51:4084 -> My.Host.63.4:21	Source Address and Port, data flow indicator, Destination address and port.
TCP TTL:43 TOS:0x0 ID:15757	Encapsulated protocol, Time To Live, Type Of Service, Packet Identifier
IpLen:20 DgmLen:56 DF	IP header Length and Datagram length expressed in Bytes with Do not fragment flag set.
AP	Set TCP flag indicator
Seq: 0xB0257ABB Ack: 0x31444272	Sequence and Acknowledgement numbers
Win: 0xFAD9 TcpLen: 20	Window size and TCP header length

Table 2.0

2.2 Detect 1- NETBIOS SMB C Access

```
[**] NETBIOS SMB C access [**]
06/25-13:42:52.517611 65.186.245.85:4156 -> My.Host.61.12:139
TCP TTL:109 TOS:0x0 ID:1381 IpLen:20 DgmLen:206 DF
***AP*** Seq: 0x1FFB1E4  Ack: 0x144D26  Win: 0x21D3  TcpLen: 20
0x0000: 00 05 5E 37 56 82 00 01 42 5F 16 00 08 00 45 00  ..^7V...B_....E.
0x0010: 00 CE 05 65 40 00 6D 06 C6 FA 41 BA F5 55 xx xx  ...e@.m...A..U..
0x0020: xx xx 10 3C 00 8B 01 FF B1 E4 00 14 4D 26 50 18  =..<........M&P.
0x0030: 21 D3 3A 7A 00 00 00 00 00 A2 FF 53 4D 42 73 00  !.:z.......SMBs.
0x0040: 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0050: 00 00 00 00 D9 1B 01 00 03 98 0D 75 00 84 00 68  ...........u...h
0x0060: 0B 32 00 00 00 00 00 00 00 18 00 00 00 00 00 00  .2..............
0x0070: 00 05 00 00 00 47 00 D0 99 0E C6 B5 FB DE EC BF  .....G..........
0x0080: 15 BD CA 21 97 B5 A1 E6 AA E1 25 96 21 4F AA 4A  ...!......%.!O.J
0x0090: 49 4D 20 57 49 4C 4B 49 52 53 4F 4E 00 54 4F 50  IM WILKIRSON.TOP
0x00A0: 49 54 4F 46 46 00 57 69 6E 64 6F 77 73 20 34 2E  ITOFF.Windows 4.
0x00B0: 30 00 57 69 6E 64 6F 77 73 20 34 2E 30 00 04 FF  0.Windows 4.0...
0x00C0: 00 00 00 02 00 01 00 13 00 00 5C 5C 49 4E 54 42  ..........\\INTB
0x00D0: 4F 4F 4B 2D 32 30 5C 43 00 41 3A 00              xxx-20\C.A:.
```

```
[**] NETBIOS SMB C access [**]
06/25-13:43:09.687611 65.186.245.85:4158 -> My.Host.61.14:139
TCP TTL:109 TOS:0x0 ID:5989 IpLen:20 DgmLen:205 DF
***AP*** Seq: 0x1FFF4E8  Ack: 0x4DDAC357  Win: 0x21D3  TcpLen: 20
0x0000: 00 05 5E 37 56 82 00 01 42 5F 16 00 08 00 45 00  ..^7V...B_....E.
0x0010: 00 CD 17 65 40 00 6D 06 B4 F9 41 BA F5 55 xx xx  ...e@.m...A..U..
0x0020: xx xx 10 3E 00 8B 01 FF F4 E8 4D DA C3 57 50 18  =..>......M..WP.
0x0030: 21 D3 63 CE 00 00 00 00 00 A1 FF 53 4D 42 73 00  !.c........SMBs.
0x0040: 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0050: 00 00 00 00 D9 1B 01 00 03 99 0D 75 00 84 00 68  ...........u...h
0x0060: 0B 32 00 00 00 00 00 00 00 18 00 00 00 00 00 00  .2..............
0x0070: 00 05 00 00 00 47 00 D6 DA 1B 1D 33 93 32 52 36  .....G.....3.2R6
0x0080: 0E D9 E7 9D 90 58 B5 38 39 2F 69 BC 4B F9 23 4A  .....X.89/i.K.#J
0x0090: 49 4D 20 57 49 4C 4B 49 52 53 4F 4E 00 54 4F 50  IM WILKIRSON.TOP
0x00A0: 49 54 4F 46 46 00 57 69 6E 64 6F 77 73 20 34 2E  ITOFF.Windows 4.
0x00B0: 30 00 57 69 6E 64 6F 77 73 20 34 2E 30 00 04 FF  0.Windows 4.0...
0x00C0: 00 00 00 02 00 01 00 12 00 00 5C 5C 43 43 32 32  ..........\\host
0x00D0: 37 2D 30 31 42 5C 43 00 41 3A 00                 7-01B\C.A:.
```

Figure 2.1

a) Source of trace:

The traces shown in figure 2.1 were obtained on June 25 2002. The captured trace was detected on the network depicted in figure 2.0 and was directed at a single class C network located on Campus 1. There were 62 total packets captured for this attack. Each packet contained a different destination host address, but the same source address. Figure 2.2 is a Whois query from the American Registry for Internet Numbers (ARIN), pertaining to the displayed packet's source address.

```
Telocity (NETBLK-TELOCITY-4)
   10355 N. De Anza Blvd
   Cupertino, CA 95014
   US

   Netname: TELOCITY-4
   Netblock: 65.184.0.0 - 65.191.255.255
   Maintainer: TELO

   Coordinator:
      Telocity  (ZT26-ARIN)  ip-admin@telocity.net
      408-863-6600

   Domain System inverse mapping provided by:

   NS1.TELOCITY.NET  216.227.56.20
   NS2.TELOCITY.NET  216.227.112.50

   ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

   Record last updated on 18-Oct-2001.
   Database last updated on  25-Jul-2002 20:00:35 EDT.
```

Figure 2.2

b) Detect was generated by:
This detect was generated by the sensor monitoring the ingress/egress port on the autonomous system's core router. The sensor is Snort version 1.8.6 running on a 350MHz Pentium II running RedHat 7.2. The rule set that generated this detect was current at the time. Figure 2.3 shows the rule used to generate the alert. (The single line rule is wrapped for clarity).

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB C$ access";
flags:A+; content: "|5c|C$|00 41 3a 00|";reference:arachnids,339;
classtype:attempted-recon; sid:533; rev:5;)
```

Figure 2.3

The alert was generated based on the following rule matches.

a) Packet contained TCP information
b) The traffic flow was external to internal
c) Destination port TCP 139 was set (NETBIOS Session Service)
d) ACK and PSH flags are set
c) Packet contents matched "|5c|C$|00 41 3a 00|"

c) Probability the source address was spoofed:
It is unlikely the source address is spoofed in this case. This packet would be part of an established TCP session. If no response were obtained, the attacker would not be able to complete the attack. No other references or information concerning previous attacks were found with regards to this IP address.

d) Description of attack:
This attack is a file share access attempt. The attacker is attempting to make a connection to the victim's C$ administrative share, as described in arachnids IDS339. The attacker would try to make a connection to port 139 (NETBIOS session services). Port 139 is used when SMB is run on NETBIOS over TCP/IP (NBT). This is consistent with Windows operating systems before Windows 2000. With the introduction of Windows 2000, port 445 is now used to run SMB directly over TCP/IP. At the time this paper was written, this attack was still under review with CVE and can be referenced with CAN-1999-0621.

e) Attack Mechanism:
The intent of this attack is to map the administrative default C$ share of a Windows workstation. If a weak password, or no password, is set, the attacker may be able to gain access to the file system on the C:\ drive. This type of attack can be accomplished using the Windows "net use \\remotehost\C$" command. Software applications for system scanning such as GFI LANguard Network Security Scanner, will also automate the process, allowing large amount of systems to be accessed in a short period of time.

f) Correlations:
The following resources were used for information gathering and to correlate information on the NETBIOS SMB C Access.
http://www.whitehats.com/IDS/339

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0621
http://www.gfisoftware.com/lannetscan/index.htm

g) Evidence of active targeting:
There were a total of 62 packets detected against a single class C network range. There were no previous scans or attacks detected from this address. It would seem the network attacked might have been randomly chosen.

h) Severity:
Severity = (criticality + lethality) – (system countermeasures + network countermeasures)
Scale = 1 lowest 5 highest

Criticality = 2	Windows 95/NT/2000 host only, no servers or critical systems affected.
Lethality = 3	The ability to obtain and crack local passwords and access network resources becomes a concern. The attacker could have access to confidential information and also set up an account on the system. The attacker may also be able to compromise or get access to an administrator account.
System Countermeasures = 4	Most systems do not allow for remote drive mapping. Those that do allow mapping, have strong passwords in place.
Network Countermeasures = 1	Network intrusion detection system detected the attack.

Severity = (2 + 3) – (4+1) = 0

I) Defensive Recommendation:
1) Ensure all current patches and service packs are installed
2) If file and print sharing is not needed, turn it off
3) If file and print sharing is needed, use strong, hard to break passwords
4) Personal firewalls such as Zone Alarm will help to prevent access
5) Block access to ports 135, 137, 139, and 445 at the router or firewall
6) Implement a VPN if remote access is required

j) Multiple Choice Test Question
NETBIOS Session Services used port 139 pre Windows 2000. What port is used with Windows 2000?
a) 136
b) 554
c) 139
d) 445
Answer: d

2.3 Detect 2 – Apache Chunked-Encoding worm attempt

```
[**] WEB-MISC Apache Chunked-Encoding worm attempt [**]
08/21-04:12:36.668218 207.99.15.5:3184 -> My.Host.200.10:80
TCP TTL:45 TOS:0x0 ID:2115 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x96956394  Ack: 0x6899367A  Win: 0x4470  TcpLen: 20
0x0000: 00 03 32 BE C7 07 00 07 EB AE C3 41 08 00 45 00  ..2........A..E.
0x0010: 05 DC 08 43 40 00 2D 06 0B 47 CF 63 0F 05 xx xx  ...C@.-..G.c....
0x0020: xx xx 0C 70 00 50 96 95 63 94 68 99 36 7A 50 10  ...p.P..c.h.6zP.
0x0030: 44 70 83 A0 00 00 50 4F 53 54 20 2F 20 48 54 54  Dp....POST / HTT
0x0040: 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 55 6E 6B  P/1.1..Host: Unk
0x0050: 6E 6F 77 6E 0D 0A 58 2D 43 43 43 43 43 43 43 3A  nown..X-CCCCCCC:
0x0060: 20 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAA
.
.            (60 identical lines removed for clarity)
.
0x0430: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0x0440: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0x0450: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0x0460: 41 68 47 47 47 47 89 E3 31 C0 50 50 50 50 C6 04  AhGGGG..1.PPPP..
0x0470: 24 04 53 50 50 31 D2 31 C9 B1 80 C1 E1 18 D1 EA  $.SPP1.1........
0x0480: 31 C0 B0 85 CD 80 72 02 09 CA FF 44 24 04 80 7C  1.....r....D$..|
0x0490: 24 04 20 75 E9 31 C0 89 44 24 04 C6 44 24 04 20  $. u.1..D$..D$.
0x04A0: 89 64 24 08 89 44 24 0C 89 44 24 10 89 44 24 14  .d$..D$..D$..D$.
0x04B0: 89 54 24 18 8B 54 24 18 89 14 24 31 C0 B0 5D CD  .T$..T$...$1..].
0x04C0: 80 31 C9 D1 2C 24 73 27 31 C0 50 50 50 50 FF 04  .1..,$s'1.PPPP..
0x04D0: 24 54 FF 04 24 FF 04 24 FF 04 24 FF 04 24 51 50  $T..$..$..$..$QP
0x04E0: B0 1D CD 80 58 58 58 58 58 3C 4F 74 0B 58 58 41  ....XXXXX<Ot.XXA
0x04F0: 80 F9 20 75 CE EB BD 90 31 C0 50 51 50 31 C0 B0  .. u....1.PQP1..
0x0500: 5A CD 80 FF 44 24 08 80 7C 24 08 03 75 EF 31 C0  Z...D$..|$..u.1.
0x0510: 50 C6 04 24 0B 80 34 24 01 68 42 4C 45 2A 68 2A  P..$..4$.hBLE*h*
0x0520: 47 4F 42 89 E3 B0 09 50 53 B0 01 50 50 B0 04 CD  GOB....PS..PP...
0x0530: 80 31 C0 50 68 6E 2F 73 68 68 2F 2F 62 69 89 E3  .1.Phn/shh//bi..
0x0540: 50 53 89 E1 50 51 53 50 B0 3B CD 80 CC 0D 0A 58  PS..PQSP.;.....X
0x0550: 2D 43 43 43 43 43 43 43 3A 20 41 41 41 41 41 41  -CCCCCCC: AAAAAA
0x0560: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0x0570: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0x0580: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0x0590: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0x05A0: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0x05B0: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0x05C0: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0x05D0: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0x05E0: 41 41 41 41 41 41 41 41 41 41                    AAAAAAAAAA
```

Figure 2.4

a) Source of trace:

The trace in figure 2.4 was obtained on August 21, 2002. It was captured from the network depicted in figure 2.0 and was directed at a single host located on Campus 1. There were 21 total packets captured for this attack. The following is a Whois search pertaining to the displayed packet's source address.

```
OrgName:    Net Access Corporation
```

```
OrgID:      NAC

NetRange:   207.99.0.0 - 207.99.127.255
CIDR:       207.99.0.0/17
NetName:    NAC-NETBLK01
NetHandle:  NET-207-99-0-0-1
Parent:     NET-207-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.NAC.NET
NameServer: NS2.NAC.NET
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

            * Reassignment information for this network is
available
            * at whois.nac.net 43
RegDate:    1996-04-23
Updated:    2001-08-22

TechHandle: ZN77-ARIN
TechName:   Net Access Corporation
TechPhone:  +1-800-638-6336
TechEmail:  legal@nac.net
```

Figure 2.5

b) Detect was generated by:

This detect was generated by the sensor monitoring the ingress/egress port on the autonomous systems core router. The sensor is Snort version 1.9.0beta6 running on a 350MHz Pentium II running RedHat 7.2. The rule set that generated this detect was current at the time. Figure 2.6 shows the rule used to generate the alert.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC Apache
Chunked-Encoding worm attempt"; flow:to_server,established;
content:"CCCCCCC\: AAAAAAAAAAAAAAAAAAA"; nocase; classtype:web-application-
attack; reference:bugtraq,4474;
```

Figure 2.6

The alert was generated based on the following rule matches.

a) Packet contained TCP information
b) The traffic flow was external to an internal web server
c) Destination port TCP 80 was set (HTTP Port)
d) Packet contents matched "CCCCCCC: AAAAAAAAAAAAAAAAAAA"

c) Probability the source address was spoofed:

This is a self-launching worm, it is software only; it requires no human intervention beyond initial configuration. The worm will need to know if a host is live, and if it is running Apache. For the worm to acquire this information it must receive a response from the victim. The worm randomly generates the addresses it attacks, so it is highly unlikely the worm is in a position to sniff the traffic coming from the victim affording the information required. Therefore, it is highly unlikely the source address in this attack is spoofed.

d) Description of attack:

Taking advantage of the known vulnerability "Apache Web Server Chunk Handling" in un-patched versions of Apache web servers running on the FreeBSD platform, the Apache worm will attempt to infect a vulnerable host, and set up a backdoor to UDP port 2001. The backdoor will allow remote control of the worm and such functions as;

a) Obtain email address from the infected host
b) Distributed denial of service (ddos) agent/client
c) Access to local web pages
d) Send email (spam)
e) DNS, UDP, TCP flooding

The Apache worm is known by many names and includes different variants. Some of the known names include; BSD/Scalper.worm, ELF_Scalper.A, Scalper.A, FreeBSD.Scalper.worm, and Unix/Scalper.A

e) Attack Mechanism:

There is a known vulnerability in the Apache web server version 1.2.2 and above, 1.3 to 1.3.24, and 2.0 to 2.0.36 known as the Apache Web Server Chunk Handling Vulnerability. This vulnerability is described in Cert Advisory CA-2002-17 and can be found at http://www.cert.org/advisories/CA-2002-17.html. This vulnerability is based on how Apache handles data encoded in chunks based on the HTTP 1.1 standard described in RFC2616. Crafting an invalid chunk encoded request will have varied results depending on the operating system in use by the victim computer. Results range from increased processor load to a segmentation violation terminating the child process, causing a denial of service. On the FreeBSD platform, a stack overflow can occur that can be controlled allowing any addresses on the stack to be further exploited, allowing arbitrary code to be run at the same user level as the child process. This will allow the Apache worm to compromise the FreeBSD system running the vulnerable Apache server.

The Apache worm will first scan a predefined address range, based on the first octet of the ip address hard coded into the worm. The second octet of the address is randomly generated and the third and fourth are incremental. The worm issues an HTTP get request on port 80 of the ip address to locate the Apache web servers. Once a server is located the Apache Web Server Chunk Handling vulnerability exploit is attempted. Apache web servers running on FreeBSD hosts are vulnerable to a controlled buffer overflow resulting from the Chunk Handling exploit. Therefore, If the exploit is successful, the infected computer will execute a command shell on the target host and send itself in Uuencoded form as /tmp/.uua . It will then decode as /tmp/.a and executes. This will set up a backdoor to UDP port 2001 and begin the scanning process for more vulnerable hosts. The backdoor will allow for remote control of the worm and to perform actions such as sending of email messages, ddos attacks, execution of arbitrary programs, TCP, UDP and DNS flooding. Information regarding the Apache worm can be found at http://www.mycert.org.my/advisory/MA-044.072002.html

f) Correlations:

The following resources were used for information gathering and to correlate information on the Apache Chunked-Encoding worm.

http://www.cert.org/advisories/CA-2002-17.html
http://httpd.apache.org/info/security_bulletin_20020617.txt
http://www.auscert.org.au/Information/Advisories/advisory/AL-2002.06.txt
http://www.mycert.org.my/advisory/MA-044.072002.html

g) Evidence of active targeting:

There were a total of 21 packets detected against a single server. A total of 2383 scans to port 80 across 4 class C subnets was first detected. The following is a partial list of the scan taken from the Snort port scan log.

```
Aug 21 04:12:34 207.99.15.5:3044 -> My.Host.200.10:80 SYN
******S*
Aug 21 04:12:31 207.99.15.5:3046 -> My.Host.201.124:80 SYN
******S*
Aug 21 04:12:34 207.99.15.5:3043 -> My.Host.202.13:80 SYN
******S*
Aug 21 04:12:31 207.99.15.5:3045 -> My.Host.203.244:80 SYN
******S*
```

Figure 2.7

There was also a "Transfer-Encoding: chunked" alert detected for the local ip address affected by the worm attack. This alert indicates an attempt to transfer encoded chunks to the victim servers. The following is the packet detected for this alert followed by the rule used to trigger the alert.

```
[**] WEB-MISC Transfer-Encoding: chunked [**]
08/21-04:12:37.108218 207.99.15.5:3184 -> My.Host.200.10:80
TCP TTL:45 TOS:0x0 ID:2140 IpLen:20 DgmLen:234 DF
***AP*** Seq: 0x9695E10C  Ack: 0x6899367A  Win: 0x4470  TcpLen: 20
0x0000: 00 03 32 BE C7 07 00 07 EB AE C3 41 08 00 45 00  ..2........A..E.
0x0010: 00 EA 08 5C 40 00 2D 06 10 20 CF 63 0F 05 xx xx  ...\@.-.. .c....
0x0020: xx xx 0C 70 00 50 96 95 E1 0C 68 99 36 7A 50 18  ...p.P....h.6zP.
0x0030: 44 70 91 C2 00 00 00 00 0D 0A 58 2D 41 41 41 41  Dp........X-AAAA
0x0040: 3A 20 00 DE BF BF 00 DE BF BF 00 DE BF BF 00 DE  : ..............
0x0050: BF BF 00 DE BF BF 00 DE BF BF 00 00 00 00 00 00  ................
0x0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0D 0A  ................
0x0080: 58 2D 41 41 41 41 3A 20 00 DE BF BF 00 DE BF BF  X-AAAA: ........
0x0090: 00 DE BF BF 00 DE BF BF 00 DE BF BF 00 DE BF BF  ................
0x00A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x00B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x00C0: 00 00 00 00 0D 0A 54 72 61 6E 73 66 65 72 2D 45  ......Transfer-E
0x00D0: 6E 63 6F 64 69 6E 67 3A 20 63 68 75 6E 6B 65 64  ncoding: chunked
0x00E0: 0D 0A 0D 0A 35 0D 0A 42 42 42 42 42 0D 0A 66 66  ....5..BBBBB..ff
0x00F0: 66 66 66 66 36 65 0D 0A                          ffff6e..
```

Figure 2.8

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-MISC
Transfer-Encoding\: chunked"; flow:to_server,established; content:"Transfer-
Encoding\:"; nocase; content:"chunked"; nocase;)
```

Figure 2.9

Finally, the following entries in the Httpd error log file indicating the Chunked Code Handling exploit was successful at terminating a child process belonging to the Apache web server. Also, the access log file indicates the attempt to discover what web server is running.

```
(Error Log)
[Wed Aug 21 04:12:33 2002] [error] [client 207.99.15.5] client sent
HTTP/1.1 request without hostname (see RFC2616 section 14.23): /

[Wed Aug 21 04:12:37 2002] [notice] child pid 10162 exit signal
Segmentation fault (11)

(Access Log)
207.99.15.5 - - [21/Aug/2002:04:12:33 -0700] "GET / HTTP/1.1" 400 385 "-" "-
"
```

Figure 2.10

Although the Apache worm is self-launching, the worm will begin by scanning a random network. Once Apache is found active on a system the worm will begin to actively target that server to attempt a compromise. Therefore I have concluded that active targeting is taking place on the local network against the Apache server on this host.

h) Severity:
Severity = (criticality + lethality) – (system countermeasures + network countermeasures)
Scale = 1 lowest 5 highest

Criticality = 4 This system is used in a data security role within the organization.

Lethality = 4 The Apache Web Server Chunk Handling Vulnerability attempt, can result in anything from increased resource consumption on the victim, to a denial of service resulting in a compromise. This would allow for further exploits to be run from the victim such as a denial of service attacks against other hosts, email spamming, remote access to the compromised host allowing arbitrary programs to be run, and further propagation of the worm.

System Countermeasures = 2 System did not maintain current patches or an updated Apache web server. There was also no virus protection on the system or host based firewall. The OS although vulnerable to the Apache Web Server Chunk Handling Vulnerability, is not vulnerable to the Apache worm.

Network Countermeasures = 1 Network intrusion detection system detected the

attack. At the time there was no firewall or virus detection monitoring the network.

Severity = (4 + 4) – (2+1) = 5

i) Defensive Recommendation:

1) Ensure all current patches and service packs are installed
2) Updating Apache to version 1.3.26 or 2.0.39
3) Run intrusion detection / virus scanning at the network and host level
4) A code update is available from http://www.securiteam.com/tools/5WP0M0U7FS.html to deny and log any chunked encoding at the post-read request phase.

j) Multiple Choice Test Question

The Apache worm will compromise and allow a host system to be used as a DDoS agent. In a DDoS attack what role does the agent play?

a) The agent is a computer used to issue commands to a resource such as an IRC channel or other compromised computer that is responsible for directing a group of computers to perform a DDoS attack.

b) The agent is a method or resource such as an IRC channel or other compromised computer used to distribute commands to a group of computers performing a DDoS attack.

c) The agent is one of many computers performing a DDoS who receiving orders from a resource such as an IRC channel or other compromised computer.

d) Agents monitor network traffic looking for previously compromised systems to attack.

Answer: C

2.4 Detect 3 – Code Red Worm / IIS Indexing Service Buffer Overflow

```
[**] WEB-IIS ISAPI .ida attempt [**]
05/16-00:21:25.684488 65.48.127.17:4755 -> Remote.Host.212.165:80
TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504
***AP*** Seq: 0x7A7D95E9  Ack: 0x535F89F1  Win: 0x7D78  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 10  .....3....&...E.
0x0010: 05 E0 00 00 00 00 F0 06 00 00 41 30 7F 11 xx xx  ..........A0..N%
0x0020: xx xx 12 93 00 50 7A 7D 95 E9 53 5F 89 F1 50 18  .....Pz}..S_..P.
0x0030: 7D 78 00 00 00 00 47 45 54 20 2F 64 65 66 61 75  }x....GET /defau
0x0040: 6C 74 2E 69 64 61 3F 4E 4E 4E 4E 4E 4E 4E 4E 4E  lt.ida?NNNNNNNNN
0x0050: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0060: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0070: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0080: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0090: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00A0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00B0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00C0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00D0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00E0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00F0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0100: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0110: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0120: 4E 4E 4E 4E 4E 4E 4E 00 00 00 00 00 00 00 00 00  NNNNNNN.........
0x0130: 00 00 00 00 00 00 C3 03 00 00 00 78 00 FA 20 25  ...........x.. %
0x0140: 75 39 30 39 30 25 75 36 38 35 38 25 75 63 62 64  u9090%u6858%ucbd
0x0150: 33 25 75 37 38 30 31 25 75 39 30 39 30 25 75 36  3%u7801%u9090%u6
0x0160: 38 35 38 25 75 63 62 64 33 25 75 37 38 30 31 25  858%ucbd3%u7801%
0x0170: 75 39 30 39 30 25 75 39 30 39 30 25 75 38 31 39  u9090%u9090%u819
0x0180: 30 25 75 30 30 63 33 25 75 30 30 30 33 25 75 38  0%u00c3%u0003%u8
0x0190: 62 30 30 25 75 35 33 31 62 25 75 35 33 66 66 25  b00%u531b%u53ff%
0x01A0: 75 30 30 37 38 25 75 30 30 30 30 25 75 30 30 3D  u0078%u0000%u00=
0x01B0: 61 20 20 48 54 54 50 2F 31 2E 30 0D 0A 43 6F 6E  a  HTTP/1.0..Con
0x01C0: 74 65 6E 74 2D 74 79 70 65 3A 20 74 65 78 74 2F  tent-type: text/
0x01D0: 78 6D 6C 0A 48 4F 53 54 3A 77 77 77 2E 77 6F 72  xml.HOST:www.wor
0x01E0: 6D 2E 63 6F 6D 0A 20 41 63 63 65 70 74 3A 20 2A  m.com. Accept: *
0x01F0: 2F 2A 0A 43 6F 6E 74 65 6E 74 2D 6C 65 6E 67 74  /*.Content-lengt
0x0200: 68 3A 20 33 35 36 39 20 0D 0A 0D 0A 55 8B EC 81  h: 3569 ....U...
0x0210: EC 18 02 00 00 53 56 57 8D BD E8 FD FF FF B9 86  .....SVW........
0x0220: 00 00 00 B8 CC CC CC CC F3 AB C7 85 70 FE FF FF  ............p...
0x0230: 00 00 00 00 E9 0A 0B 00 00 8F 85 68 FE FF FF 8D  ...........h....
0x0240: BD F0 FE FF FF 64 A1 00 00 00 00 89 47 08 64 89  .....d......G.d.
0x0250: 3D 00 00 00 00 E9 6F 0A 00 00 8F 85 60 FE FF FF  =.....o.....`...
0x0260: C7 85 F0 FE FF FF FF FF FF FF 8B 85 68 FE FF FF  ............h...
0x0270: 83 E8 07 89 85 F4 FE FF FF C7 85 58 FE FF FF 00  ...........X....
0x0280: 00 E0 77 E8 9B 0A 00 00 83 BD 70 FE FF FF 00 0F  ..w.......p.....
0x0290: 85 DD 01 00 00 8B 8D 58 FE FF FF 81 C1 00 00 01  .......X........
0x02A0: 00 89 8D 58 FE FF FF 81 BD 58 FE FF FF 00 00 00  ...X.....X......
0x02B0: 78 75 0A C7 85 58 FE FF FF 00 00 F0 BF 8B 95 58  xu...X.........X
0x02C0: FE FF FF 33 C0 66 8B 02 3D 4D 5A 00 00 0F 85 9A  ...3.f..=MZ.....
0x02D0: 01 00 00 8B 8D 58 FE FF FF 8B 51 3C 8B 85 58 FE  .....X....Q<..X.
0x02E0: FF FF 33 C9 66 8B 0C 10 81 F9 50 45 00 00 0F 85  ..3.f.....PE....
0x02F0: 79 01 00 00 8B 95 58 FE FF FF 8B 42 3C 8B 8D 58  y.....X....B<..X
0x0300: FE FF FF 8B 54 01 78 03 95 58 FE FF FF 89 95 54  ....T.x..X.....T
0x0310: FE FF FF 8B 85 54 FE FF FF 8B 48 0C 03 8D 58 FE  .....T....H...X.
0x0320: FF FF 89 8D 4C FE FF FF 8B 95 4C FE FF FF 81 3A  ....L.....L....:
```

```
0x0330: 4B 45 52 4E 0F 85 33 01 00 00 8B 85 4C FE FF FF  KERN..3.....L...
0x0340: 81 78 04 45 4C 33 32 0F 85 20 01 00 00 8B 8D 58  .x.EL32.. .....X
0x0350: FE FF FF 89 8D 34 FE FF FF 8B 95 54 FE FF FF 8B  .....4.....T....
0x0360: 85 58 FE FF FF 03 42 20 89 85 4C FE FF FF C7 85  .X....B ..L.....
0x0370: 48 FE FF FF 00 00 00 00 EB 1E 8B 8D 48 FE FF FF  H...........H...
0x0380: 83 C1 01 89 8D 48 FE FF FF 8B 95 4C FE FF FF 83  .....H.....L....
0x0390: C2 04 89 95 4C FE FF FF 8B 85 54 FE FF FF 8B 8D  ....L.....T.....
0x03A0: 48 FE FF FF 3B 48 18 0F 8D C0 00 00 00 8B 95 4C  H...;H.........L
0x03B0: FE FF FF 8B 02 8B 8D 58 FE FF FF 81 3C 01 47 65  .......X....<.Ge
0x03C0: 74 50 0F 85 A0 00 00 00 8B 95 4C FE FF FF 8B 02  tP........L.....
0x03D0: 8B 8D 58 FE FF FF 81 7C 01 04 72 6F 63 41 0F 85  ..X....|..rocA..
0x03E0: 84 00 00 00 8B 95 48 FE FF FF 03 95 48 FE FF FF  ......H.....H...
0x03F0: 03 95 58 FE FF FF 8B 85 54 FE FF FF 8B 48 24 33  ..X.....T....H$3
0x0400: C0 66 8B 04 0A 89 85 4C FE FF FF 8B 8D 54 FE FF  .f.....L.....T..
0x0410: FF 8B 51 10 8B 85 4C FE FF FF 8D 4C 10 FF 89 8D  ..Q...L....L....
0x0420: 4C FE FF FF 8B 95 4C FE FF FF 03 95 4C FE FF FF  L.....L.....L...
0x0430: 03 95 4C FE FF FF 03 95 4C FE FF FF 03 95 58 FE  ..L.....L.....X.
0x0440: FF FF 8B 85 54 FE FF FF 8B 48 1C 8B 14 0A 89 95  ....T....H......
0x0450: 4C FE FF FF 8B 85 4C FE FF FF 03 85 58 FE FF FF  L.....L.....X...
0x0460: 89 85 70 FE FF FF EB 05 E9 0D FF FF FF E9 16 FE  ..p.............
0x0470: FF FF 8D BD F0 FE FF FF 8B 47 08 64 A3 00 00 00  .........G.d....
0x0480: 00 83 BD 70 FE FF FF 00 75 05 E9 38 08 00 00 C7  ...p....u..8....
0x0490: 85 4C FE FF FF 01 00 00 00 EB 0F 8B 8D 4C FE FF  .L...........L..
0x04A0: FF 83 C1 01 89 8D 4C FE FF FF 8B 95 68 FE FF FF  ......L.....h...
0x04B0: 0F BE 02 85 C0 0F 84 8D 00 00 00 8B 8D 68 FE FF  .............h..
0x04C0: FF 0F BE 11 83 FA 09 75 21 8B 85 68 FE FF FF 83  .......u!..h....
0x04D0: C0 01 8B F4 50 FF 95 90 FE FF FF 3B F4 90 43 4B  ....P......;..CK
0x04E0: 43 4B 89 85 34 FE FF FF EB 2A 8B F4 8B 8D 68 FE  CK..4....*....h.
0x04F0: FF FF 51 8B 95 34 FE FF FF 52 FF 95 70 FE FF FF  ..Q..4...R..p...
0x0500: 3B F4 90 43 4B 43 4B 8B 8D 4C FE FF FF 89 84 8D  ;..CKCK..L......
0x0510: 8C FE FF FF EB 0F 8B 95 68 FE FF FF 83 C2 01 89  ........h.......
0x0520: 95 68 FE FF FF 8B 85 68 FE FF FF 0F BE 08 85 C9  .h.....h........
0x0530: 74 02 EB E2 8B 95 68 FE FF FF 83 C2 01 89 95 68  t.....h........h
0x0540: FE FF FF E9 53 FF FF FF 8B 85 68 FE FF FF 83 C0  ....S.....h.....
0x0550: 01 89 85 68 FE FF FF 8B 4D 08 8B 91 84 00 00 00  ...h....M.......
0x0560: 89 95 6C FE FF FF C7 85 4C FE FF FF 04 00 00 00  ..l.....L.......
0x0570: C6 85 D0 FE FF FF 68 8B 45 08 89 85 D1 FE FF FF  ......h.E.......
0x0580: C7 85 D5 FE FF FF 5B 53 53 FF C7 85 D9 FE FF FF  ......[SS.......
0x0590: 63 78 90 90 8B 4D 08 8B 51 10 89 95 50 FE FF FF  cx...M..Q...P...
0x05A0: 83 BD 50 FE FF FF 00 75 26 8B F4 6A 00 8D 85 4C  ..P....u&..j...L
0x05B0: FE FF FF 50 8B 8D 68 FE FF FF 51 8B 55 08 8B 42  ...P..h...Q.U..B
0x05C0: 08 50 FF 95 6C FE FF FF 3B F4 90 43 4B 43 4B 83  .P..l...;..CKCK.
0x05D0: BD 50 FE FF FF 64 7D 5C 8B 8D 50 FE FF FF 83 C1  .P...d}\..P.....
0x05E0: 01 89 8D 50 FE FF FF 8B 95 50                    ...P.....P
```

Figure 2.11

a) Source of trace:

The capture in figure 2.11 was obtained from http://www.incidents.org/logs/Raw/. A total of 12 packets were captured from 12 distinct source addresses. All 12 packets were destined for the same host. Figure 2.12 is a Whois query from the American Registry for Internet Numbers (ARIN), pertaining to the displayed packet's source address.

Whois Lookup:

```
Rogers Cable Inc. (NET-ROGERS-CAB-8)
   1 Mount Pleasant Road
   Toronto, Ontario M4Y 2Y5
   CA

   Netname: ROGERS-CAB-8
   Netblock: 65.48.0.0 - 65.48.175.255
   Maintainer: RONB

   Coordinator:
      Budd, Paul  (AD30-ARIN)  abuse@rogers.com
      416) 935-4729

   Domain System inverse mapping provided by:

   NS1.WLFDLE.RNC.NET.CABLE.ROGERS.COM 24.153.22.13
   NS2.WLFDLE.RNC.NET.CABLE.ROGERS.COM 24.153.22.14
   NS1.YM.RNC.NET.CABLE.ROGERS.COM 24.153.22.141
   NS2.YM.RNC.NET.CABLE.ROGERS.COM 24.153.22.142

   Record last updated on 24-Jul-2002.
   Database last updated on  25-Jul-2002 20:00:35 EDT.
```

Figure 2.12

b) Detect was generated by:

This detect was acquired from the Incidents.org website. The packet was obtained as a binary TCPDump file and then run through a Snort version 1.8.6 intrusion detection system on a 800MHz Pentium III running Windows 2000. The rule set that generated this detect was current at the time. Figure 2.13 shows the rule used to generate the alert. The single line rule is wrapped for clarity.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS
ISAPI.ida access"; uricontent:".ida"; nocase; flags:A+;
reference:arachnids,552; classtype:web-application-activity;
reference:cve,CAN-2000-0071; reference:bugtraq,1065; sid:1242;  rev:6;)
```

Figure 2.13

The alert was generated based on the following rule matches.

a) Packet contained TCP information
b) The traffic flow was external to internal
c) Destination port TCP 80 was set
d) ACK and PSH flags are set
c) Packet contents matched ".ida"

c) Probability the source address was spoofed:

It is unlikely the source address is spoofed in this case. The attacking system would require a response from the victim to complete the attack.

d) Description of attack:
This detect is part of a Code Red version 1 IIS worm attack against multiple hosts on port 80. Information on Code Red can be found in Cert advisory CA-2001-19. If the host is running IIS 4.0 or 5.0 with indexing services, the Code Red worm will attempt to exploit a buffer overflow, giving full access to the system as discussed in CVE-2001-0500. If port 80 is open, but IIS is not running, a "403 forbidden" error will be issued. Once compromised, a message will be displayed on all requested pages from the server. The worm will then continue to attempt to propagate to random IP addresses. The attack seen here is an attempted compromise and does not always indicate a successful attack.

e) Attack Mechanism:
The Code Red worm will issue a crafted "HTTP Get request" to port 80 on random hosts. Once a connection is made the worm will attempt a buffer overflow to the Indexing Service Application Programming Interface extension (IDQ.DLL). The indexing services do not have to be running; the only condition that must be met is the IIS server (4.0 or 5.0) must be running with script mappings for Internet Data Administration (.ida) and Internet Data Query (.idq) files. Part of the code that handles input URLs in the idq.dll contains an unchecked buffer that would allow the attacker to execute a buffer overrun. This will overwrite the program code with the attackers code, changing the programs operation as determined by the attacker. If the buffer overflow is successful, the message "HELLO! Welcome to http://www.worm.com! Hacked By Chinese!" will be displayed on any page requested from the web server. The worm does not make any attempt to connect to www.worm.com. Other compromises include time-based activity:

Day of Month	Activity
1st – 19th	The worm will continue to propagate to random address
20th – 27th	Code Red will pick an address and attempt a denial of service attack
28th – end of month	No activity

Once the system has been compromised, Code Red will continue to scan random IP addresses searching for other hosts to exploit.

f) Correlations:
The following resources were used for information gathering and to correlate information on the IIS Indexing Service Buffer Overflow / Code Red Worm
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0500
http://www.cert.org/advisories/CA-2001-19.html
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp

g) Evidence of active targeting:
Code Red is a self-propagating worm that chooses its targets randomly. There is no

evidence of active targeting in this case.

h) Severity:
Severity = (criticality + lethality) – (system countermeasures + network countermeasures)
Scale = 1 lowest; 5 highest

Criticality = 3	This attack could have a great impact on the organization's web servers. Although they may not be critical, they do portray an image of the organization. Any compromise could cause the organizations clients to lose faith in the security of their data.
Lethality = 4	Compromised systems would display defaced web pages. The systems may also be responsible for a denial of service flood on other systems possibly outside the organization. Worm propagation would also have continued. This could have a negative impact on the organization.
System Countermeasures = 5	If the server is maintained with current patches and virus protection, along with host based intrusion detection, compromise by this attack should not be a concern. No other packets indicating compromise were detected.
Network Countermeasures = 4	Implementation of a restrictive firewall blocking internal hosts and servers. Network based intrusion detection.

.

Severity = (3 + 4) – (5+4) = -2

i) Defensive Recommendation:

1) Ensure all current patches are installed;
   Windows 2000 Server
   http://www.microsoft.com/Downloads/Release.asp?ReleaseID=30800
   Windows NT 4.0
   http://www.microsoft.com/Downloads/Release.asp?ReleaseID=30833
2) Network / Host based firewalls
3) Updated virus scanners and signature databases

j) Multiple Choice Test Question
Code Red will attack which port on a host?
a) TCP port 8080
b) UDP port 80
c) TCP port 80

d) UDP port 8080

Answer: C
k) Below are comments and my response to the Code Red detect taken from the incidents.org email list.

1) -----Original Message-----
From: Smith, Donald [mailto:Donald.Smith@xxx.xxx]
Sent: Monday, July 29, 2002 7:31 AM
To: 'Tim Smoljanovic'; 'intrusions@incidents.org'
Subject: RE: Post for GCIA v.3.2 practical

DS: In line comments. But first a question. WHICH code red is this?
TS: Version 1 (more detailed answer further down)

DS: Did you examine the logs for signs that the attack was or wasn't successful?

TS: There was nothing in the logs indicating a successful attack. I would have expected to see at least a 403 forbidden message but there was nothing. Your comment got me to thinking a bit more about the response. If IIS was not running but port 80 was open then a 403 would be issued. If the system was attacked and compromised, we would see connection attempts coming from the server. What I'm not sure of is what would be generated on a system running IIS but is patched. Would the packet just be dropped and the only indication of the attempt found in the logs? I will take a closer look at this.

DS: I recommend you list a few things you might see if it were successful.

TS: A partial list was included but as I have discovered it could have been much more detailed such as the increase in processor/network utilization and increased connections.

DS: Also based on my first question which code red was this? There are different things to look for. Later versions of code red didn't replace the web pages.

TS: I based the paper on version 1. V.2 will attempted to install a backdoor (root.exe). Snort will indicate version 2 based on this information. I will include more information on determining the difference between the versions as I have discovered while answering everyone's questions I have indeed missed allot of valuable information.
DS: What did you base these on?
Scale = 1 lowest 5 highest
Criticality = 3
Lethality = 4

TS: As noted at the beginning of the paper I have zero knowledge of the network and system used in this scenario. I based my rating for the Criticality and Lethality on a vulnerable system and the effects it may have. A home user running a web page with pictures of their cat would not be affected the same as a major e-commerce site. For Criticality, I tried for a "Happy Medium". I increased the Lethality to 4 based on the fact that a denial of service could result in legal action.

System Countermeasures = 5
DS:So you assume the server is up to date. If so this number makes sense.

TS: Again this goes back to lack of knowledge. I guess what I was trying to make clear here is that if the

system is patched and secure, code red should be of very little concern.

Network Countermeasures = 4
DS: If the server is internal and if there is a firewall then this makes sense. Can you tell from the logs anything about the ip address under attack?Is it in the same subnet as the rest of the ip addresses being logged? Or on a different subnet?

TS: The most I can determine from the information given to me is they are using 78.37.x.x. Any further subnet or vlan information is unknown. The network countermeasures were difficult to determine. I did base it on the fact that if the server was internal then any external attempts could be blocked at the wall. Other steps could be taken but the list could be extensive depending on bandwidth, money, etc... I considered keeping it simple here.

2) -----Original Message-----
From: kyle.haugsness@xxx.xxx [mailto:kyle.haugsness@xxx.xxx]
Sent: Sunday, July 28, 2002 8:16 PM
To: smoljano@camosun.bc.ca; intrusions@incidents.org
Subject: RE: Post for GCIA v.3.2 practical

KH: Tim,
Here are a couple of comments/questions on your writeup. Overall, excellent job!

1. You mentioned that 12 packets were received from 12 different hosts. Were all of these the same attack?

TS: Yes, all packets were identical with the exception of header information (source IP, time, etc..).

KH: 2. Are there other clues in the packet decode that this is positively the Code Red worm?

TS: The "www.worm.com" located in the payload would be the biggest give away. I should have noted also that Code Red does not attempt a connection to this url.

KH: 3. You explain how to check for a successful attack on the machine itself. How could you identify a successful attack by looking only at a network trace?

TS: Good question, something I never really thought about. Off the top of my head, because the worm will continue to propagate, any packets containing code red using the web server as a source would be a good indication. Other than that, processor and network utilization will increase. Netstat will show connections that should not be there also. I will definitely poke around and try and get more info on this.

3) -----Original Message-----
From: Tod Beardsley [mailto:todb@xxx.xxx]
Sent: Thursday, August 01, 2002 8:37 AM
To: Tim Smoljanovic
Cc: 'intrusions @ incidents . org'
Subject: Re: Post for GCIA v.3.2 practical

TB: Heya Tim -- a solid discussion of Code Red you have here, and no problems with your conclusions that I could see. I only have a couple of tiny suggestions:

a) For some historical context, an analysis of Code Red can't be hurt by mentioning its relationship to Nimda and sadmind. Perhaps a sentence or two describing this?

b) "the Code Red worm will exploit a buffer overflow giving full access to the system as discussed in CVE-2001-0500." Since Code Red is so well-known at a signature and results level, it might be worth going into (briefly, but concisely) how buffer overflows work. If you could pull that off (and not be boring about it), it would definitely make your analysis of Code Red stand out from most others.

TS: The relationship to Nimda and sadmind is a great idea. I will add this to the updated version that will be re-posted in about a week. I'm currently working on more information concerning the overflow. This suggestion seems to be a very popular one and only makes sense.

3. Assignment 3 – Analyze This

3.1 Executive Summary

This analysis was conducted at the request of the SANS GCIA University. The analysis was completed on five consecutive days of Snort alerts and scan data, and five days of Out-of-Specification (OOS) data. The intent of this analysis is to provide insight into any potential compromises or vulnerabilities, and to provide recommendations for improving security at the University.

The following logs were used in the analysis process;

Alert logs:
alert.020720.gz; alert.020721.gz; alert.020722.gz; alert.0207.023.gz; alert.020724.gz

Scan logs:
scan.020720.gz; scan.020721.gz; scan.020722.gz; scan.020723.gz; scan.020724.gz

OOS logs:
oos_Jun.14.2002; oos_Jun.15.2002.gz; oos_Jun.19.2002.gz; oos_Jun.20.2002.gz; oos_Jun.21.2002.gz

The OOS logs were from June as these were the most recent oos logs available at the time this analysis was written. It should also be mentioned that date format on the files is misleading. The dates of the logs are actually the day before the file date. In other words, oos_Jun.14.2002.gz are actually the logs for June 13, 2002. The only exception is oos_Jun.20.2002.gz contains the logs for June 20, but oos_Jun.21.2002.gz contains the logs for June 19, 2002.

Total number of alerts detected was;
Alerts = 1,039,142
Scans = 7,054,867
OOS = 2,947
Total logs = 8,096,956

Section 3.2 deals with the Snort alert logs. Included is a list of all alerts generated and a total count of each alert. I then took the five alerts that I felt required some detailed examination and broke them down. The five alerts were chosen for further analysis as each alert indicates typical traffic seen on the network. The following is a breakdown of each alert and why it was chosen;

1. **Possible Trojan Server Activity**
   This alert is included as Trojan activity can be very common on a network and generate many alerts. This particular University is showing a lot of Trojan activity and should be aware of the many different types of Trojans in use and how to prevent infection.

2. **Connect To 515 From Outside**
   The Connect to 515 from outside is included as it shows how sometimes things are not always what they seem. This alert first indicates that an attempt to connect to

port 515 from outside the network was made. This would seem at first to indicate an attempt to exploit a known vulnerability with the Berkley LPD printing services, but during the analysis process we also discover possible Trojan activity. This indicates that even if the University was not vulnerable to the LPD exploit, the alert should not be ignored as it could be an indicating other activity.

3. TCP SRC and DST outside network

This alert and analysis shows how suspicious traffic could also be the result of misconfigured hosts or network gear. At first it would seem we are dealing with source address spoofing but soon realize that misconfigured hosts generate the majority of alerts. In a large network misconfigured systems and bad IP protocol stacks can be common. This is an example of the type of traffic the analyst may see generated from such errors.

4. IDS452/web-iis_http-iis-unicode-binary

Microsoft's Internet Information Services software is known to have weak security features, especially with a default install. Although not always an indication of malicious activity, this type of alert can be common on a network running web servers and could be a strong indication of possible compromise attempts. Alerts such as this although common should not go unchecked. This analysis is a break down of what a possible attack may look like.

5. MYPARTY - Possible My Party infection

I choose to include this alert, as it is a good indication of a possible false positive. Although My party is a known Trojan, it is date activated and the date has long since passed and therefore activity from this Trojan should not be seen. Although circumstances such as incorrectly set clocks on a system and possible a new variant of the Trojan may indicate hostile activity, this alert most likely is a false positive.

Included in each breakdown is a sample of the alert, a description of the alert, a description of the attack, address information of the attacker, and, finally, any recommendations.

Section 3.3 looks at the alerts generated by the scan logs. We first take a look at a breakdown of the scan alerts; we then take a look at the five top scanners. The breakdown for the five top scanners includes: the break down of the scan; the port-scan summary, including address information; and any recommendations.

Section 3.4 is a breakdown of the OOS logs. First we look at the details of the alerts. Next we do an analysis of one of the alerts, including address information. Finally, we look at any recommendations.

Section 3.5 covers the top ten talkers. The top ten talkers list is based on the amount of alerts and traffic generated. A breakdown is done of the top talkers based on the alerts, scans, and oos files.

Section 3.6 is a pair of link graphs that give insight into the flow of traffic during an attack or anomalous activity. We first take a look at possible Trojan activity within the University network, and then at one specific server that seems to draw a lot of attention.

Section 3.7 gives a brief list of hosts I feel require further investigation. This list is a "critical activities" list put together from detects analyzed, and also includes other hosts that I feel require attention.

Section 3.8 summarizes the defensive measures I believe the University should consider. This is a list accumulated from best practices and also from information gained during the analysis process.

Section 3.9 covers a breakdown of the process I used to complete the analysis. I detail the steps I used to gather, organize, and understand the large amounts of data put before me.

Section 3.10 is the final section and lists the resources used to complete the analysis.

3.2 Alert Summary

Attack	Total
NIMDA - Attempt to execute cmd from campus host	655508
UDP SRC and DST outside network	132338
Watchlist 000220 IL-ISDNNET-990517	64923
spp_http_decode IIS Unicode attack detected	38060
SMB Name Wildcard	37213
Port 55850 tcp - Possible myserver activity - ref. 010313-1	28329
spp_http_decode CGI Null Byte attack detected	22450
TFTP - External UDP connection to internal tftp server	18705
SYN-FIN scan!	14025
External RPC call	8319
Watchlist 000222 NET-NCFC	5340
Possible trojan server activity	2285
IDS452/web-iis_http-iis-unicode-binary	1513
IRC evil - running XDCC	1481
High port 65535 tcp - possible Red Worm - traffic	1238
connect to 515 from outside	1202
Queso fingerprint	1140
EXPLOIT x86 NOOP	947
Attempted Sun RPC high port access	832
Incomplete Packet Fragments Discarded	767
Null scan!	433
SCAN Proxy attempt	370
IDS475/web-iis_web-webdav-propfind	228
High port 65535 udp - possible Red Worm - traffic	227
SNMP public access	202
beetle.ucs	170
Port 55850 udp - Possible myserver activity - ref. 010313-1	94
Tiny Fragments - Possible Hostile Activity	91
SUNRPC highport access!	85
SMTP relaying denied	82
INFO - Possible Squid Scan	70
EXPLOIT x86 setuid 0	67
TFTP - Internal UDP connection to external tftp server	62
IDS305/web-iis_http-iis_translate_f	57
NMAP TCP ping!	45

Attack	Total
EXPLOIT x86 setgid 0	32
STATDX UDP attack	32
SCAN Synscan Portscan ID 19104	28
EXPLOIT x86 stealth noop	19
EXPLOIT x86 NOPS	17
IDS552/web-iis_IIS ISAPI Overflow ida nosize	16
SCAN FIN	14
TCP SRC and DST outside network	10
BACKDOOR NetMetro Incoming Traffic	9
External FTP to HelpDesk 130.85.70.50	8
External FTP to HelpDesk 130.85.70.49	7
Watchlist 000219	5
IDS50/trojan_trojan-active-subseven	5
HelpDesk 130.85.70.50 to External FTP	4
FTP DoS ftpd globbing	4
MYPARTY - Possible My Party infection	4
ICMP SRC and DST outside network	3
RFB - Possible WinVNC - 010708-1	3
TFTP - External TCP connection to internal tftp server	2
HelpDesk 130.85.83.197 to External FTP	2
Probable NMAP fingerprint attempt	2
SMB CD...	2
SMTP chameleon overflow	2
Back Orifice	2
HelpDesk 130.85.70.49 to External FTP	2
SCAN - wayboard request - allows reading of arbitrary files as http service	1
SCAN - webspirs request - allows reading of arbitrary files as http service	1
EXPLOIT solaris NOOP	1
SCAN - sendtemp request - allows reading of arbitrary files as http service	1
SCAN - palscgi request - allows reading of arbitrary files as http service	1
SCAN - commerce request - allows reading of arbitrary files as http service	1
SCAN XMAS	1
IDS535/web-iis_http-iis5-printer-beavuh	1
STATDX TCP attack	1
External FTP to HelpDesk 130.85.83.197	1

Table 3.0

3.2.1 Five Detects

This section will deal with 5 detects that I feel require a more detailed analysis. This section will include information on the attack and the source address that triggered the alert. Also included is a breakdown of information concerning the top attacking IP for each alert. Each alert will close with defensive recommendations.

l) Possible Trojan Server Activity

a) Alert Sample

07/20-23:46:21.523960 [**] Possible trojan server activity [**] My.Host.83.6:27374 -> 12.18.91.229:3981

07/20-23:46:22.142287 [**] Possible trojan server activity [**] 12.18.91.229:3981 -> My.Host.83.6:27374

07/20-23:46:22.142713 [**] Possible trojan server activity [**] My.Host.83.6:27374 -> 12.18.91.229:3981

Figure 3.0

b) Alert Description

07/20-23:46:21.523960	Month, day, time
[**] Possible trojan server activity [**]	Alert type
My.Host.83.6:27374	Source address and port
->	Traffic flow indicator
12.18.91.229:3981	Destination address and port

Table 3.1

This alert was generated by a Snort Intrusion Detection System. The version of Snort, and hardware/software platform are unknown. I was unable to find a rule that would generate this type of alert in the current Snort rule set, so I believe this rule is custom to the University. Based on the alerts generated, it could be assumed the rule is triggered on a match of the source or destination port 27374. The following list shows the number of alerts per day.

DATE	Number of Alerts
July 20	605
July 21	548
July 22	37
July 23	121
July 24	974
Total	2285

Table 3.2

c) Attack Description

This alert is indicating possible Trojan activity. A Trojan is best described as a program hidden inside another program or file. A Trojan is usually malicious and is intended to give the attacker access to the victim's system, or use the victim's system for malicious purposes.

The alert generated by Snort could have been caused by a number of known Trojans that use port 27374 for communications. We will take a look at a couple of the more popular Trojans.

i) SubSeven version 2.1 and higher:

SubSeven is a backdoor program that allows remote attackers to take control of a system. SubSeven is aimed towards Microsoft Windows Platforms. The earlier versions of SubSeven used port 1243. With version 2.1 and higher a default port of 27374 is set but is configurable by the user. When SubSeven is running on a remote computer the attacker will have the ability to;

- Restart the remote computer
- Open the CD-ROM drive
- Set up an ftp server
- Execute programs
- Access files
- Scan other systems
- Edit registry information

There are many other options available depending on the version of SubSeven in use. Currently, I am aware of 19 versions of SubSeven available on the Internet.

ii) Ramen Worm:

Ramen is a worm that affects Linux versions 6.2 and 7. The worm uses a tool known as synscan to scan randomly generated Ips, and try and determine if and what version of Red Hat Linux is being used. The worm will then attempt to exploit a vulnerable rpc.statd or wuftpd service in Red Hat 6.2, or LPRng bug for version 7. Once access is gained, the worm will open a HTTP service on port 27374 and download a copy of itself to the victim machine. Once the worm is on the system it will replace the index.html to show the following message, along with a picture of a bag of noodles.

RameN Crew
Hackers looooooooooooooooove noodles.™

iii) Lion Worm:

The Lion worm takes advantage of well known vulnerabilities in BIND (for more information on the vulnerabilities see CERT Advisory CA-2001-02). Lion will scan randomly generated class "B" network spaces for TCP port 53 Domain Name Server (DNS). Once a system running port 53 is found, the worm will check to see

if the system is vulnerable and, if so, will run the BIND exploit and install a rootkit. The worm will then email huckit@china.com with the contents of /etc/passwd, /etc/shadow and the network settings of the infected system. The Lion worm will open and listen for any request on Port 27374 and continue to scan other systems for potential victims.

A "false positive" is caused when an alert is triggered indicting possible hostile activity, when actually the traffic is legitimate. In the case of this detect, a false positive will be triggered if port 27374 is used as an ephemeral port. This may be the cause of many of these alerts. It should also be noted that, in total, 609 scans were detected to and from port 27374 and may also have been responsible for triggering the alert.

d) Address Information

There were a total of 1240 external addresses detected as the source for the alert. Of these, the address 209.122.242.117 accounted for the most alerts (totaling 110). There were also 85 scans with 209.122.242.117 as a source address and port 27374 as the destination port. The following information was obtained by a Whois lookup from the American Registry for Internet Numbers, pertaining to 209.122.242.117:

```
RCN Corporation (NET-RCN-BLK-3)
   105 Carnegie Center
   Princeton, NJ 08540
   US

   Netname: RCN-BLK-3
   Netblock: 209.122.0.0 - 209.122.255.255
   Maintainer: RCN

   Coordinator:
      RCN Corporation  (ZR40-ARIN)  noc@rcn.com
      888-972-6622

   Domain System inverse mapping provided by:

   AUTH1.DNS.RCN.NET          207.172.3.20
   AUTH2.DNS.RCN.NET          206.138.112.20
   AUTH3.DNS.RCN.NET          207.172.3.21
   AUTH4.DNS.RCN.NET          207.172.3.22

   Record last updated on 04-Apr-2001.
   Database last updated on  2-Aug-2002 20:00:14 EDT.
```

Figure 3.1

e) Recommendations

This section dealt with the alert "Possible Trojan Server Activity". It has been determined that the majority of alerts can be associated to a scan on port 27374, or as a "false positive" with port 27374 being used as an ephemeral port. There was very little activity generated internally that could be related to these alerts. An indication of successful compromise for the listed Trojans would be internal hosts scanning for port 27374, or port 53 from 27374. The only concerns are as follows:

There was an alert generated indicating My.Host.157.2 and My.Host.104.2 as possibly being compromised by SubSeven. The following alert was detected in the logs:

07/20-03:12:41.897883[**]IDS50/rojan_rojan-active-subseven[**]
130.85.157.239:1243 -> 161.58.182.181:45664

Although this is a SubSeven detect, it uses the port 1243, indicating it may be a different version than the ones we are currently looking at. It is recommended however, that both these hosts be verified.

My.Host.83.6 shows port 27374 as a source port generating 15 alerts and 16 alerts as a destination port with six distinct destination addresses involved.
My.Net.83.9 shows port 27374 as a source port generating 16 alerts and 14 alerts as a destination port. There were a total of six distinct addresses involved.

The scan logs only indicated My.Host.70.207 and My.Host.82.2 as possibly compromised. Both addresses show a single scan to a single external host. The external hosts were also distinct addresses.

Although it is not possible to determine from this alert if, in fact, these hosts have been compromised, it would be recommended that all six hosts be verified. Defensive recommendations include ensuring host/email based virus protection, blocking of port 27374 on the firewall or router, and education for end users on the dangers of possible Trojans in email attachments.

II) Connect To 515 From Outside

a) Alert Sample

07/20-12:56:05.404192 [**]connect to 515 from outside[**] 255.255.255.255:31337->My.Host.134.71:515

07/20-15:23:16.533683 [**]connect to 515 from outside[**] 66.32.235.166:3711-> My.Host.85.70.198:515

Figure 3.2

b) Alert Description

07/20-12:56:05.404192	Month, day, time
[**] connect to 515 from outside [**]	Alert type
66.32.235.166:3711	Source address and port
->	Traffic flow indicator
My.Host.70.198:515	Destination address and port

Table 3.3

As with the last attack, I am unable to locate a Snort rule for this alert. It is believed that this alert is generated by a custom rule written by the University. Port 515 is associated with the Line Printer Demon (LPD). This alert is indicating an external connection to port 515 on the internal host.

DATE	Number of Alerts
July 20	8
July 21	45
July 22	1145
July 23	3
July 24	1
Total	1202

Table 3.4

c) Attack Description

There is a known vulnerability in the Berkeley lpr printing service LPRng version 3.6.25 and earlier. This is a format string attack that takes advantage of a flaw in the syslog() wrapper. A successful compromise will overwrite addresses in the lpd process address space, allowing the attacker to execute arbitrary code. This will allow the attack to install backdoors and rootkits, obtain access to the password files, and manipulate system files and logs. More detailed information can be found in Cert Advisory CA-2000-22.html. While viewing the alerts I came across a few that confused me. The alerts were from address 255.255.255.255 with a source port of 31337. The packets were destined for port 515 and triggered the "connect to 515 from outside" alert. IP address 255.255.255.255 is the broadcast address and would not be used as a host address. Port 31337 is a very well-known port for the Trojan BackOrifice. Tyler Schacht submitted a GCIA assignment on August 16, 2001 that explained what I was seeing. Tyler describes this attack as a reporting feature used by BackOrifice. An attacker looking for BackOrifice infected systems will try a half-open connect to an address using the source address of 255.255.255.255, the destination address's network broadcast address. When the destination address replies to 255.255.255.255 port 31337, any BackOrifice infected systems on the same broadcast domain will also receive the reply. The infected system will then join a pre-defined IRC channel and display information informing the attacker of its presence. The attacker is able to discover compromised systems while remaining completely anonymous.

d) Address Information

There were a total of six distinct addresses, including the broadcast address triggering this alert. The following table summarizes the totals for each address.

Address	Number of Alerts
24.123.46.10	799
64.30.217.125	340
62.150.48.122	45
255.255.255.255	10
66.1.1.121	5
66.32.235.166	3

Table 3.5

The following information was obtained by a Whois lookup from the American Registry for Internet Numbers, pertaining to the address with the most alerts. (24.123.46.10):

```
ROADRUNNER-COMMERCIAL-CENTRAL (NETBLK-RR-COMMERCIAL-CENTRAL)
   13241 Woodland Park Road
   Herndon, VA 20171
   US

   Netname: RR-COMMERCIAL-CENTRAL
   Netblock: 24.123.0.0 - 24.123.255.255
   Maintainer: RCCT

   Coordinator:
      ServiceCo LLC  (ZS30-ARIN)  abuse@rr.com
      1-703-345-3416

   Domain System inverse mapping provided by:
   NS1.BIZ.RR.COM            24.30.200.19
   NS2.BIZ.RR.COM            24.30.201.19
   DNS4.RR.COM               65.24.0.172

   ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

   Record last updated on 09-Apr-2002.
   Database last updated on  3-Aug-2002 20:00:01 EDT.
```

Figure 3.3

e) Recommendations

There were no responses to the port 515 alerts detected. There were a total of 926 scans with port 515 as the destination port detected. The attempts from 255.255.255.255 also did not generate any detected responses.

During the analysis of this attack, a possible BackOrifice compromise was also noticed. Two alerts were generated for this attack. The following information relates to this activity.

```
07/22-09:07:24.012965 [**] Back Orifice [**] 66.129.222.70:39849->My.Host.162.226:31337
07/22-09:07:24.199539 [**] Back Orifice [**] 66.129.222.70:39849->My.Host.162.226:31337
```

Figure 3.4

This could be an indication of a BackOrifice compromise on the host having the address My.Host.162.226 and should be verified.

Defensive recommendations include blocking port 515 at the firewall from external access, if it is not needed. Also, a filter to block port 31337 should also be put into place. Patches for the port 515 LPR exploit are also available from the operating systems' vendors and should also be applied and updated.

III) TCP SRC and DST outside network

a) Alert Sample

(Alerts wrapped for clarity)

07/23-15:40:27.008932 [**] TCP SRC and DST outside network [**]
169.254.50.54:1861 -> 169.254.106.13:139

07/23-15:40:27.009129 [**] TCP SRC and DST outside network [**]
169.254.50.54:1861 -> 169.254.106.13:139

Figure 3.5

b) Alert Description

This again was an alert that I was unable to identify a Snort rule for. This alert is indicating that both the source and destination IP address in the packet are originating from outside the University network.

07/23-15:40:27.008932	Month, day, time
[**] TCP SRC and DST outside network [**]	Alert type
169.254.50.54:1861	Source address and port
->	Traffic flow indicator
169.254.106.13:139	Destination address and port

Table 3.6

c) Attack Description

There were a total of 10 alerts of this type. These alerts are indicating that the source and destination address are not internal to the University's network. The following table lists the addresses detected.

Source IP	Source Port	Destination IP	Destination Port	Total
3.0.0.0	2678	3.0.0.0	6346	1
192.168.5.2	2786	216.254.108.22	59938	1
192.168.0.16	36157	212.84.209.27	6667	1
169.254.50.5	1861	169.254.106.13	139	6
192.168.1.101	50076	204.62.44.167	80	1

Table 3.7

The source address of all the alerts indicates mainly mis-configured internal systems. The alert for address 3.0.0.0 is intriguing. This address represents "this network" or the all 0s address for the class A address space 3.0.0.0. The port the packet was destined for is listed as gnutella-svc. Gnutella is a well know "peer to peer" (p2p) file-sharing network. A Whois search turns up the following information:

```
General Electric Company (NET-GE-INTERNET)
   1 Independence Way
   Princeton, NJ 08540
   US

   Netname: GE-INTERNET
```

```
   Netblock: 3.0.0.0 - 3.255.255.255

   Coordinator:
      General Electric Company   (GET2-ORG-ARIN)   GENICTech@GE.COM
      518-612-6672
   Record last updated on 12-Nov-1998.
   Database last updated on  3-Aug-2002 20:00:01 EDT.
```

Figure 3.6

The second alert uses a source IP that is part of a class B private address range, or "non-routable". This address should not be able to pass through a router to the outside of an organization's autonomous system. The destination address of 216.254.108.22 is a legitimate address. A search on the destination port did not turn up any useful information. A Whois search turns up the following information on this address.

```
RIO MOTOR SPORTS, INC (NETBLK-SPEK-272665-0)
   25 Broadway
   New York, NY 10004
   US
 Netname: SPEK-272665-0
   Netblock: 216.254.108.16 - 216.254.108.31

   Coordinator:
      Stollar, Andreas   (AS3414-ARIN)   abuse@speakeasy.net
      +1-206-728-9770 (FAX) (206)728-1500
   Record last updated on 09-Nov-2001.
   Database last updated on  3-Aug-2002 20:00:01 EDT.
```

Figure 3.7

The third packet also uses a private source address and a public destination address. The destination port of 6667 is a well known IRC port. A Whois search turns up the following information for the address 212.84.209.27;

```
inetnum:        212.84.209.0 - 212.84.209.63
netname:        ASH-NET
descr:          Hauke Johannknecht
country:        DE
admin-c:        HJ422-RIPE
tech-c:         HJ422-RIPE
status:         ASSIGNED PA
mnt-by:         TRMD-MNT
changed:        cplieth@transmedia.de 19991011
source:         RIPE
route:          212.84.192.0/18
descr:          TRANSMEDIA GmbH
origin:         AS9132
mnt-by:         MEDIASCAPE-MNT
changed:        cm@mediascape.de 20010112
source:         RIPE
```

Figure 3.8

The next address is also odd, as both the source and destination addresses are reserved by the Internet Assigned Numbers Authority (IANA). The destination port of 139 is the NETBIOS Session Service. Microsoft Windows will assign this address in the event the host cannot connect to a DHCP server and obtain a legitimate University address.

The final packet also has a source address that is private and a public destination address. The destination port is set for port 80 HTTP. It would seem port 80 is open on this system, yet I was denied access to the page. The destination IP of 204.62.44.167 is registered to:

```
Towson State University (NETBLK-TSU)
   Academic Computing Service
   Towson, MD 21204
   US

   Netname: TSU
   Netblock: 204.62.32.0 - 204.62.51.255

   Coordinator:
      Houston, Samuel  (SH1243-ARIN)  shouston@BACH.TOWSON.EDU
      (410) 830-4084 (FAX) (410) 830-2661

   Domain System inverse mapping provided by:
   HAL.TOWSON.EDU             204.62.32.10
   TRANTOR.UMD.EDU            128.8.10.14

   Record last updated on 07-May-1996.
   Database last updated on  3-Aug-2002 20:00:01 EDT.
```

Figure 3.9

I have come to the conclusion that not all of the alerts are related yet I believe all packets originated from within the University network. I was also able to find a scan alert with the source address of My.Host.81.27 and a destination address of 3.0.0.0. This scan was detected two seconds after the alert was detected. The port involved in the scan was UDP port 137 for both the source and destination. Certain features of Gnutella may describe what is happening here.

Gnutella has a feature that allows a user to spoof his IP address that is being advertised to the Gnutella network. This, in turn, will cause other users of the network to try and access the spoofed address's system for downloadable files. If the spoofed address were to advertise a very popular file, many users would attempt to access the spoofed address, and, in turn, create large amounts of traffic to the actual owner of the spoofed address, which could possibly cause a denial of service. Spoofing both the source and destination addresses would make it very difficult to track down the individual responsible. With respect to the detected scan to 3.0.0.0, I believe the scan could have one of two meanings. This could simply be someone internal preparing to attempt to access the spoofed address to retrieve a file or, it is the individual who is spoofing the 3.0.0.0 address verifying if a denial of service attack was successful. The only other scenario I can see is a "Land attack". A "Land attack" is when an attacker spoofs the source address of a packet to that of the destination. Some implementations of TCP will send the packet into an infinite loop, causing the system to crash.

I believe there may be a couple of answers to the private address in the 192.168.x.x range we are seeing. I understand the users of the residential network are able to set

up network address translation (NAT) and Proxy servers. These alerts may be caused by a broken NAT server that is allowing the private address to escape without being translated. The other option is a rogue DHCP server issuing private address. The scan log has also picked up many of these same addresses, most of which are destination addresses for internal addresses accessing Gnutella.

The address 169.254.50.5 reserved by IANA is also an address Microsoft Windows will issue if a DHCP client is unable to obtain an address via the DHCP server. I believe what we are seeing here is a system that was unable to obtain an address trying to establish a connection to a second system, also using the 169 address.

d) Address Information
Address information is included with the attack description for this alert.

e) Recommendations
There is no hard evidence to support the information I have provided with respect to this alert. However, with limited knowledge about the network layout for the University and not knowing the location and traffic being analyzed by the Snort sensor, I believe the conclusions reached here would best describe the alerts. There are a couple of steps to prevent this type of traffic. First, IP spoofing must be blocked on both the ingress port and egress port on the routers. This would prevent all traffic, such as that seen here, from escaping the local network. Verification of DHCP, NAT, and Proxy servers for correct functionality would also be advised and may clear up a lot of this type of traffic.

IV) IDS452/web-iis_http-iis-unicode-binary

a) Trace Sample
(Alerts wrapped for clarity)

```
07/20-00:04:53.160337 [**] IDS452/web-iis_http-iis-unicode-binary [**]
166.114.127.6:38169 -> My.Host.111.221:80

07/20-00:04:53.167209 [**]spp_http_decode IIS Unicode attack detected [**]
166.114.127.6:38170 -> My.Host.111.221:80
```

Figure 3.10

b) Alert Description
The following information is a sample Snort rule that may trigger this alert. Following the rule is a breakdown of the alert generated by Snort. The rule has been wrapped for clarity.

```
alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: "IDS452/web-iis_http-iis-
unicode-binary"; flags: A+; uricontent: "..|c0af|"; nocase; classtype:
system-attempt; reference: arachnids,452;)
```

Figure 3.11

07/20-00:04:53.167209	Month, day, time
[**]spp_http_decode IIS Unicode attack detected [**]	Alert type
166.114.127.6:38170	Source address and port
->	Traffic flow indicator
My.Host.111.221:80	Destination address and port

Table 3.8

This alert is generated when matches are made with the Snort rule. In this case, the destination port is 80, the ACK flag, plus one other, is set and a contents match of c0af was met.

c) Attack Description

The purpose of this attack is to take advantage of a known exploit in the default install of Microsoft's IIS version 4.0. The attacker will send the binary character \xc0 instead of %c0. This Unicode content replaces the typical / or \ characters with the ../ character, allowing a directory or folder traversal. If successful in the exploit, the attacker will be able to upload a backdoor to the web server, or execute arbitrary commands and access the file system.

d) Address Information

There were a total of 107 addresses attempting this exploit. Of these, 48 were internal hosts, and 59 were external hosts. The external address 80.135.50.45 had the highest number of alerts, with 454 detects to a total of 429 destination addresses on port 80. There were also 7,872 hosts scanned by this address. The alert "spp_http_decode IIS Unicode attack detected" was also triggered 430 times from this same address. The following information was obtained by a Whois lookup from the American Registry for Internet Numbers, pertaining to this host:

```
inetnum:        80.128.0.0 - 80.146.159.255
netname:        DTAG-DIAL16
descr:          Deutsche Telekom AG
country:        DE
admin-c:        DTIP-RIPE
tech-c:         ST5359-RIPE
status:         ASSIGNED PA
notify:         auftrag@nic.telekom.de
notify:         dbd@nic.dtag.de
mnt-by:         DTAG-NIC
changed:        auftrag@nic.telekom.de 20020108
source:         RIPE
route:          80.128.0.0/11
descr:          Deutsche Telekom AG, Internet service provider
origin:         AS3320
mnt-by:         DTAG-RR
changed:        bp@nic.dtag.de 20010807
source:         RIPE
```

Figure 3.12

e) Recommendations

The directory traversal style of attack against IIS servers is very popular. The attacks seen here are most likely automated scripts attempting the exploit at any system they find with port 80 open. The system My.Net.70.69 was the only host that replied to 80.135.50.45. It is recommended that this system be verified for any compromise. In general, all IIS servers should be verified for the most recent security patches and service packs. I would also recommend host-based intrusion detection systems or running the server with VMware so you can easily roll back to a "known good" system.

V) MYPARTY - Possible My Party infection

a) Trace Sample

07/24-19:51:22.214661 [**] MYPARTY - Possible My Party infection [**]
My.Host.109.47:1495 -> 209.151.250.170:80

07/24-19:51:22.215703 [**] MYPARTY - Possible My Party infection [**]
My.Host.109.47:1495 -> 209.151.250.170:80

Figure 3.13

b) Alert Description

I was unable to locate a Snort rule for this alert. This attack is outdated and unlikely to cause an alert. I believe it has been removed from the Snort rule set. In this case the rule would have triggered on the IP address 209.151.250.170.

07/24-19:51:22.214661	Month, day, time
[**] MYPARTY - Possible My Party infection [**]	Alert type
My.Host.109.47:1495	Source address and port
->	Traffic flow indicator
209.151.250.170:80	Destination address and port

Table 3.9

c) Attack Description

The "My Party" worm is a good example of a Trojan worm that also takes advantage of social engineering. This worm is spread via email and contains the contents:

Subject: new photos from my party!
Message:
Hello!

My party... It was absolutely amazing!
I have attached my web page with new photos!
If you can please make color prints of my photos. Thanks!

The email message contains an attachment that misleads the reader to believe it is a short cut to a web site. Unsuspecting users may be curious and open the attachment to view the pictures. Once the attachment is opened, the virus will become active and

install a Trojan known as BackDoor-FB.svr.gen. This Trojan will install a backdoor and send a message to the author informing him of the compromise. The back door will then try to connect to http://209.151.250.170/ in order to download the files necessary to run the backdoor. This Trojan spreads by mailing itself out to all the recipients in the victim's address book and inbox. The Trojan will only mass mail on January 25 – 29, 2001. There is another variant that will follow the same procedure but only email between January 20 – 24, 2001. After these dates, the virus will not attempt to propagate from the infected host, but the backdoor will remain open.

d) Address Information

The address involved in this alert are My.Host.109.47 and 209.151.250.170:80. The destination address is the same address that the Trojan will contact in the event of infection. The following is a Whois search on the destination address;

```
Cyberverse Online (NETBLK-CYBERVERSE)
   2221 Rosecrans Avenue Suite 130
   El Segundo, CA  90245
   US

   Netname: CYBERVERSE
   Netblock: 209.151.224.0 - 209.151.255.255
   Maintainer: CYBO

   Coordinator:
      Cyberverse Online Hostmaster  (COH3-ORG-ARIN)  domain@CYBERVERSE.COM
      (310) 643-3783
Fax- (310) 643-3794
Domain System inverse mapping provided by:

   NS1.CYBERVERSE.NET        209.151.224.62
   NS2.CYBERVERSE.NET        209.151.232.62
   NS3.CYBERVERSE.NET        38.185.152.49

   ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

   Record last updated on 27-Sep-1999.
   Database last updated on  4-Aug-2002 20:00:00 EDT.
```

Figure 3.14

e) Recommendations

It would appear an internal host accessing a particular IP address known to be used by a Trojan triggered this alert. I was unable to get a response from the site on port 80. The worm will only mail itself out at the end of January 2001, so it is unlikely the worm is spreading through the network. It is possible that the system My.Host.109.47 has an incorrectly set clock, and may have triggered the virus if it was on the system. Although it is a high probability that this is a false positive, I would recommend verifying this host. Installing virus scanners on hosts and keeping the signature database up-to-date can reduce infection from Trojans, viruses, and worms. Email systems should be running a virus-check front end such as Antigen for Exchange Servers. Education for users on the dangers of opening, downloading, or executing files is also recommended.

3.3 Scan Summary

Total scans detected	7054867
Total distinct internal addresses scanning as source	1983
Total distinct external address scanning as source	487

Table 3.10

3.3.1 Top Five External Address

This section will deal with the top five external source addresses scanning the University Network. A breakdown of the scan will be given along with information about the IP. Each detect will be completed with any defensive recommendations.

I) 205.188.228.X

This first scan involves 5 addresses in total. All addresses are registered with America Online Inc. The destination ports were identical for each address, and scan totals were also very similar. All scans used the UDP protocol. The following table lists the address involved, number of internal hosts scanned, ports scanned, and total scans per port;

Address	Destinations	Ports	Total Scans
205.188.228.129	59	6970	75947
		6972	1068
205.188.228.65	59	6970	73387
		6972	266
205.188.228.1	59	6970	71949
		6972	850
205.188.228.3	57	6970	71548
		6972	636
205.188.228.17	54	6970	60447
		6972	500

Table 3.11

Port / Scan Summary

Real Time Transport Protocol/Real Time Streaming Protocol (RTP/RTSP) uses UDP

ports 6970 – 7170. When a program such as Real Player or Quicktime is run, the audio/video stream uses ports 6970 through 7170 as the connection ports on the client. The following information was obtained by a Whois lookup from the American Registry for Internet Numbers pertaining to the source IP:

```
America Online, Inc (NETBLK-AOL-DTC)
       22080 Pacific Blvd
       Sterling, VA 20166
       US

       Netname: AOL-DTC
       Netblock: 205.188.0.0 - 205.188.255.255

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

       Domain System inverse mapping provided by:
       DNS-01.NS.AOL.COM            152.163.159.232
       DNS-02.NS.AOL.COM            205.188.157.232

       Record last updated on 27-Apr-1998.
       Database last updated on  1-Aug-2002 20:00:38 EDT.
```

Figure 3.15

Recommendations

This activity would appear not to be a scan. The activity points to internal University individuals accessing streaming video/audio using Real Player. If this type of activity is unacceptable, it is suggested that a policy be put in place regarding streaming audio/video and a filter, or ACL, denying access to these ports on the firewall or router.

II) 24.187.87.107

Total Scans Detected = 31002
Total Hosts Scanned = 7330
The following table indicates which ports were scanned, total scans, protocol / flags used, and port assignments:

Port	Total Scans	Protocol / Flags	Assignment
1090	1	UDP	FF Fieldbus Message Specification
137	1798	UDP	NETBIOS Name Service
139	14571	TCP / SYN	NETBIOS Session Service
445	14632	TCP / SYN	Microsoft-DS

Table 3.12

Port / Scan Summary

This scan was directed at NETBIOS services on the internal hosts. Ports 137 and 139 are used with Windows NT to run Server Message Block (SMB) on top of NETBIOS over TCP/IP (NBT) for file sharing. With the onset of Windows 2000, port 445 is used

to run SMB directly over TCP/IP without the NBT layer.

A search of the alert logs turned up 3030 alerts for SMB Name Wildcard. This could be generated by a couple of possible attacks.

a) Scan and attempted access to unprotected file shares on host systems

b) Network.vbs script. This script will attempt a connection, also to an unprotected file share, and access the c:\network.log. If successful, the script will generate random IP addresses and attempt to mount the c:\ drive on the remote system as j:\ on the local system. More information can be obtained from Cert Incident Note IN-2000-02.

There were also two alerts generated with regards to port 137 on My.Host.70.69 . The alert was “beetle.ucs”. I was unable to find any reference for this alert in the Snort rules, indicating a custom rule, so I did a search on the Internet. I was able to find a FAQ, making reference to a host system used for burning CDs. Beetle.ucs was part of the hostname of this system. An ICMP echo request to the host confirmed that this system was indeed the system being referenced by the Beetle.ucs alert. I believe this is generated when an external address tries to access My.Host.70.69.

The scan to UDP port 1090 seems to be a bit out of place from the rest of the scans generated by 24.187.87.107. This scan was destined for My.Host.150.198. This port seems to have a couple of well-known functions. The port is also used by the Extreme Trojan, and also by Real One Player, streaming media proxy. The following information was obtained by a Whois lookup from the American Registry for Internet Numbers, pertaining to the source IP;

```
Cablevision Systems Corp (NETBLK-OOL-104CORMNY1-0110)
   111 New South Road
   Hicksville, NY 11801
   US

   Netname: OOL-104CORMNY1-0110
   Netblock: 24.187.80.0 - 24.187.95.255

   Coordinator:
      OOL Hostmaster  (OH4-ORG-ARIN)  hostmaster@CV.NET
      (516)393-3281

   Record last updated on 01-Nov-2001.
Database last updated on  1-Aug-2002 20:00:38 EDT.
```

Figure 3.16

Recommendations

Although the activity seen here is malicious, I am unable to find any indication that compromise has been successful. There were no SMB Wildcard alerts generated from internal systems to external systems. The attempted access to port 1090 did not generate any subsequent alert from either the destination or source address indicating any connection to the port 1090 attack. I am unable to detect any alerts from

My.Host.150.198 that may be related to this attack, but would suggest the system be analyzed, as I am not aware of what would cause this rule to trigger. It is also recommended that the web page indicating the use of this system be removed from public viewing, as it gives attackers information such as a host and domain name, along with the intended use of the system. This type of listing is an invitation for an attacker to attempt to compromise the host. It is recommended that if no external access to ports 137, 139, 445, or 1090 are required, then these ports be blocked at the firewall. It is also recommended that print and file sharing be turned off on all hosts not requiring its use, and strong passwords be used on hosts requiring print and file sharing services.

III) 80.135.50.45
Total Scans Detected = 28704
Total Hosts Scanned = 7872
Ports Scanned = Port 80, HTTP Services
Protocol / Flags = TCP / SYN

Port / Scan Summary
This scan seems to indicate a typical half-open scan to port 80 on internal hosts. This would indicate that the attacker is doing reconnaissance for possible web servers. The alert file shows some alert activity from this source. The alerts generated are listed below.

a) beetle.ucs
b) IDS452/web-iis_http-iis-unicode-binary
c) spp_http_decode IIS Unicode attack detected

The beetle.ucs alert seems to indicate access to a CD writing station on port 80. This is a custom rule, so it is unknown what triggered the alert. The IDS462/web-iis alert indicates the attacker is sending the binary character \xc0 instead of "%c0". This would allow the attacker to execute arbitrary commands on a web server running the default install of IIS 4.0 or 5.0. The spp_http_decode IIS alert indicates an attempt to get a directory listing from a web server. This is the first step in an attempt to upload a backdoor or deface a web site. The following information was obtained by a Whois lookup from the RIPE database, pertaining to the source IP:

```
inetnum:        80.128.0.0 - 80.146.159.255
netname:        DTAG-DIAL16
descr:          Deutsche Telekom AG
country:        DE
admin-c:        DTIP-RIPE
tech-c:         ST5359-RIPE
status:         ASSIGNED PA
notify:         auftrag@nic.telekom.de
notify:         dbd@nic.dtag.de
mnt-by:         DTAG-NIC
changed:        auftrag@nic.telekom.de 20020108
source:         RIPE
```

Figure 3.17

Recommendations
There were six alerts of “beetle.ucs” generated as the source address from My.Host.70.69 on port 80 with a destination address of 80.135.50.45. There were also six requests from 80.135.50.45 to port 80 on My.Host.70.69. This is a good indication that port 80 is active on My.Host.70.69. Due to the lack of knowledge pertaining to this alert, it is unknown exactly what transpired between the two systems. As indicated earlier, it is recommended this system be verified for any compromise. This scan was a simple SYN scan to port 80. There really is nothing one can do to prevent scanning of public servers. It is recommended, based on the alerts generated from this address that all systems running IIS 4.0 or 5.0 be verified for recent patches and secure installs. The servers should also be checked for compromise. No other alerts or indications of compromises were generated from this IP beyond what was discussed here.

IV) 213.93.159.116
Total Scans Detected = 22043
Total Hosts Scanned = 7867
Ports Scanned = Port 80, HTTP Services
Protocol / Flags = TCP / SYN

Port / Scan Summary
This scan is almost identical to the previously discussed scan. Once again, we are seeing a half-open SYN scan to port 80 on multiple hosts. The same alerts have also been generated within the alert logs. The alerts generated are listed below.

a) beetle.ucs, Total alerts = 3
b) IDS452/web-iis_http-iis-unicode-binary; Total alerts = 283
c) spp_http_decode IIS Unicode attack detected; Total alerts = 282

Again, we are seeing My.Host.70.69 respond to requests on port 80, indicating port 80 as open on this host. The other two alerts are identical to the last detect analyzed. The IDS462/web-iis alert indicates the attacker is sending the binary character \xc0 instead of "%c0". This would allow the attacker to execute arbitrary commands on a web server running the default install of IIS 4.0 or 5.0. The spp_http_decode IIS alert indicates an attempt to get a directory listing from a web server. This is the first step in an attempt to upload a backdoor or deface a web site.
The following information was obtained by a Whois lookup from the RIPE database pertaining to the source IP;

```
inetnum:        213.93.158.0 - 213.93.159.255
netname:        UPC-KT-CABLE30-31
descr:          Chello Com21
country:        NL
admin-c:        LG40-RIPE
tech-c:         RC482-RIPE
tech-c:         HMCB1-RIPE
status:         ASSIGNED PA
notify:         hostmaster@chello.at
```

```
mnt-by:         CHELLO-MNT
changed:        hostmaster@chello.at 20020711
source:         RIPE
```

Figure 3.18

Recommendations

The recommendations are identical to that of the previous detect. In summary, all web servers running IIS 4.0 and 5.0 should be verified for the latest patch and service pack installs. No servers should be installed with default settings. My.Host.70.69 should be verified for all patches and checked for compromises, and the FAQ regarding this host should be removed from public access on the University's web site.

V) 216.63.246.166

Total Scans Detected = 12552
Total Hosts Scanned = 7703
Ports Scanned = Port 1433, Microsoft SQL Server
Protocol / Flags = TCP / SYN

Port / Scan Summary

This scan is a half-open TCP SYN scan to port 1433 looking for Microsoft SQL Servers. The recently discovered Spida Worm could have generated this. This worm takes advantage of weak or nonexistent "sa" passwords in Microsoft SQL. If a successful connection is made from a previously compromised system, the worm will attempt to use the xp_cmdshell utility to enable and set a password for the guest user. If this step is successful, the worm will make the guest user a part of the Local and domain admin groups, copy itself to the victim's system and disable the guest account. It will then set the sa password to that of the guest account and begin running on the victim's system. The worm will email a copy of the local password database, the network configuration and SQL server configuration to ixtld@postone.com. At this point the worm will then continue to scan for more systems to infect. More information on this worm can be found in Cert Incident Note IN-2002-04. The following information was obtained by a Whois lookup from the American Registry for Internet Numbers, pertaining to the source IP:

```
Society of Exploration Geophysicists (NETBLK-SBCIS31663)
   2701 W. 15th St.
   PMB 236
   Plano, TX 75075
   US
   Netname: SBCIS31663
   Netblock: 216.63.246.0 - 216.63.247.255

   Coordinator:
      Southwestern Bell Internet Services  (ZS44-ARIN)  ipadmin@swbell.net
      888-212-5411

   Record last updated on 10-Aug-1999.
   Database last updated on  2-Aug-2002 20:00:14 EDT.
```

Figure 3.19

Recommendations

There were no alerts generated by this scan, although it must be noted that My.Host.70.207 and My.Host.82.2, have both generated large scans with destination port 1433 to random IP addresses. I would recommend that both these systems be checked for the Spida Worm or any other compromises. Defense against this attack is fairly easy: simply set a strong sa password. Also, the email address ixtld@postone.com should be blocked and logged. Any logging of this email address coming from internal addresses is a sure sign of compromise.

3.4 Out-of-Specification (OOS)

Out-of-Specification packets are packets that do not meet specifications listed in RFC 793 Transmission Control Protocol and RFC 791 Internet Protocol. An attacker can craft these types of packets for malicious purposes, or they may be a result of a broken protocol stack or bad transmission media.

3.4.1 OOS Alerts

The following information gives a breakdown of the OOS files over the five-day period.
Total Packets Captured = 2946
Total Internal IP as Source = 0
Total Distinct External Address = 110
Total Distinct Internal Address = 62

The following table shows a breakdown of the different flag sets and total count

Flag	Count
SF**	1
SF*U	1
SF*P	1
**SF*PAU	1
SFRU	1
**SFR*AU	1
SFRP	1
**SFRP*U	1
**SFRPAU	2
*1SF****	1
*1SF**A*	1
*1SF**AU	1
*1SF*P**	2
*1SF*P*U	4
*1SFR***	1
*1SFR*AU	1
*1SFRPAU	1
2*SF****	2

Flag	Count
2*SF**A*	1
2*SF**AU	1
2*SF*P**	1
2*SF*PA*	1
2*SFR**U	3
2*SFR*A*	2
2*SFRPA*	3
21**R***	2
21**R**U	4
21**R*AU	2
21**RP*U	4
21*F****	1
21*F*P*U	1
21*F*PA*	1
21*FR***	3
21*FR*A*	1
21*FRPAU	3
21S*****	2866

Flag	Count
21S***A*	3
21S***AU	3
21S**P**	1
21S**P*U	1
21S**PA*	1
21S**PAU	1
21S*R***	1
21S*R*AU	1
21S*RPA*	1
21SF**A*	1
21SF*PA*	1
21SF*PAU	1
21SFR**U	2
21SFR*A*	1
21SFR*AU	1
21SFRPAU	1

Table 3.13

3.4.2 Packet Analysis

In this section we will take our top talker from the OOS alerts and analyze the activity. In total, there were 1,287 alerts generated for this host. All packets were directed at a single internal host. The following is a portion of the packets captured:

```
06/18-13:52:58.741859 68.32.126.64:13369 -> MY.NET.6.7:110
TCP TTL:47 TOS:0x0 ID:10510  DF
21S***** Seq: 0xC5B9F5DC   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 39574738 0 EOL EOL EOL EOL

6/19-00:09:36.271897 68.32.126.64:29222 -> MY.NET.6.7:110
TCP TTL:47 TOS:0x0 ID:47994  DF
21S***** Seq: 0xDDCFEBD0   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 43274410 0 EOL EOL EOL EOL

06/19-00:10:39.665718 68.32.126.64:29248 -> MY.NET.6.7:110
TCP TTL:47 TOS:0x0 ID:20405  DF
21S***** Seq: 0xE1DB26DB   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 43280750 0 EOL EOL EOL EOL


06/19-00:11:43.120772 68.32.126.64:29277 -> MY.NET.6.7:110
TCP TTL:47 TOS:0x0 ID:47970  DF
21S***** Seq: 0xE5D5395A   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 43287096 0 EOL EOL EOL EOL


06/20-11:37:50.157484 68.32.126.64:27187 -> MY.NET.6.7:110
TCP TTL:48 TOS:0x0 ID:54888  DF
21S***** Seq: 0x4258FE68   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 56043520 0 EOL EOL EOL EOL
```

Figure 3.20

All packets have a destination port of 110, the post office protocol (pop). The packets were first detected at 13:52 on the 18th of June. This is the earliest my logs date back to. I do not have the logs for the 16th or 17th, so it is possible this pattern started earlier. These packets range in time from a few seconds to a few minutes apart. The pattern is continuous to the end of my logs so it is possible this continued long after the 20th. Also each packet is a SYN packet. This is the first packet in a three-way handshake. For each of these packets received by My.Net.6.7, a SYN/ACK would be sent in response. The remote system would then send an ACK to complete the process or a RST to terminate it. If no RST or ACK is sent, TCP will hold the half-open session in a buffer for a pre-determined amount of time selected by the OS. The buffer can only hold a certain amount of half-open sessions. When the amount of requests rises above the threshold, the server will refuse any more connections. What we may be seeing here is a denial of service attack against port 110 of a mail server. The attacker is sending continuous requests and keeping the buffer full, causing legitimate traffic to be turned away.
The following is a Whois query from the American Registry for Internet Numbers (ARIN), pertaining to the displayed packet's source address.

```
Comcast Cable Communications, Inc. (NETBLK-JUMPSTART-BALTIMOR-A3)
   3 Executive Campus
   Cherry Hill, NJ 08002
   US

   Netname: JUMPSTART-BALTIMOR-A3
   Netblock: 68.32.112.0 - 68.32.143.255

   Coordinator:
      Comcast Cable Communications, Inc.  (IC161-ARIN)  cips-ip-
registration@cable.comcast.com
      856-317-7300

   Record last updated on 15-Jun-2002.
   Database last updated on  4-Aug-2002 20:00:00 EDT.
```

Figure 3.21

Recommendations

With an event such as this, the system under attack would either be running very slow or would be frozen. No connections would be possible to the server. This would be a good indication of what is happening, and would confirm this analysis. The source address should be blocked at the router or firewall, and a complaint registered with the user's service provider.

3.5 Top Ten Talkers Summary

This section will summarize the top ten talkers by amount of alerts generated over the five day period. This section is broken down into three sections: alerts, scans, and OOS.

3.5.) Alerts

Top Ten Alerts

Alert	Count	Description
NIMDA - Attempt to execute cmd from campus host	655508	Possible Nimda worm infection
UDP SRC and DST outside network	132338	External source and destination address
Watchlist 000220 IL-ISDNNET-990517	64923	Watches for traffic from a network in Israel
spp_http_decode IIS Unicode attack detected	38060	Directory traversal attempt
SMB Name Wildcard	37213	NETBIOS reconnaissance
Port 55850 tcp - Possible myserver activity - ref. 010313-1	28329	Denial of service Trojan
spp_http_decode CGI Null Byte attack detected	22450	Directory traversal attempt
TFTP - External UDP connection to internal tftp server	18705	External connection to trivial file transfer server

SYN-FIN scan!	14025	Scan technique to evade IDS and Firewalls
External RPC call	8319	Remote procedure call outside local network

Top Ten Source Addresses

Address	Count
My.Host.157.247	337075
My.Host.157.246	264690
63.250.213.12	100513
212.179.35.118	59491
My.Host.117.27	53387
3.0.0.99	26393
My.Host.70.149	14938
212.244.34.30	14022
134.68.32.18	13126
216.106.80.130	11159

Top Ten Destination Addresses

Address	Count
233.28.65.148	100500
My.Host.86.108	57162
10.0.0.1	26391
192.168.0.216	18683
216.241.219.28	16183
134.68.32.18	14931
My.Host.70.149	13223
My.Host.99.174	4432
152.163.210.84	3858
My.Host.163.107	2961

3.5.2 Scans

Top Ten Source Addresses Internal

Address	Count
My.Host.70.207	1694756
My.Host.82.2	1251693
My.Host.114.37	1164453
My.Host.86.108	412633
My.Host.157.247	338250
My.Host.157.246	273503
My.Host.83.153	174595
My.Host.70.101	114625
My.Host.6.40	113484
My.Host.116.104	50467

Top Ten Source Addresses External

Address	Count
205.188.228.129	77015
205.188.228.65	73653
205.188.228.1	72799
205.188.228.33	72184
205.188.228.17	60947
24.187.87.107	31002
80.135.50.45	28704
213.93.159.116	22043
216.63.246.166	12552
134.192.84.43	11197

Top Ten Destination Addresses Internal

Address	Count
My.Host.153.45	28601

Top Ten Destination Addresses External

Address	Count
My.Host.184.23	13007

My.Host.70.111	12897
My.Host.178.41	12863
My.Host.106.78	12595
My.Host.114.46	12455
My.Host.150.120	12171
My.Host.158.25	12134
My.Host.151.72	11891
My.Host.71.243	11185

192.50.75.16	202957
68.64.225.59	21568
216.254.108.19	19814
68.81.150.8	19807
24.170.50.191	19335
216.254.108.22	16276
66.220.21.18	14180
193.237.187.18	12788
65.185.150.97	12016
194.251.249.103	10772

Top Ten Destination Ports

Port	Count	Description
1214	1853470	Kazza
12203	1672990	Metal of Honor Server
80	865104	HTTP
6970	353549	RealAudio
6257	182551	WinMX File Sharing
25	109979	SMTP
21	65706	FTP Control
1433	58927	Microsoft SQL Server
0	45760	Reserved
1025	37512	network blackjack

3.5.3 Out-of-Specification

Top Ten Source Addresses

Address	Count

Top Ten Destination Addresses

Address	Count

68.32.126.64	1287
209.116.70.75	447
62.76.241.129	385
65.210.154.210	135
212.35.180.17	83
128.241.21.30	68
213.250.44.19	59
209.132.232.101	43
202.178.132.185	35
65.214.43.159	31
MY.NET.6.7	1308
MY.NET.97.217	281
MY.NET.100.217	212
MY.NET.111.198	156
MY.NET.97.238	104
MY.NET.253.125	88
MY.NET.253.20	86
MY.NET.100.165	80
MY.NET.6.40	66
MY.NET.111.140	66

Top Ten Flags

Flag	Count
21S*****	2866
21**R**U	4
*1SF*P*U	4
21**RP*U	4
2*SFR**U	3
2*SFRPA*	3
21*FR***	3
21*FRPAU	3
21S***A*	3
21S***AU	3

Top Ten Destination Ports

Port	Count
110	1287
25	597
113	411
80	302
4662	157
21	76
6346	34
6347	7
1269	5
888	3

3.6 Link Graph

3.6.1 Trojan Activity

This first link graph is intended to aid in understanding the flow of possible Trojan activity. All internal hosts depicted in this graph should be examined for possible compromises.

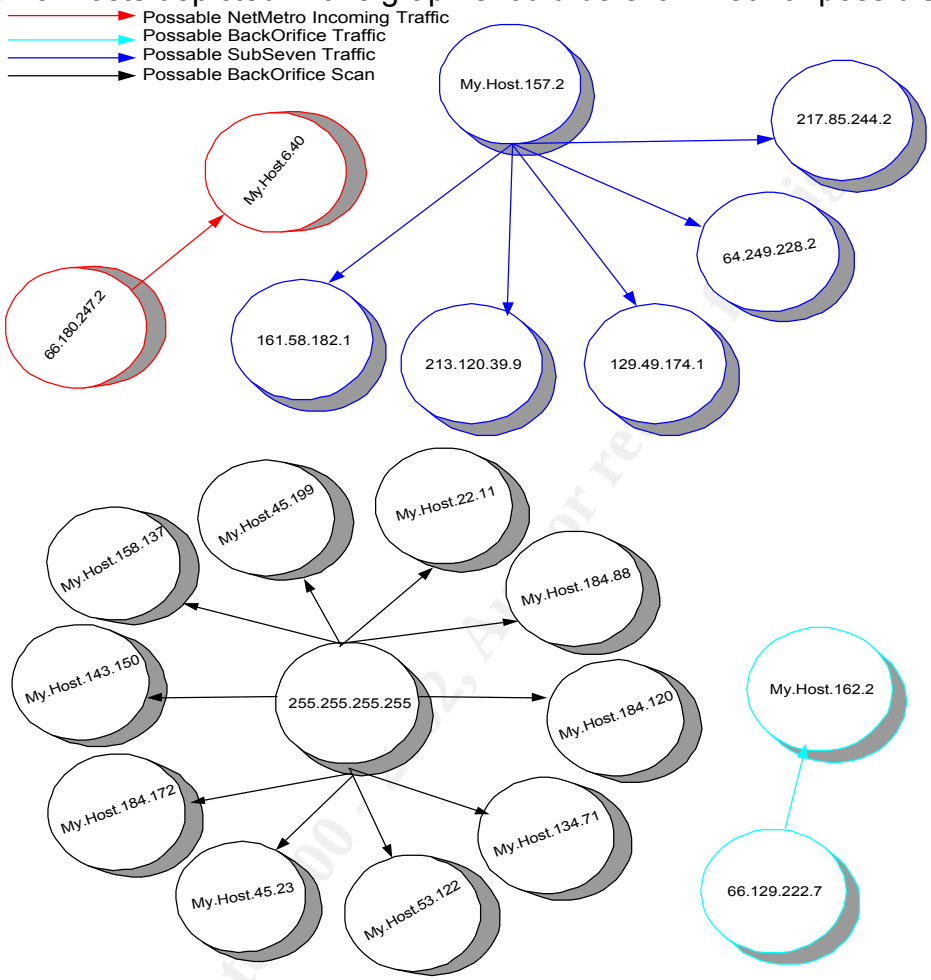

Figure 3.22

3.6.2 Beetle.ucs

The second link graph is in response to the activity taking place against My.Host.70.69, which seems to be a high profile server. The rule beetle.ucs seems to have been written to indicate access to this host. The rule seems to be in place to give an indication of when the server is being accessed, but does not give any indication of traffic content to and from the server. The intent of this link graph is to give the reader a clearer understanding of the alert patterns detected to and from My.Host.70.69 over the five-day period. This graph represents the type of alert, alert count, and traffic flow.

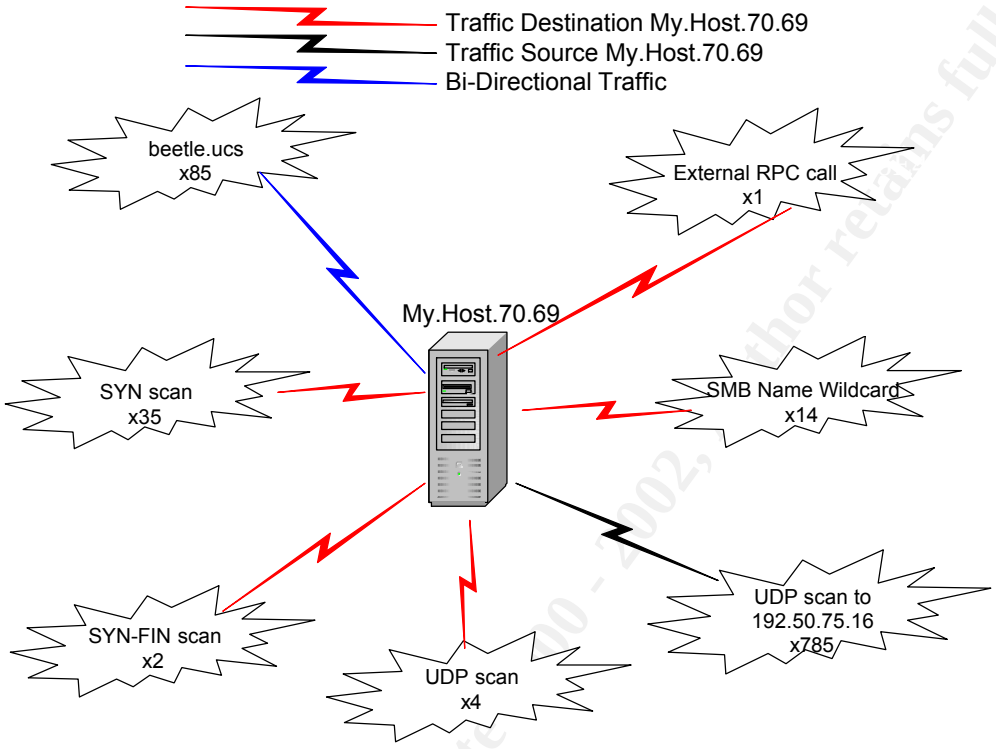

Figure 3.23

3.7 Summary of Critical Activity

This section will give a brief summary of the hosts that require further investigation. It is recommended that all hosts listed here be verified.

i) NIMDA - Attempt to execute cmd from campus host

There were a total of 655508 alerts for the Nimda worm. The following hosts should be verified for compromise by the Nimda worm:

My.Host.100.59	My.Host.117.27
My.Host.157.246	My.Host.157.247
My.Host.53.46	My.Host.84.176

ii) Other Compromises

The following hosts show signs of Trojan activity and compromises. All of these systems should be verified immediately:

My.Host.157.2 –	SubSeven infection	My.Host.162.226 –	BackOrifice infection
My.Host.83.6 –	SubSeven infection	My.Host.247.2 –	NetMetro infection
My.Host.70.207 –	SubSeven infection	My.Host.83.9 –	SubSeven infection
My.Host.82.2 –	SubSeven infection		

The following hosts need to be verified in response to attacks that may indicate a compromise:

My.Host.109.47 - My party worm
My.Host.70.69 – Should be verified for most recent IIS service packs, patches and any indication of possible compromises.

3.8 Summary of Defensive Recommendations

Educational Institutions may be one of the most difficult organizations to secure. When securing a University or College, management and the security/network administrators must keep in mind that what may seem as malicious or non-productive traffic, may be necessary for the purpose of education. Many may feel their right to "freedom of education" has been placed in jeopardy simply by denying access to a particular web site or service. This must be kept in mind while implementing any defensive measures.

1) Core Routers should be set to deny all spoofed addresses on both the ingress and egress ports. This means no internal address should be arriving on the ingress port as a source address. No external address should be arriving on the egress port as a source address.

2) A firewall should be set up and the administrators fully trained on the functionality and user interface of the system.

3) Host based firewalls and virus protection should be installed on all critical systems and virus protection on all hosts.

4) Monitor intrusion detection system for rules that trigger large amounts of false positives. Tailor the rule set to reduce the false positives, this will allow easier identification of legitimate attacks.

5) Perform regular audits and penetration testing on the University network and servers to identify and secure any potential weak points.

6) Have a strong security and acceptable use policy in place. Ensure the policy is strongly enforced or it will not be taken seriously. Although a security and acceptable use policy will not prevent all unwanted traffic originating from within the University it will help to reduce the amount of occurrences.

7) Filter p2p ports, if this type of activity is frowned on, and well known Trojan ports to help reduce the number of Trojan compromises and reduce unwanted traffic.

3.9 Analysis Process

I began the analysis process by deciding what method and format would make the task easier and more effective. I decided on a layout for the report that separated the type of alerts. I found this to make the report much easier to read and comprehend. The next step was to download the log files. Recent OOS files were not available, so I decided to leave them until closer to the end to see if more recent ones would be posted. Once I had downloaded my five days of scan and alert files, I began to open and view them to get an idea of what I was dealing with. I began the process of parsing the data with Microsoft Word, but soon realized the program did not have the ability to deal with such large files. I moved the files over to a Red Hat Linux box where I was able to use the grep command to begin filtering the data. I began by separating the port scan alerts into separate files to help reduce the file size. At this point, I began to look at using the sed command to set delimiters in the files but with limited knowledge of Unix I found it easier to bring the files back into a Windows environment. At this point I began searching for a program that would handle the large files. I found a program called EmEditor from www.BHS.com that was able to set the delimiters in the log files quickly. Once the files were edited, I created a table in MS Access and moved the files into a database. I further parsed the files by creating separate tables based on the alerts. Once the tables were created, I was able to create queries and sort information giving me a better understanding of what I was seeing. Beyond using queries and comparisons, I also used MS Excel to sort and manipulate some of the more detailed information I was dealing with. When I was ready, I checked again for more recent OOS files but the earliest available were from the end of June 2002. I downloaded the 5 most recent files from this time period. The files were small enough that I was able to manipulate them with EmEditor and put them into a database with little effort. Most of the research required was done on the web and by referencing other GCIA student practicals.

3.10 Correlations

The following websites were used to obtain information:

http://www.cert.org/incident_notes/IN-2000-02.html
http://www.cert.org/incident_notes/IN-2002-04.html
http://www.sans.org/newlook/resources/IDFAQ/subseven.htm
http://www.symantec.com/avcenter/venc/data/backdoor.subseven.html
http://www.cert.org/advisories/CA-2001-02.html
http://www.sans.org/newlook/alerts/port515.htm
http://www.cert.org/advisories/CA-2000-22.html
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0884
http://www.sans.org/
www.incidents.org
http://ww1.arin.net/whois/
http://www.ripe.net/ripencc/pub-services/db/whois/whois.html
http://www.snort.org
http://bhs.com

The following student practicals were used as a reference;

0408 Tyler Schacht - http://www.giac.org/practical/Tyler_Schacht_GCIA.doc
0489 Hee So - http://www.giac.org/practical/Hee_So_GCIA.doc