Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 3.5

Jeremy Scott
SANS Rocky Mountain 2004
Denver, CO, USA
Submitted: Sept. 20, 2004

Table of Contents

Assignment #1: Describe the State of Intrusion Detection

Using ManHunt on High-Speed Networks

Introduction

Network intrusion detection systems (NIDS) are a means to gather network traffic and compare it to a set of signatures to determine if an intrusion has taken place. In today's networks, NIDS is now commonplace to detect malicious activity. NIDS are particularly needed in the corporate realm of networks that make up much of the Internet today. However, with the ever-changing pace of today's technology, intrusion detection is having a hard time keeping up. One can look at the modern network of today and see that it is changing rapidly and along with it the speed is increasing. 100Mbps is no longer the standard for enterprise networks. 1Gbps and faster are quickly replacing the older 100Mbps standard. The vulnerability within the network and to NIDS exists with the increase in the number of bits traveling across a network at any given time. Not only is keeping up with the speed a problem, but the increased amount of traffic capable of being transmitted at this speed creates an increase in the amount of data that a NIDS must process in order to determine if an attack has happened. Place into the equation that most networks today are switched creates another problem in itself. It is said well by Jim Hurst in an article posted at SANS, "As networks evolve, NIDS vendors must offer relevant solutions or be left behind."[1]

Vulnerable?

There are many possible solutions to the Gigabit problem that is now crippling the management of NIDS on the network, but the solution is unique to each individual one. One must weigh the costs of implementing the protection that is needed on their network. Are you vulnerable? What risk can your organization take if your intrusion detection is not up to par with the rest of your network? If your company is like mine, the risk that can be taken is minimal or none at all. Vulnerability management is one approach to associate the risks of your network. This approach is not just the method of attaching vulnerability scanners to your network to see what systems are vulnerable but assessing as to the real and perceived threats that your network may have.[2] Vulnerability management is beyond the scope of this paper, but should be taken in to account when deciding the vulnerability of the assets that need to be protected with intrusion detection systems. An article in Information Security Magazine titled "Feeling Vulnerable?" explains the concept and key questions you should ask yourself to assess the risks.[3]

---

[1] Hurst, Jim
[2] Balayan, Misael
[3] Berg, Al

What method do I use?

In looking at a possible solution, the detection methods utilized by the various products now available for intrusion detection must be reviewed. At one time, it was easy to determine the best possible methods because most all NIDS used what is called pattern matching based on a set of known signatures. Now, the choice is a lot harder because many companies that provide NIDS solutions are using various techniques such as: protocol decoding, heuristic and anomaly-based analysis or a mixture of them all.

Let's review the different methods of detection available provided by NSS, an independent organization that does network and security testing:[4]

- Pattern Matching – The most basic form of analysis. Each individual packet is analyzed against a predefined signature to base detection. While this method will allow you to tune the signatures to reduce the inspection that needs to be done, it can also be to specific resulting in numerous false positives.

- Stateful Pattern Matching – This is an enhanced form of pattern matching. Stateful means that it will take in the TCP stream to match patterns rather than to the individual packet. This method is good for evasive techniques such as fragmented packets but requires more resources to track open sessions and reassemble the streams.

- Protocol Decode – Protocol Decode is an approach comparing how the protocol should behave according to how the protocol is defined by the RFC. If a protocol is well defined and properly adhered to, of course, then detection could require less inspection of the packet for detection. This is also known as Protocol Anomaly Detection.

- Heuristic Analysis – This approach uses algorithms to base detection. For example, a ping sweep could be detected by a set threshold number of SYN packets destined for a specific port. The algorithm would have to be tailored for each unique network based on the network's traffic, and could possibly result in numerous false positives.

- Anomaly Analysis – This method analyzes the traffic on a network to determine what is to be considered "normal". Once the NIDS determines what "normal" should look like, detection is based on what is outside "normal" traffic. Changes in traffic could potentially be seen as an intrusion resulting in false positives or worse, "normal" traffic could be a constant flow of malicious traffic on the network.

[4] NSS

Determining which method to use is not an easy task. One must look at the network that they are employed to protect and decide what risks can be taken. As I said about my network, minimal risk or none at all, we have looked at the solution of using some of the different methods discussed in our transition to a more modern detection approach.

The solution, possibly?

Let's look at a possibility that we have considered within my own company. The Gigabit solution we have looked at and are currently in the process of implementing is the Symantec ManHunt™ 3.0 R2. The ManHunt™ node is connected to the network by a Gigabit network tap to maximize bandwidth and reduce the possibility that an attack could go un-noticed by the use of a span port, as addressed in a paper by Brian Laing.[5]

Symantec states that:

> "ManHunt provides high-speed, network intrusion detection, real-time analysis and correlation, and proactive prevention and response to protect enterprise networks against internal and external intrusions and denial-of-service attacks. The ability to detect unknown threats, using protocol anomaly detection, helps eliminates network exposure and the vulnerability inherent in signature-based intrusion detection products. Symantec ManHunt™ traffic rate monitoring capability allows for detection of stealth scans and denial-of-service attacks that can cripple even the most sophisticated networks."[6]

Symantec ManHunt™ provides reliable intrusion detection through a hybrid detection architecture known as "Hybrid Mode." ManHunt uses an array of detection methodologies to enhance attack identification and to collect evidence about malicious activities. This is done by the use of signature detection, protocol anomaly detection, traffic rate monitoring, protocol state tracking, and IP packet re-assembly.

Symantec ManHunt™ combines network intrusion detection and incident or event management capabilities in one product. Symantec calls this a "three-pronged approach" that includes:

- Detection – ManHunt detects a variety of threats by using a layered detection model. This model includes traffic protocol anomaly and stateful and custom signature detection and recognition.
    - Protocol anomaly detection – ManHunt inspects network traffic and compares it to structured protocols. ManHunt compares and notes deviations in network protocol exchanges. ManHunt covers numerous common protocols that are used on the network.

[5] Laing, Brian
[6] Symantec

    - Stateful and custom signature detection – ManHunt can detect a combination of signatures providing a powerful detection capability with the protocol anomaly detection. The stateful signatures are provided by updates from Symantec. The custom signatures allow the user to create signatures to detect traffic according to rules defined by the user.
    - Traffic Monitoring – ManHunt provides high-speed monitoring to detect malicious traffic flow and then tracking that traffic back to the source. Traffic that it detects is DoS attacks and Scan attempts.
    - External Event Dispatch Protocol (EDP) – ManHunt has the ability to use third-party sources such as firewalls, IDS sensors and host-based IDS devices to further correlate data for an attack. The data collected by these sources will be converted and transmitted to the ManHunt node.

- Analysis – ManHunt provides basic provides basic event grouping based on heuristic functions to judge similarities in events by several criteria such as time, type, location, source, and destination.
    - Aggregation – ManHunt reduces traffic load by noting but not logging every packet in an identical attack. For example, if a DDoS attack occurs, ManHunt may aggregate the multiple DDoS packets into one event and show an aggregate packet count.
    - Correlation – ManHunt correlates events to form incidents. Correlation can be based on time, source/destination IP address, or other matching criteria.

- Response – Response policies are a collection of rules and the specific actions taken in response to the events. The response policies can be configured to respond automatically to contain and respond to an intrusion.
    - Policy application – ManHunt compares each event against configurable match parameters. If a match occurs, ManHunt executes the specified action. One rule is processed at a time. ManHunt can be configured to chain parameters or stop after the first match.
    - Automated response – ManHunt uses an automated policy-based response system that includes alerting, session traffic recording, flow tracing, session resetting, and QoS ACL suggestion. ManHunt can be configured for multiple responses to one event and the order in which to respond based on multiple criteria.

Without going into great detail about the technical aspects of the way ManHunt handles detection, ManHunt has proven to be a reliable solution based on performance testing. NSS recently conducted testing on several different Gigabit solutions that are available commercially in the Gigabit IDS Group Test. Results indicated, ManHunt was able to detect “real-world” traffic at a maximum of 400,000 packets per second at 1000Mbps at 100% with zero packet loss and at 100% CPU utilization was still able to provide 620Mbps throughput, this occurred

while ManHunt was running in “Hybrid Mode” with approximately 85 custom signatures loaded.[7]

An intrusion detection solution should not be solely based on the capability to provide reliable Gigabit speed detection, although, it should be the most important part. However, a solution should also provide the user with ease of installation and administration.

Symantec ManHunt™ is relatively easy to install with the provided documentation and a little Linux background. Symantec ManHunt™ 3.0 R2 currently supports Solaris 8 and 9 and Redhat Enterprise Linux 3.0 operating systems. The install on an Intel® server consists of a default install of Redhat Enterprise Linux. The configuration changes that need to be done at the time of the Redhat install is to disable telnet and rlogin. The documentation also says to install SSH, but open-source sshd is done by default. After the OS has been installed, ManHunt™ can be installed simply by entering `./install.sh` and following the provided documentation. Hardening of the operating system should be done after ManHunt™ has been installed.

Although the installation is relatively easy, a complete and reliable knowledge of the network is recommended. During the configuration of the ManHunt node, several top-level objects are needed to ensure that traffic is effectively monitored. These top-level objects include:

- Locations – Any physical or logical grouping of network segments.
- Network devices – Any device in the network such as routers, switches, hubs, and interfaces (monitoring and monitored).
- Symantec ManHunt™ nodes – All nodes placed on the network.
- External sensors – ManHunt can accept input from several third-party external sensors and various other Symantec products.
- Network borders – The autonomous system that connects your network.
- Managed network segments – ManHunt can automatically create a managed network segment object for each unique subnet.

Once the ManHunt node has been set up, then it is time to install the ManHunt console. The console can be run on a Solaris 8 or 9, Redhat Enterprise Linux, or Windows 2000/XP workstation. The only requirement to run the console is that the workstation have Java Runtime Environment (JRE) 1.4 installed. The installation will provide the correct version of JRE if it needed. The ManHunt console provides centralized management interface of all existing nodes. It is used to administer all updates and user and system configurations, as well as the ability to monitor all activity being logged by any of the nodes on the network.[8]

[7] NSS
[8] WhiteHat, Inc.

The Symantec ManHunt™ console enables encrypted and authenticated communication over a propriety protocol and maintains a secure SSH session. It contains two main tabs that provide a view of Monitored Devices and Incidents. The Monitored Devices tab provides a tree-oriented view of the network topology, with a detail summary for each device. The Incident tab provides detailed descriptions of incidents and events taking place in the monitored network, and includes a drilled-down environment for multiple detail levels. In this drilled-down environment, you are not able to see the actual signature like so many of the vendor solutions but you can see what the event actually triggered on.

The console includes reporting using dynamic chart and graph generation, again, with drill-down and data retrieval. Several pre-defined reports include view and print options and can be saved in PDF or HTML format to send through email. The reporting feature can be defined through report scheduling to automatically generate selected reports on a pre-determined time.

The Symantec ManHunt™ solution is so feature rich that a paper could be written to address each different element of this solution in great detail; however, the intent of this paper is to provide a possible solution for intrusion detection on high-speed networks. What we are looking at is one possible solution, out of several. The Symantec ManHunt™ Intrusion Detection System that can be used by itself or along side any other possible solution. Within my company, we have looked at utilizing the signature-based solution of Snort™ with ManHunt to take some of the signature-based pattern-matching load off of the ManHunt node.

Conclusion

While Symantec ManHunt™ proves to be a viable solution for intrusion detection; it is not the only one and may not fit everyone's needs or budget. Intrusion detection on high-speed networks is not cheap or easy, but it can be done. Regardless of the solution that you choose, careful thought should go into the decision making process. Take into account the vulnerabilities that exist and the risks associated with each one. Determine what method or methods are best for your environment. There are several solutions available and each should be considered on an individual basis according to your needs. Plan out the network and proper placement of each sensor so that monitoring and reliable detection can be achieved. If you over do it or think it may be overkill, don't worry about it. You can never be too safe.

References

Bayalan, Miseal. “Intrusion Management.” SANS.org. URL: http://www.giac.org/practical/GCIA/Misael_Balayan_GCIA.pdf. (Feb. 28, 2003)

Berg, Al. “Feeling Vulnerable?” Information Security Magazine. Feb. 2002. URL: http://infosecuritymag.techtarget.com/2002/feb/features_vulnerable.shtml.

Hurst, Jim. “What are some emerging options for NIDS?” SANS.org. URL: http://www.sans.org/resources/idfaq/emerg_nids.php.

Laing, Brian. “How do you implement IDS (network based) in a heavily switched environment?” SANS.org. URL: http://www.sans.org/resources/idfaq/switched.php.

NSS. “Gigabit IDS Group Test – Edition 2.” Aug. 2003. URL: http://www.nss.co.uk/download_form.htm. (Aug. 15, 2003)

Symantec Enterprise Solutions. “Symantec ManHunt.” URL: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=156.

Symantec Security Services. “Symantec ManHunt 3.0 Setup, Application, and Maintenance.” Student Guide. Symantec Education. August 4, 2003.

WhiteHat, Inc. “Symantec ManHunt.” URL: http://www.whitehatinc.com/symantec/manhunt/.

Assignment #2 – Network Detects

Network Detect 1: Backdoor Q access

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] BACKDOOR Q access [**]
10/24/02-18:34:41.216507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
255.255.255.255:31337 -> 32.245.249.236:515 TCP TTL:15 TOS:0x0 ID:0 IpLen:20
DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
63 6B 6F                                          cko

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] BACKDOOR Q access [**]
10/24/02-18:48:23.276507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
255.255.255.255:31337 -> 32.245.133.244:515 TCP TTL:15 TOS:0x0 ID:0 IpLen:20
DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
63 6B 6F                                          cko

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

1. Source of Trace:

The raw log was taken from the GIAC Certification Practical logs at random: http://www.incidents.org/logs/RAW/2002.9.30. The IDS is on the 32.245.0.0/16 network. The true IP addresses are obviously obscured because of the bad checksums in the IP and TCP headers.

2. Detect was generated by:

This detect was generated by myself. I was actually just setting up a computer to run tcpdump and snort for testing purposes and ran this log through tcpdump in standard dump format to look at the packets and determine the host IP address.

```
tcpdump –r ../logs/2002.9.24 –nevXS
```

I saw several packets coming from the IP address 255.255.255.255, all attempting to access port 515. Noticing that the packet also used the hacker's 31337 port as the source port, I decided to run the log back through using a filter to extract only the data I wanted to see.

```
tcpdump –r ../logs/2002.9.24 –nevXS 'src host 255.255.255.255 and dst
port 515'
```

Now, let's look at an example of the packets that I thought were suspicious.

```
18:34:41.216507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4, length
60: IP (tos 0x0, ttl  15, id 0, offset 0, flags [none], length: 43, bad cksum
7bd4 (->90ec)!) 255.255.255.255.31337 > 32.245.249.236.515: R [bad tcp cksum
30fc (->4614)!] 0:3(3) ack 0 win 0 [RST cko]
0x0000   4500 002b 0000 0000 0f06 7bd4 ffff ffff        E..+......{.....
0x0010   20f5 f9ec 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 30fc 0000 636b 6f00 0000             P...0...cko...

18:48:23.276507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4, length
60: IP (tos 0x0, ttl  15, id 0, offset 0, flags [none], length: 43, bad cksum
efcc (->4e5)!) 255.255.255.255.31337 > 32.245.133.244.515: R [bad tcp cksum
a4f4 (->ba0c)!] 0:3(3) ack 0 win 0 [RST cko]
0x0000   4500 002b 0000 0000 0f06 efcc ffff ffff        E..+............
0x0010   20f5 85f4 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 a4f4 0000 636b 6f00 0000             P.......cko...
```

To alleviate my suspicion, I then ran the log back through Snort to see what alerts were detected using the following command:

```
snort -c ../etc/snort.conf -r ../logs/2002.9.24 -h 32.245.0.0/16 -k
none -deyv
```

3. Probability the source address was spoofed:

The source address in this series of packets is most likely spoofed. There are several reasons for the assumption. Let's look at a sample packet again for analysis.


```
18:34:41.216507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4, length
60: IP (tos 0x0, ttl  15, id 0, offset 0, flags [none], length: 43, bad cksum
7bd4 (->90ec)!) 255.255.255.255.31337 > 32.245.249.236.515: R [bad tcp cksum
30fc (->4614)!] 0:3(3) ack 0 win 0 [RST cko]
0x0000   4500 002b 0000 0000 0f06 7bd4 ffff ffff        E..+......{.....
0x0010   20f5 f9ec 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 30fc 0000 636b 6f00 0000             P...0...cko...

18:48:23.276507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4, length
60: IP (tos 0x0, ttl  15, id 0, offset 0, flags [none], length: 43, bad cksum
efcc (->4e5)!) 255.255.255.255.31337 > 32.245.133.244.515: R [bad tcp cksum
a4f4 (->ba0c)!] 0:3(3) ack 0 win 0 [RST cko]
0x0000   4500 002b 0000 0000 0f06 efcc ffff ffff        E..+............
0x0010   20f5 85f4 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 a4f4 0000 636b 6f00 0000             P.......cko...
```


Source Address – The source address is 255.255.255.255, which is the broadcast address normally used to send broadcast messages to all listening devices on the Internet. 255.255.255.255 is normally the destination address within a packet and not the source. This address was most likely chosen to obscure the true source address and return any response back to broadcast where the true source would be listening.

Source port – 31337. After seeing that traffic was coming from broadcast, I noticed the "eleet" hacker port. While there is chance that an ephemeral port of

31337 could be negotiated, however, several attempts to connect to port 515 from 31337 is somewhat "fishy."

TTL – Without a true source address the time to live is hard to determine, however, all of the packets in this series had the same TTL value of 15.

IP ID – The IP ID is set to 0. Generally, the IP ID is set to a value between 1 and 65,535. All of the packets have an IP ID of 0 and unless they were retransmissions the IP ID should change. This is a sign that the packets were crafted.

TCP Flags – The RST and ACK flags are set. The RST in itself is not suspicious but it is most often the response in an established session. The proper response to the RST/ACK would be an ACK back to the source address acknowledging the receipt of the RST. This is quite possibly the purpose behind crafting the broadcast address as the source.

4. Description of attack:

On the Network Associates web site http://vil.nai.com/vil/content/v_100468.htm is a posting for Backdoor Q.

> There are several variants of this trojan. Filenames used by this trojan can vary between versions. This description is based on one of the later versions. This threat, as with most remote access trojans, consists of 2 components: the client and server. Once the server is running on the victim machine, the hacker is able to connect (and administer that machine) using the client component.[9]

This attack seems to be a scan in attempt to elicit some type of response to determine if there are listening hosts on the network.

5. Attack mechanism:

The intent of this scan seems to be to elicit a response from the destination host. Theoretically, the attacker is probably looking for an ACK back from the sent packet containing the RST and ACK flag set.

This scan to me seems to be crafted, however, I would say poorly. The response to an unsolicited RST/ACK should be no response at all. The intent may have been to elicit a response from poorly configured systems with the assumption that if an ACK was returned that the systems are not maintained and vulnerable.

[9] NAI

6. Correlation:

I found a discussion on the same type of traffic at the following link:
http://www.shmoo.com/mail/firewalls/jun01/msg00006.shtml

CERT has advisories posted regarding some of the vulnerabilities in the Line Printer Daemon (LPD) at the following links:
http://www.cert.org/advisories/CA-2001-30.html
http://www.cert.org/advisories/CA-2001-32.html

There have been quite a few postings in practicals regarding this same type of scan. One of particular interest was one posted by Trenton Riddell (http://www.giac.org/practical/Trenton_Riddell_GCIA.doc). Unlike myself, he captured the same traffic outside his network that alerted snort as "Backdoor Q access."

7. Evidence of active targeting:

I do not feel that this was active targeting. The scans are using addresses throughout the network in an attempt to find a listening host. The way the tool was crafted is probably the reason for the specific port the scan was looking for.

8. Severity: 0

severity = (criticality + lethality) – (system countermeasures + network countermeasures)
Each value is ranked on a scale from 1 (lowest) to 5 (highest).

Criticality: 1

The scan was targeting the LPD with a RST/ACK that should elicit no response.

Lethality: 3

If this scan would have responded with the intended response, the attacker could exploit the LPD with a specially crafted packet to allow the attacker to gain access to the system and traverse the network.[10]

System countermeasures: 2

Since this is an older vulnerability, if systems that run the Line Printer Daemon are present it should have been patched with the available patch.

[10] CERT

Network countermeasures: 2

The ranking seems a little high but the fact that we saw traffic from 255.255.255.255 into the network makes me think that the network is poorly secured. My company network denies broadcast traffic at the border and is again defined on the firewall.

9. Defensive recommendations:

If the network owns the border router, it should configure the ACLs to deny all traffic incoming with the source address of 255.255.255.255 to start with. Blocking ports at the router is also good practice for services that are not utilized. The targeted port 515, definitely, should be blocked at the router. There should not be any legitimate traffic from the external network coming in directly to a printer daemon.

If vulnerable systems are located on the network, they should be analyzed to determine if a compromise has occurred.

10. Multiple choice test question:

What should the response be to an unsolicited RST/ACK?

a. ACK
b. RST
c. FIN/ACK
d. None of the above

Answer: c
An unsolicited RST/ACK should not elicit a response according to RFC 793 Transmission Control Protocol.[11]

Network Detect 2: Proxy scan

```
Jun  20  00:00:00  tcp    216.232.9.229(1422)        xxx.xxx.xxx.231(3127),
denied
Jun  20  00:00:06  tcp    220.99.138.166(4867)       xxx.xxx.xxx.35(3127),
denied
Jun  20  00:00:08  tcp    220.99.138.166(4738)       xxx.xxx.xxx.37(3127),
denied
```

[11] IETF

```
Jun  20  00:00:08  icmp   217.88.124.253            xxx.xxx.xxx.221
denied
Jun  20  00:00:14  tcp    220.99.138.166(3208)      xxx.xxx.xxx.35(3128),
denied
Jun  20  00:00:16  tcp    216.232.9.229(1690)       xxx.xxx.xxx.231(1080),
denied
```

1. Source of Trace:

This was taken from the external router log produced on my company network. Our intrusion detection is inside of the external router and the attack did not reach anything that would log; therefore, no other logs are available to correlate the data. The internal IP addresses have been obfuscated for security reasons.

2. Detect was generated by:

Self observation of the external router logs pulled on a daily basis. Looking at the logs for signs of malicious behavior by hand can be tedious but there can also be a wealth of knowledge in what is happening just outside your border. Here is a compiled example of the scan:

```
                        Access Attempts logged By MY_ROUTER
Mon  Day Time      Type   Source Address (Port)      Destination Address (Port)
---  --
Jun  20  00:00:06  tcp    220.99.138.166(4867)      xxx.xxx.xxx.35(3127),
denied
Jun  20  00:00:06  tcp    220.99.138.166(4867)      xxx.xxx.xxx.35(3127),
denied
Jun  20  00:00:06  tcp    220.99.138.166(4867)      xxx.xxx.xxx.35(3127),
denied
Jun  20  00:00:22  tcp    220.99.138.166(3486)      xxx.xxx.xxx.35(1080),
denied
Jun  20  00:00:40  tcp    220.99.138.166(4068)      xxx.xxx.xxx.36(3128),
denied
Jun  20  00:01:01  tcp    220.99.138.166(4738)      xxx.xxx.xxx.37(3127),
denied
Jun  20  00:01:28  tcp    220.99.138.166(3825)      xxx.xxx.xxx.40(3128),
denied
Jun  20  00:01:36  tcp    220.99.138.166(4105)      xxx.xxx.xxx.40(1080),
denied
Jun  20  00:01:44  tcp    220.99.138.166(4382)      xxx.xxx.xxx.41(3127),
denied
Jun  20  00:01:52  tcp    220.99.138.166(4869)      xxx.xxx.xxx.41(3128),
denied
Jun  20  00:01:58  tcp    220.99.138.166(3825)      xxx.xxx.xxx.40(3128),
denied
Jun  20  00:02:04  tcp    220.99.138.166(4105)      xxx.xxx.xxx.40(1080),
denied
Jun  20  00:02:24  tcp    220.99.138.166(3209)      xxx.xxx.xxx.41(1080),
denied
Jun  20  00:02:40  tcp    220.99.138.166(3507)      xxx.xxx.xxx.42(3127),
denied
```

```
Jun  20  00:02:53  tcp    220.99.138.166(4053)       xxx.xxx.xxx.42(1080),
denied
Jun  20  00:03:02  tcp    220.99.138.166(4326)       xxx.xxx.xxx.43(3127),
denied
Jun  20  00:03:04  tcp    220.99.138.166(3698)       xxx.xxx.xxx.44(3128),
denied
Jun  20  00:03:11  tcp    220.99.138.166(4727)       xxx.xxx.xxx.43(3128),
denied
Jun  20  00:03:16  tcp    220.99.138.166(3427)       xxx.xxx.xxx.44(3127),
denied
Jun  20  00:03:28  tcp    220.99.138.166(3698)       xxx.xxx.xxx.44(3128),
denied
Jun  20  00:03:33  tcp    220.99.138.166(3970)       xxx.xxx.xxx.44(1080),
denied
Jun  20  00:03:52  tcp    220.99.138.166(4247)       xxx.xxx.xxx.45(3127),
denied
Jun  20  00:04:00  tcp    220.99.138.166(4629)       xxx.xxx.xxx.45(3128),
denied
Jun  20  00:04:17  tcp    220.99.138.166(3345)       xxx.xxx.xxx.46(3127),
denied
Jun  20  00:04:30  tcp    220.99.138.166(3902)       xxx.xxx.xxx.46(1080),
denied
Jun  20  00:04:37  tcp    220.99.138.166(4176)       xxx.xxx.xxx.47(3127),
denied
Jun  20  00:04:48  tcp    220.99.138.166(4969)       xxx.xxx.xxx.47(1080),
denied
Jun  20  00:05:04  tcp    220.99.138.166(3554)       xxx.xxx.xxx.48(3128),
denied
```

3. Probability the source address was spoofed:

Since there were no other logs to correlate the data to, I can't tell if the packets have any signs of crafting. However, because of the ports that are being scanned the source address is probably real.

Source Address – The source address is 220.99.138.166. If you do a whois on the IP address, you will be returned with the following information from APNIC.

```
 % [whois.apnic.net node-1]
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html
inetnum:      220.96.0.0 - 220.99.255.255
netname:      OCN-JPNIC-JP
descr:        OCN Provided By NTT-Communications which is ISP
descr:        in Chiyoda-ku, Tokyo, Japan
country:      JP
admin-c:      JNIC1-AP
tech-c:       JNIC1-AP
remarks:      ************************************************
remarks:      Allocated to JPNIC member. Authoritative
remarks:      information regarding assignments and allocation
remarks:      made from within this block can also be queried
remarks:      at whois.nic.ad.jp. To obtain an English output
remarks:      query whois -h whois.nic.ad.jp x.x.x.x/e
remarks:      Email address for spam or abuse complaints : abuse@ocn.ad.jp
remarks:      ************************************************
mnt-by:       MAINT-JPNIC
mnt-lower:    MAINT-JPNIC
changed:      hm-changed@apnic.net 20020904
changed:      ip-apnic@nic.ad.jp 20040413
status:       ALLOCATED PORTABLE
```

```
source:       APNIC
role:         Japan Network Information Center
address:      Kokusai-Kougyou-Kanda Bldg 6F, 2-3-4 Uchi-Kanda
address:      Chiyoda-ku, Tokyo 101-0047, Japan
country:      JP
phone:        +81-3-5297-2311
fax-no:       +81-3-5297-2312
e-mail:       hostmaster@nic.ad.jp
admin-c:      SS13-AP
tech-c:       SY7-AP
nic-hdl:      JNIC1-AP
mnt-by:       MAINT-JPNIC
changed:      apnic-ftp@nic.ad.jp 19990629
changed:      ip-staff@nic.ad.jp 20030806
source:       APNIC
inetnum:      220.99.128.0 - 220.99.255.255
netname:      PLALA
descr:        Plala Networks Inc.
country:      JP
admin-c:      MN2905JP
tech-c:       HS3694JP
remarks:      This information has been partially mirrored by APNIC from
remarks:      JPNIC. To obtain more specific information, please use the
remarks:      JPNIC whois server at whois.nic.ad.jp. (This defaults to
remarks:      Japanese output, use the /e switch for English output)
changed:      apnic-ftp@nic.ad.jp 20030203
remarks:      This information has been partially mirrored by APNIC from
remarks:      JPNIC. To obtain more specific information, please use the
remarks:      JPNIC whois server at whois.nic.ad.jp. (This defaults to
remarks:      Japanese output, use the /e switch for English output)
changed:      apnic-ftp@nic.ad.jp 20040609
source:       JPNIC
```

In that it is a valid IP address according to APNIC and the ports that it is trying to probe may return some type of response to allow the hacker to gain access to the system leads me to believe that the IP address is not spoofed.

4. Description of attack:

The attack is obviously a scan for Socks and squid proxies in attempt to locate a compromised system or one that is running the service to exploit possible vulnerabilities within the service.

A CVE that covers the squid proxy vulnerability can be found at the following URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0068

A CAN, which is under review at this time, that covers the socks proxy vulnerability can be found at the following URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0315

Along with the proxies, the attacker appears to be probing for port 3127. This port is left open as a back door by the myDoom virus[12] which future could be an avenue for a denial of service attack in the future, according to CERT at http://www.cert.org/incident_notes/IN-2004-01.html.

[12] LinkLogger

5. Attack mechanism:

The attacker is probing the network in search of responding ports. Since there are no other logs to correlate with, I can only assume that the TCP packets that are being sent are to elicit some response to find a listening or compromised host. For example, the attacker sends an unsolicited TCP packet with the ACK flag set. The normal response would be for the host to respond back with RST. If that is indeed the case, the attacker now knows that: 1) the host is alive and listening, and 2) there is no filtering in place.

If the attacker receives no response then he can assume: 1) some type of filtering is in place, or 2) that the host is not alive.

If the attacker receives the response that is being solicited then he can attempt to exploit the vulnerabilities with a buffer overflow using specially crafted packets. In the case of the back door left by the myDoom virus, the attacker could possibly take control of that system to launch a denial of service attack.

6. Correlations:

I found a posting at http://lists.sans.org/pipermail/unisog/2004-March/006955.php that states that they had seen some scans, particularly on the weekends, just a couple of months back. I was unable to see any other postings to correlate my findings.

7. Evidence of active targeting:

It appears to be targeted in the sense that it is targeting three specific ports as possible back doors to compromised systems. Also, the attacker is scanning the full range of addresses on this particular subnet. The scans are not rapid in succession but fairly consistent.

8. Severity: 1

severity = (criticality + lethality) – (system countermeasures + network countermeasures)
Each value is ranked on a scale from 1 (lowest) to 5 (highest).

Criticality: 3

I believe that criticality is low based on the fact that we do not run socks or squid proxies on our network. It is possible that the myDoom virus could be introduced

but active virus scanning with updated definitions is in use. However, the criticality of this attack can lead to compromised systems.

Lethality: 3

I do not think that this scan in itself is very lethal but the nature of business on my network a compromise could be very lethal. A continued monitoring of this type of scan and any other associated IP addresses should be considered.

System countermeasures: 2

The systems on this network are patched and updated regularly or as they become available. The systems administrators do a fair job of ensuring that the systems are secure and up to date.

Network countermeasures: 3

I am happy to say that this scan was denied at the border router. The network currently uses a defense-in-depth approach. Inside the border router, using extensive ACLs, is monitored by a network IDS. A firewall is then in place that all traffic to the internal network is routed through. The internal network is monitored by multiple network IDS on various segments along with Cisco IDS modules in the switches. Correlation with various agencies allows us to put preventative measures up at the border ahead of time.

9. Defensive Recommendations:

The border router should be configured to block incoming requests for services that are not in use or vulnerable to remote access. If the scan is coming from a specific host, then that IP address can be blocked at the router. Firewalls, should also be set to block any unused services. Since the scan also includes the port commonly associated with the myDoom virus, a full system scan with updated definitions should be done on all systems to ensure that none of the systems have been compromised.

10. Multiple choice test question:

What range of ports does the myDoom virus open up and listen on?

a. 4200-4000
b. 3127-3198
c. 1024-1029
d. 135-139

Answer: b.
A system that has been compromised by the myDoom virus opens ports 3127-3198, according to CERT http://www.cert.org/incident_notes/IN-2004-01.html.

Questions from the intrusions@incidents.org mailing list

I posted this detect to the intrusions mailing list. I only received one reply from Donald Smith. Mostly, comments were made to my post as helpful hints but he did post some questions to be answered.

1) From Donald Smith 'Donald.Smith@qwest.com'

```
>Jun  20  00:02:40  tcp    220.99.138.166(3507)       xxx.xxx.xxx.42(3127),
>denied
>Jun  20  00:02:53  tcp    220.99.138.166(4053)       xxx.xxx.xxx.42(1080),
>denied
>Jun  20  00:03:02  tcp    220.99.138.166(4326)       xxx.xxx.xxx.43(3127),
>denied
Look at the source and dst ports notice anything?
There appears to be some "near" matches (1st and forth digit matching.
```

I reviewed the log and found a few that fit that pattern. Take for example the top line:

```
>Jun  20  00:02:40  tcp    220.99.138.166(3507)       xxx.xxx.xxx.42(3127),
>denied
```

Here the first and forth digits in the source and destination ports do match. However, it is not consistent throughout the log. I did notice that the source port is using a small range of ports.

2) From Donald Smith 'Donald.Smith@qwest.com'

```
Did you notify the abuse department listed above?
```

I did not notify the abuse department. Our normal procedure is to notify US DOE-**CIAC** (Computer Incident Advisory Capability). Since I work for a government contractor, notifications are made to CIAC in order to correlate events to see if there are any patterns to such attacks. CIAC will usually make the call on whether an event warrants notification of the abuse department. At times, the decision to watch for repeated traffic without notification will allow you to get a better picture of what an attacker is trying to accomplish.

3) From Donald Smith 'Donald.Smith@qwest.com'

```
>The attack is obviously a scan for Socks and squid proxies in attempt to
YES
>locate a compromised system or one that is running the service to exploit
Compromised?
```

The firewall logs were analyzed to ensure that the source address did not make any attempts to pass through. The targets machines were scanned with Nessus

to make sure that the targeted ports were not listening. There were no signs of compromise on any of the machines that were targeted.

4) From Donald Smith 'Donald.Smith@qwest.com'

```
Some servers scan for proxies to prevent spammers from using one proxies to
send mail to them.
Did you check and see what that ip is?
```

I was unable to determine any information other than the output by APNIC. What I did find though is that the IP address could possibly be infected and the scan is a result of worm propagation. MyDoom.B propagates by the opening up ports to serve as proxies. The ports used by MyDoom.B variant are: TCP 80, 1080, 3128, 8080, 10080. Along with these ports it may look for port 3127 opened by a previous infection of the A variant.

5) From Donald Smith 'Donald.Smith@qwest.com'

```
>If the attacker receives no response then he can assume: 1) some type of
>filtering is in place, or 2) that the host is not alive.
Any other possibilities?
```

This is the only information that I could find.

6) From Donald Smith 'Donald.Smith@qwest.com'

```
>What range of ports does the myDoom virus open up and listen on?
Mydoom.a or.b?
```

W32.Mydoom.A@mm (also known as W32.Novarg.A) is a mass-mailing worm that arrives as an attachment with the file extension .bat, .cmd, .exe, .pif, .scr, or .zip.

When a computer is infected, the worm sets up a backdoor into the system by opening TCP ports 3127 through 3198, which can potentially allow an attacker to connect to the computer and use it as a proxy to gain access to its network resources.[13]

Network Detect 3: DNS named version attempt

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] DNS named version attempt [**]
06/23/02-08:22:36.474488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x48
203.197.102.167:2168 -> 46.5.139.43:53 UDP TTL:42 TOS:0x0 ID:59165 IpLen:20
DgmLen:58
Len: 30
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4...........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

[13] Symantec Security Response

```
[**] DNS named version attempt [**]
06/23/02-06:33:46.514488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x48
203.197.102.180:1434 -> 46.5.220.1:53 UDP TTL:44 TOS:0x0 ID:13154 IpLen:20
DgmLen:58
Len: 30
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4..........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

1. Source of Trace:

The raw log was taken from the GIAC Certification Practical logs at random: http://www.incidents.org/logs/RAW/2002.5.10. The IDS is on the 46.5.0.0/16 network. The true IP addresses are obviously obscured because of the bad checksums in the IP and TCP headers.

2. Detect was generated by:

This detect was generated by Snort. The alerts were generated using the default configuration file that comes with the Snort install. All of the signatures were used. The following command was issued to run Snort:

```
snort -c ../etc/snort.conf -r ../logs/2002.5.10 -h 46.5.0.0/16 -k none
-deyv
```

The data was then analyzed using the following tcpdump command:

```
tcpdump -nevXS -r /media/cdrom/2002.5.23 'dst port 53'

06:33:46.514488 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4, length
72: IP (tos 0x0, ttl  44, id 13154, offset 0, flags [none], length: 58, bad
cksum 23d9 (->1ed1)!) 203.197.102.180.1434 > 46.5.220.1.53: [bad udp cksum
f7fa!]  4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)
0x0000   4500 003a 3362 0000 2c11 23d9 cbc5 66b4        E..:3b..,.#...f.
0x0010   2e05 dc01 059a 0035 0026 8e01 1234 0080        .......5.&...4..
0x0020   0001 0000 0000 0000 0776 6572 7369 6f6e        .........version
0x0030   0462 696e 6400 0010 0003                       .bind.....

08:22:36.474488 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4, length
72: IP (tos 0x0, ttl  42, id 59165, offset 0, flags [none], length: 58, bad
cksum c300 (->bdf8)!) 203.197.102.167.2168 > 46.5.139.43.53: [bad udp cksum
f7fa!]  4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)
0x0000   4500 003a e71d 0000 2a11 c300 cbc5 66a7        E..:....*.....f.
0x0010   2e05 8b2b 0878 0035 0026 dc06 1234 0080        ...+.x.5.&...4..
0x0020   0001 0000 0000 0000 0776 6572 7369 6f6e        .........version
0x0030   0462 696e 6400 0010 0003                       .bind.....

18:50:19.824488 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4, length
72: IP (tos 0x0, ttl  47, id 21233, offset 0, flags [none], length: 58, bad
cksum 762c (->7026)!) 210.195.43.35.4507 > 46.5.156.176.53: [bad udp cksum
f9f9!]  4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)
0x0000   4500 003a 52f1 0000 2f11 762c d2c3 2b23        E..:R.../.v,..+#
```

```
0x0010   2e05 9cb0 119b 0035 0026 f6e2 1234 0080        .......5.&...4..
0x0020   0001 0000 0000 0000 0776 6572 7369 6f6e        .........version
0x0030   0462 696e 6400 0010 0003                       .bind.....
```

3. Probability the source address was spoofed:

The source IP address is probably not spoofed because the attacker is doing reconnaissance by requesting the version of BIND that the DNS is using. In an attempt to gain this information, the reply would need to respond to a valid IP address.

Source Address – The source addresses are three different IP addresses.

Destination Address – Again, three different IP addresses.

Destination Port – All three packets are destined for port 53, the reserved port for DNS.

Payload – All three packets are querying for the version of BIND.

4. Description of the attack:

This is a reconnaissance attempt to gain information from the DNS server to possibly exploit known vulnerabilities in earlier versions of BIND. The payload shows that a version.bind attempt was sent. There was no indication that a reply was sent back, so this attempt was unsuccessful.

5. Attack Mechanism:

An attacker can query a DNS server for the version of BIND running. Some versions of BIND, by default, respond to these queries while BIND version 9; by default, does not. A response to this query can assist an attacker in discovering servers that are potentially vulnerable to exploits associated with specific versions of BIND.[14]

6. Correlations:

I found some information on the same type of reconnaissance on a University network at the following URL: http://www.dshield.org/pipermail/intrusions/2001-June/000492.php.

[14] Snort.org

Also, there have been several postings in GCIA practicals on the same type of probe resulting in “DNS named version attempt” alerts.

There are several CVEs listed on buffer overflows in BIND:

CVE-1999-0009 Inverse query buffer overflow in BIND 4.9 and BIND 8.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0009

CVE-1999-0833 Buffer overflow in BIND 8.2 via NXT records.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0833

CVE-2001-0010 Buffer overflow in transaction signature (TSIG) handling code in BIND 8 allows remote attackers to gain root privileges.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0010

CVE-2001-0011 Buffer overflow in nslookupComplain function in BIND 4 allows remote attackers to gain root privileges.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0011

7. Evidence of active targeting:

Using this log is a small picture of the total traffic and only a couple of systems were targeted, however, this does not seem to indicate active targeting. The attacker seems to be doing reconnaissance or probing for potential targets based on a response from a DNS server running a version of BIND that responses to named version requests.

8. Severity: 2

severity = (criticality + lethality) – (system countermeasures + network countermeasures)
Each value is ranked on a scale from 1 (lowest) to 5 (highest).

Criticality: 5

If the servers targeted were actually DNS servers, the information that is contained on this server is critical in that it provides name resolution to the entire organization.

Lethality: 2

This type of traffic is reconnaissance in nature and is not considered lethal. If the request would have provided what version of BIND the server was running, then a more lethal exploit could have been initiated.

System countermeasures: 2

The system is unknown by the traffic in the log, but since there was not a reply to the request, systems are updated and patched.

Network countermeasures: 3

Assuming that this is an external DNS server, it should be located outside the firewall. No response from the server suggests that there was no information leakage to the attacker.

9. Defensive Recommendations:

Ensure that all external DNS servers are placed outside the firewall. The firewall should be configured to block all traffic destined for port 53 on the internal network. Patch and update all DNS servers to the most current version of BIND. Configure the BIND not to respond to version requests is a possible solution. Check firewall logs to see if other reconnaissance by the same IP address has been logged.

10. Multiple choice test question:

```
23:38:27.655334 24.34.240.10.53 > 207.189.175.3 13079*- 1/0/0 CHAOS)
TXT 9.1.1 (48) (DF)
```

What type of traffic does this indicate?

a. DNS named version request
b. Echo reply
c. GET request
d. DNS named version reply

Answer: d.
If a DNS server is queried with version.bind and elicits a response, then the traffic seen would be the response. This response indicates that the DNS server is using BIND version 9.1.1.

Assignment #3 – Analyze This

Executive Summary

The following report is the result of a preliminary audit of the University network as requested by GIAC.org. After reading this report, the University should have a

better understanding of the current state of the network and the defensive measures needed to protect their assets on the network in the future.

Over the five days that were analyzed, there were 130161 Alerts, 13816216 Port Scans, and 5098 Out of Spec packets used in the analysis. The early analyses of the data lead me to believe that just about every computer on the network had been compromised. However, this does not appear to be the case. Although there are some signs of compromise, the majority of the traffic appears to have generated a great amount of false positives. This results in the usefulness of the intrusion detection system being minimized. There are some simple changes to the configurations that will be addressed in the Defensive Recommendations that will help resolve these issues.

The only major issue that I have noted is the amount of what appears to be Peer-to-Peer (P2P) file sharing that is taking place on the network. This type of traffic is most likely the reason for the compromised systems. Not only does file sharing expose the network to viruses and Trojans, but also the legal ramifications of sharing copyrighted material that can be imposed on the University and students.

Along with the P2P traffic, there appears to be the use of remote desktop applications by various hosts on the University network. Remote desktop applications could potentially allow unauthorized access to the network through vulnerabilities associated with these applications, which will be discussed in detail later in this report.

On a good note, there were no alerts generated for internal-to-internal traffic. This is usually a good thing. There appears to be signs of worm propagation, but nothing has alerted to such traffic. However, there is a possibility that there are no sensors or rules in place to monitor internal traffic.

Suspicious Internal Hosts

MY.NET.109.25	MY.NET.150.226	MY.NET.69.210
MY.NET.112.189	MY.NET.153.195	MY.NET.80.119
MY.NET.112.139	MY.NET.17.45	MY.NET.80.224
MY.NET.112.199	MY.NET.43.10	MY.NET.84.186
MY.NET.112.226	MY.NET.69.155	

The suspicious hosts are based on extensive P2P file sharing and signs of worm propagation seen coming from these hosts. These hosts should be removed for the network and observed for signs of compromise.

Defensive Recommendations

Defensive measures are often difficult in an environment that must provide an accessible means to information to the internal network, as well as, external. Unfortunately, this exposes those environments to malicious activity that a more

secure environment does not have to worry about. The University should assess the risks and the consequences associated with the malicious and illegal activities taking place on their network and protect critical and valuable assets by the use of access control lists (ACLs) and firewall rules.

The University should consider hardening the operating systems on the network hosts to prevent compromises through security holes that exist in default installations of operating systems. A good guide for this can be at the NSA Security Recommendation Guide website at the following link: http://nsa2.www.conxion.com/win2k/download.htm[15]

A security policy should be in place and enforced. Enforcement could be through the use of disciplinary action and/or the use of a product to disable or quarantine hosts that violate this policy. Products such as netIQ[16] or StillSecure VAM[17] can provide this type of policy management, as well as, patch management to ensure that hosts connected to the network are up to date on all security patches and fixes.

These recommendations are a starting point that will help to reduce future risks to the network. A more complete audit should be performed to determine what acceptable and authorized traffic is and tune the intrusion detection system accordingly. Other recommendations will be made throughout this report.

Technical Suggestions:

To reduce the amount of false positives:
Tune Snort rules to be more content specific, rather than alerting on a specific port. Determine what makes a packet malicious and use that as the content. Also, modify the portscan preprocessor options by raising the detection period to reduce the amount of portscan alerts by random scanning.

To monitor traffic:
Tune Snort rules to log rather than alert for rules that are meant to monitor activity to valuable assets.

And last, a complete vulnerability scan of all network assets should be done periodically. In order to protect the network, a complete understanding of what is out there and what is vulnerable is a must. Implement a patch management policy to address the vulnerabilities that are found to ensure that the appropriate patches are applied in a timely manner.

[15] NSA
[16] netIQ
[17] StillSecure

Log files Analyzed

Alert logs	Scan logs	OOS logs
alert_040420	scans_040420	oos_report_040417
alert_040421	scans_040421	oos_report_040418
alert_040422	scans_040422	oos_report_040419
alert_040423	scans_040423	oos_report_040420
alert_040426	scans_040426	oos_report_040422

The dates on the OOS logs do not correspond with the Alert and Scan logs; however, the dates in the OOS log file do match with the dates of the Alert and Scan logs. The dates between 4/23 and 4/26 were skipped in order to receive a matching set of log files out the most current files available.

Alerts

Top Ten Alert Summary

Alert	Hits	Src IPs	Dst IPs	Ports	%
EXPLOIT x86 NOOP	38903	2084	916	65	29.89
MY.NET.30.4 activity	35289	313		23	27.11
High port 65535 tcp – possible Red Worm – traffic	19471	119	154	86	14.96
MY.NET.30.3 activity	15900	197		23	12.22
SMB Name Wildcard	8627	61	639	137	6.63
Tiny Fragments – Possible Hostile Activity	4412	5	18		3.39
RFB – Possible WinVNC – 010708-1	2355	15	17	1099	1.81
Null Scan!	1936	89	54	152	1.49
NMAP TCP ping!	869	218	67	46	.67
Possible trojan server activity	419	48	52	9	.32

Top 10 External Talkers

Alert type	Count
134.192.42.11	21779
131.92.177.18	5206
209.164.32.205	4763
68.55.155.26	3730
69.136.228.63	3470
64.12.24.34	3073
220.197.192.39	2611
69.138.77.62	2478
151.196.115.104	2454
64.12.24.35	2329

Top 10 Internal Talkers

Alert type	Count
MY.NET.43.8	3230
MY.NET.11.4	3108

MY.NET.69.232	2990
MY.NET.11.7	2508
MY.NET.43.13	2123
MY.NET.153.81	883
MY.NET.150.44	632
MY.NET.75.13	506
MY.NET.150.198	435
MY.NET.70.156	245

EXPLOIT x86 NOOP

In my research of the top alerts, my search began with finding the rules that caused the alert to determine what the possible trigger is. Other than close matches at Snort.org, I found this one in Eric Evans practical assignment.

alert tcp $EXTERNAL any -> $INTERNAL any (msg:"EXPLOIT x86 NOOP";
content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90|"; flags: A+; reference:arachnids,181;)

Rule Source: http://www.geocrawler.com/archives/3/5343/2001/4/50/5487319/[18]

This alert accounts for 29.89% of all the alerts collected. Out of all 38903 "EXPLOIT x86 NOOP" alerts, the majority of the traffic is related to port 80 (HTTP traffic) with the exception of a few hosts that had traffic destined for port 135. We will look at this traffic later. For the most part this appears to be normal Internet related traffic. The rule above triggers on traffic that contains a series of NOOPs (|90 90 90 90|). The alert has most likely triggered on Internet traffic that contains NOOP padding. Internet downloads of binary and image files tend to cause many false positives, as it is common to pad these files. A NOOP exploit uses what is called a "NOOP Sled" to fill memory buffers in an attempt to execute shellcode to gain access to the targeted machine. Now, the traffic related to port 135. Port 135 relates to RPC DCOM, which was exploited with the MSBlaster worm that looks for vulnerable hosts listening on port 135. The following IP addresses alerted with "EXPLOIT x86 NOOP" destined for port 135: MY.NET.190.102, MY.NET.190.93, MY.NET.190.95, MY.NET.190.97, and MY.NET.190.98.

Timestamp	Alert	SrcIP	SrcPort	DstIP	DstPort
04/23-17:44:30.697743	EXPLOIT x86 NOOP	81.203.168.147	4066	MY.NET.190.95	135
04/23-17:44:17.166145	EXPLOIT x86 NOOP	81.203.168.147	4064	MY.NET.190.93	135
04/23-17:49:37.395311	EXPLOIT x86 NOOP	130.228.160.98	1286	MY.NET.190.97	135
04/23-17:27:55.600556	EXPLOIT x86 NOOP	213.39.146.147	2394	MY.NET.190.102	135
04/23-17:26:20.858987	EXPLOIT x86 NOOP	216.114.98.129	2468	MY.NET.190.102	135
04/23-17:26:19.361561	EXPLOIT x86 NOOP	216.114.98.129	2463	MY.NET.190.97	135
04/23-17:26:17.853508	EXPLOIT x86 NOOP	216.114.98.129	2461	MY.NET.190.95	135
04/23-17:11:59.705701	EXPLOIT x86 NOOP	80.116.219.2	4334	MY.NET.190.98	135

[18] Evans, Eric

A look at the scan logs does not indicate that these machines have been compromised, as there are no scans originating from these IP addresses consistent with worm propagation. However, MY.NET.190.93,95, and 97-98 were seen in the "SMB Named Wilcard" that will be discussed later.

Recommendation: Consider modifying this rule, this will reduce the amount of false positive data that an analyst must go through. The concern of the analyst should be to look for signs of attempted shellcode against a targeted machine. The NOOP is a precursor to the execution of shellcode. If this is the purpose of this alert, tune it to include shellcode commands. A scan of the target machines should be done to ensure that there are no signs of compromise. Ensure that all hosts are update on the latest patches and block port 135 traffic from entering the network.

MY.NET.30.4 activity

This alert is generated by a custom Snort rule. The exact reason for this rule is unclear, unless its purpose is to monitor all traffic to this server. This rule seems to generate a great deal of alerts to be analyzed by the analyst. MY.NET.30.4 appears to be a Novell NetWare server running Tomcat Apache, based on the connections to ports 524, 51443, and 8009. Port 524 is the default port for NetWare Core Protocol (NCP), which handles client server requests.[19] Port 524 is also the default port for eDirectory services. Ports 51443 and 8009 are used for remote admin, iManager[20] and Web Manager[21] respectively. The rest of the traffic seen to MY.NET.30.4 is primarily to ports 80 and 443, which are commonly used for the Web Admin Interface that is equivalent to rconsole. There are several IP addresses that appear to be accessing MY.NET.30.4 on the remote administration ports. The top sources for port 51443 are:

Source Address	# of Alerts	Destination Ports
134.192.42.11	21779	51443
69.136.228.63	3470	51443
68.33.49.146	1581	51443, 80
172.209.111.241	811	51443, 80

The first IP address resolved to the University of Maryland, which leads me to believe that this is legitimate traffic. The next two IP addresses resolve to Comcast and the last to America Online.

```
#################################################################
OrgName:    University of Maryland at Baltimore
OrgID:      UMAB-1
Address:    601 W. Lombard St
City:       Baltimore
StateProv:  MD
```

[19] Faske, Mark
[20] Hughes, Timothy
[21] Que

```
PostalCode: 21201
Country:    US

NetRange:   134.192.0.0 - 134.192.255.255
CIDR:       134.192.0.0/16
NetName:    UMAB-NET
NetHandle:  NET-134-192-0-0-1
Parent:     NET-134-0-0-0-0
NetType:    Direct Assignment
NameServer: NMS1.UMARYLAND.EDU
NameServer: NMS2.UMARYLAND.EDU
Comment:
RegDate:    1988-06-15
Updated:    2004-09-02

TechHandle: FS178-ARIN
TechName:   Smith, Frederick
TechPhone:  +1-410-706-8337
TechEmail:  fsmith@umaryland.edu

OrgTechHandle: BSM40-ARIN
OrgTechName:   Smith, Bryan
OrgTechPhone:  +1-410-706-8237
OrgTechEmail:  bsmith@umaryland.edu

# ARIN WHOIS database, last updated 2004-09-07 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.

**************************************************************************
Comcast Cable Communications, Inc. JUMPSTART-3 (NET-69-136-0-0-1)
                                   69.136.0.0 - 69.143.255.255
Comcast Cable Communications, Inc BALTIMORE-B-6 (NET-69-136-224-0-1)
                                   69.136.224.0 - 69.136.239.255

# ARIN WHOIS database, last updated 2004-09-07 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.

**************************************************************************
OrgName:    America Online
OrgID:      AOL
Address:    22000 AOL Way
City:       Dulles
StateProv:  VA
PostalCode: 20166
Country:    US

NetRange:   172.192.0.0 - 172.211.255.255
CIDR:       172.192.0.0/12, 172.208.0.0/14
NetName:    AOL-172BLK-2
NetHandle:  NET-172-192-0-0-1
Parent:     NET-172-0-0-0-0
NetType:    Direct Allocation
NameServer: DAHA-01.NS.AOL.COM
NameServer: DAHA-02.NS.AOL.COM
NameServer: DAHA-07.NS.AOL.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2002-02-13
Updated:    2003-08-08

TechHandle: AOL-NOC-ARIN
TechName:   America Online, Inc.
TechPhone:  +1-703-265-4670
TechEmail:  domains@aol.net

OrgAbuseHandle: AOL382-ARIN
OrgAbuseName:   Abuse
OrgAbusePhone:  +1-703-265-4670
OrgAbuseEmail:  abuse@aol.net
```

```
OrgNOCHandle: AOL236-ARIN
OrgNOCName:   NOC
OrgNOCPhone:  +1-703-265-4670
OrgNOCEmail:  noc@aol.net

OrgTechHandle: AOL-NOC-ARIN
OrgTechName:   America Online, Inc.
OrgTechPhone:  +1-703-265-4670
OrgTechEmail:  domains@aol.net

# ARIN WHOIS database, last updated 2004-09-07 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Port 524 traffic was almost entirely comprised of four IP addresses, they are as follows:

Source Address	# of Alerts	Destination Ports
68.55.155.26	400	524, 8009
68.34.92.70	292	524
64.8.195.10	226	524
68.57.90.146	198	524

The only other port of concern is 8009, which was accessed by two IP addresses. The first one 68.55.155.26, the major talker of port 524, was again the major player. The rest of the alerts belonged to the IP address 68.55.158.146. Most likely this the same person with a dynamically assigned IP address as both IP addresses resolves to Comcast. The rest of the IP addresses also resolved to Comcast with the exception of 64.8.195.10. This IP address resolved to DSL.net, as seen in the following Whois results.

```
OrgName:    DSL.net, Inc.
OrgID:      FTCI
Address:    545 Long Wharf Dr. 5th floow
City:       New Haven
StateProv:  CT
PostalCode: 06511
Country:    US

NetRange:   64.8.192.0 - 64.8.255.255
CIDR:       64.8.192.0/18
NetName:    DSL-NET-15
NetHandle:  NET-64-8-192-0-1
Parent:     NET-64-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.DSL.NET
NameServer: NS2.DSL.NET
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2001-02-06
Updated:    2004-04-27

OrgAbuseHandle: ABUSE177-ARIN
OrgAbuseName:   Abuse
OrgAbusePhone:  +1-203-772-1000
OrgAbuseEmail:  abuse@dsl.net

OrgNOCHandle: NOC291-ARIN
OrgNOCName:   Network Operations Center
OrgNOCPhone:  +1-203-772-1000
OrgNOCEmail:  noc@dsl.net

OrgTechHandle: IPADM54-ARIN
OrgTechName:   IP Administration
```

```
OrgTechPhone:  +1-203-772-1000
OrgTechEmail:  ipadmin@dsl.net

# ARIN WHOIS database, last updated 2004-09-07 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

A good majority of the activity is requests on port 80, which would provide strong support to the fact that this node is a web server.

Mark Faske analyzed the same traffic in his practical assignment. His assumptions were the same in that the majority of the ports were related to web-based administration on Novell servers.

Recommendation: Based on the shear volume of alert activity logged, it would appear that too many generalized alerts are being logged from this web server. The activity to this web server makes up 27.11% of the alert volume for the five days analyzed. Combined with the alerts generated from a like web server MY.NET.30.3, these alerts are responsible for more than a third of the alerts generated, 39.33% to be exact. An analyst can quickly be consumed by the amount of generalized alerts only to miss more important alerts. The Snort rules should be tuned to log such activity rather than alert, this will result in more important alerts to trigger to indicate malicious activity. A periodic vulnerability scan of public access servers should be performed to ensure that only those services that are suppose to be running are the only services running. If remote management is approved practice, then consideration should be given to using secure channels. Examples would be SSH or encrypted VPN access.

High port 65535 tcp – possible Red Worm – traffic

This alert, which amounts to approximately 15% of the alerts, is another custom rule to detect port 65535 traffic that is associated with RC1 trojan and Adore worm. There seems to be a considerable amount of traffic in and out of the local network, most of which was associated with an ephemeral port connecting to port 65535. There are several alert associated with port 65535 traffic to external port 25 (SMTP).

At first, the assumption was that the network had a major worm infection. I kept trying various sources to find how the Red Worm propagates and hours of staring at all the alerts generated by port 65535. I then correlated the scan logs of various internal hosts with the timestamps of the alerts to find that prior to traffic that generates the “High port 65535 tcp – possible Red Worm – traffic” alert, there are numerous SYN scans for ports 6881-6888.

Take a look at the following logs, the first one is the scan log and the second is from the alert database with corresponding traffic.

```
Apr 22 11:37:10 130.85.153.81:1645 -> 206.170.204.115:6882 SYN ******S*
Apr 22 11:37:10 130.85.153.81:1646 -> 68.252.236.34:6883 SYN ******S*
Apr 22 11:37:10 130.85.153.81:1648 -> 164.58.59.60:6881 SYN ******S*
```

```
Apr 22 11:37:10 130.85.153.81:1649 -> 65.124.168.126:6883 SYN ******S*
Apr 22 11:37:10 130.85.153.81:1653 -> 213.51.48.5:6883 SYN ******S*
Apr 22 11:37:10 130.85.153.81:1654 -> 68.248.29.41:6881 SYN ******S*
Apr 22 11:37:10 130.85.153.81:1655 -> 65.3.81.59:6881 SYN ******S*
Apr 22 11:37:10 130.85.153.81:1656 -> 65.92.168.80:6882 SYN ******S*
Apr 22 11:37:10 130.85.153.81:1661 -> 68.162.146.118:6886 SYN ******S*
Apr 22 11:37:12 130.85.153.81:1644 -> 220.39.6.191:6887 SYN ******S*
```

Timestamp	Alert	SrcIP	SrcPort	DstIP	DstPort
04/22-11:38:47.878631	High port 65535 tcp - possible Red Worm - traffic	MY.NET.153.81	1759	195.36.245.141	65535
04/22-11:38:49.290031	High port 65535 tcp - possible Red Worm - traffic	195.36.245.141	65535	MY.NET.153.81	1759
04/22-11:38:49.290391	High port 65535 tcp - possible Red Worm - traffic	MY.NET.153.81	1759	195.36.245.141	65535
04/22-11:38:51.719914	High port 65535 tcp - possible Red Worm - traffic	195.36.245.141	65535	MY.NET.153.81	1759
04/22-11:38:51.720424	High port 65535 tcp - possible Red Worm - traffic	MY.NET.153.81	1759	195.36.245.141	65535
04/22-11:38:51.725190	High port 65535 tcp - possible Red Worm - traffic	195.36.245.141	65535	MY.NET.153.81	1759
04/22-11:38:51.730467	High port 65535 tcp - possible Red Worm - traffic	195.36.245.141	65535	MY.NET.153.81	1759
04/22-11:38:51.860697	High port 65535 tcp - possible Red Worm - traffic	195.36.245.141	65535	MY.NET.153.81	1759
04/22-11:38:51.861999	High port 65535 tcp - possible Red Worm - traffic	MY.NET.153.81	1759	195.36.245.141	65535
04/22-11:38:51.862124	High port 65535 tcp - possible Red Worm - traffic	MY.NET.153.81	1759	195.36.245.141	65535
04/22-11:38:51.862266	High port 65535 tcp - possible Red Worm - traffic	MY.NET.153.81	1759	195.36.245.141	65535
04/22-11:38:53.299850	High port 65535 tcp - possible Red Worm - traffic	195.36.245.141	65535	MY.NET.153.81	1759
04/22-11:38:53.310262	High port 65535 tcp - possible Red Worm - traffic	195.36.245.141	65535	MY.NET.153.81	1759

It appears that these hosts are using P2P software known as BitTorrent. BitTorrent listens on ports 6881-6999.[22] What I believe is happening is that the hosts are scanning for BitTorrent servers and making a connection. Once the connection is made, port 65535 is negotiated as the file transfer port. I cannot find any hard evidence that this is exactly true, but it seems logical.

The rest of the traffic generated the same alert, but is very different. It had me puzzled because it did not fit the same pattern as the other traffic. Analyzing the scan logs did not turn up anything, as I did not see any type of scans coming from the associated hosts.

Timestamp	Alert	SrcIP	SrcPort	DstIP	DstPort
04/22-14:19:29.583217	High port 65535 tcp - possible Red Worm - traffic	MY.NET.25.10	65535	65.222.188.7	25
04/22-	High port 65535 tcp - possible Red Worm -	MY.NET.25.10	65535	65.222.188.7	25

[22] FedoraForum.org

Timestamp	Alert	SrcIP	SrcPort	DstIP	DstPort
14:19:29.599019	traffic				
04/22-14:19:29.599165	High port 65535 tcp - possible Red Worm - traffic	MY.NET.25.10	65535	65.222.188.7	25
04/22-14:19:29.599307	High port 65535 tcp - possible Red Worm - traffic	MY.NET.25.10	65535	65.222.188.7	25
04/22-14:19:29.599464	High port 65535 tcp - possible Red Worm - traffic	MY.NET.25.10	65535	65.222.188.7	25
04/22-14:19:29.599515	High port 65535 tcp - possible Red Worm - traffic	MY.NET.25.10	65535	65.222.188.7	25

I believe this is legitimate SMTP traffic, possibly retries to deliver email to external mail servers. For example, the destination IP address in the table above resolves to the following:

```
UUNET Technologies, Inc. UUNET65 (NET-65-192-0-0-1)
                                 65.192.0.0 - 65.223.255.255
Public Citizen Inc. UU-65-222-188 (NET-65-222-188-0-1)
                                 65.222.188.0 - 65.222.188.255

# ARIN WHOIS database, last updated 2004-09-08 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Further digging using IP Lookup[23] resulted in the IP address belonging to `savgw.citizen.org`. I'm taking a guess, but this is most likely a valid GroupWise mail server for Citizen.org.

Peter Storm in his practical assignment stated that if this was the Red Worm/Adore infection scans for vulnerable systems would be seen from these machines. I did not see any signs of this in my analysis.

Recommendation: This Snort rule is most likely to general in that it only looks for traffic using port 65535. Tune this rule to be more specific to the signature of the Adore.worm or simply just drop this rule to reduce the amount of false positives. The Snort default rule set includes several rules that address the vulnerabilities in Linux systems associated with Adore.worm, which are: wuftpd, bind, lprng, and statd. Ensure that all Linux machines are patched and up to date to reduce exposure to vulnerabilities. An example Snort rule, that was in Peter Storm's practical that can be found on Whitehats, follows:

```
alert TCP $EXTERNAL any -> $INTERNAL 515 (msg: "IDS457/lpr_LPRng-
redhat7-overflow-security.is"; flags: A+; content: "|31DB 31C9 31C0
B046 CD80 89E5 31D2 B266 89D0 31C9 89CB|"; nocase; classtype: system-
attempt; reference: arachnids,457;)[24]
```

MY.NET.30.3 activity

This alert is generated by a custom Snort rule. The exact reason for this rule is unclear, unless its purpose is to monitor all traffic to this server. This rule seems

[23] SuperScan4
[24] Whitehats

to generate a great deal of alerts to be analyzed by the analyst. The majority of the alerts generated were port 524 traffic (86.41%). The majority of this traffic was associated with the following IP addresses: 131.92.177.18 – 5206 alerts, resolves to Army Information Systems Command – Aberdeen, 69.138.77.62 – 2457 alerts, another Comcast address, 151.196.115.104 – 2454 alerts, and 138.88.98.71 – 227 alerts, both of which resolve to Verizon Global Networks.

```
OrgName:    Army Information Systems Command - Aberdeen (EA)
OrgID:      AISCAE
Address:    AMSSB-SCI-N/BLDG E5234
City:       ABERDEEN PROVING GROUND
StateProv:  MD
PostalCode:
Country:    US

NetRange:   131.92.0.0 - 131.92.255.255
CIDR:       131.92.0.0/16
NetName:    APGEA-NET1
NetHandle:  NET-131-92-0-0-1
Parent:     NET-131-0-0-0-0
NetType:    Direct Assignment
NameServer: NS01.ARMY.MIL
NameServer: NS02.ARMY.MIL
NameServer: NS03.ARMY.MIL
Comment:
RegDate:    1988-11-01
Updated:    2001-08-09

TechHandle: RW943-ARIN
TechName:   Ward, Ronnie
TechPhone:  +1-410-436-4755
TechEmail:  RONNIE.WARD@sbccom.apgea.army.mil

# ARIN WHOIS database, last updated 2004-09-08 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.

**********************************************************************
Verizon Global Networks, Inc. VZGNI-PUB-1 (NET-138-88-0-0-1)
                                138.88.0.0 - 138.88.255.255
Verizon Internet Services VZ-DSLDIAL-RSTNVA-6 (NET-138-88-9-0-1)
                                  138.88.9.0 - 138.88.159.255

# ARIN WHOIS database, last updated 2004-09-08 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Since the IP address that belongs to the Army (Aberdeen) amounts to a third of the alerts, I would assume that this is valid traffic. IP addresses 69.138.77.62 and 151.196.115.104 was also associated with port 3019. This port is registered to Resource Manager, which is used to manage a wide range of resources. Besides port 80 traffic, other traffic observed on this host related to various ports associated with worm propagation; however, no response was observed in the alert and scan logs.

Recommendation: Much the same as MY.NET.30.4. Tuning this rule to log rather than alert will reduce the need to analyze normal traffic. Scan this machine to ensure that no signs of compromise exist. Again, a periodic vulnerability scan should be performed to ensure that only those services that are suppose to be running are the only services running.

SMB Name Wildcard

The "SMB Name Wildcard" alert is only related to SMB traffic leaving the internal network. There does not appear to be a Snort rule for this alert, therefore it must be another custom alert. My assumption, as the name implies, is that this is a "Wildcard" rule to catch all SMB traffic to external hosts. The top three sources in this alert are:

Source Address	# of Alerts
MY.NET.11.4	3108
MY.NET.11.7	2508
MY.NET.75.13	506

These three hosts, along with less significant traffic by MY.NET.11.5 and MY.NET.11.6, appear to be normal SMB traffic. SMB traffic is considered to be normal NetBIOS traffic with many Windows machines and some Linux nodes if configured as a Samba server or client. This traffic appears that this might be an attempt by the internal hosts to resolve NetBIOS information from the external host, possibly for P2P file sharing or IRC chat. Windows machines often exchange these queries when using Windows File and Print sharing to determine NetBIOS names when only IP addresses are known. Another reason for SMB traffic to outside sources may be a result of negotiations from connections such as identd requests from a mail server or IRC server.[25] IRC will be discussed in detail later.

For the most part it appears that MY.NET.11.4 talks to two external hosts, 210.120.128.117 and 129.254.25.169. These IP addresses resolve to the following:

```
% [whois.apnic.net node-1]
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html

inetnum:      210.120.128.0 - 210.120.191.255
netname:      DACOM-CHOLLIANRAS-KR
descr:        DACOM
descr:        706-1 Yeoksam-dong Kangnam-gu
descr:        SEOUL
descr:        135-610
country:      KR
admin-c:      KH206-AP
tech-c:       WS168-AP
mnt-by:       MAINT-KR-DACOM
status:       ASSIGNED NON-PORTABLE
remarks:      imported from KRNIC
changed:      hm-changed@apnic.net 20021025
source:       APNIC

person:       Kyunam Hwang
address:      DACOM
address:      706-1 Yeoksam-dong Kangnam-gu
address:      SEOUL
address:      135-610
```

[25] Whitehats

```
country:      KR
phone:        +82-2-6220-7462
fax-no:       +82-2-6220-0323
e-mail:       zprivacy1@chollian.net
nic-hdl:      KH206-AP
mnt-by:       MAINT-KR-DACOM
remarks:      imported from KRNIC
changed:      hm-changed@apnic.net 20021022
source:       APNIC

person:       Wangseok Shin
address:      DACOM
address:      706-1 Yeoksam-dong Kangnam-gu
address:      SEOUL
address:      135-610
country:      KR
phone:        +82-2-6220-7462
fax-no:       +82-2-6220-0323
e-mail:       zprivacy2@chollian.net
nic-hdl:      WS168-AP
mnt-by:       MAINT-KR-DACOM
remarks:      imported from KRNIC
changed:      hm-changed@apnic.net 20021022
source:       APNIC

*****************************************************************************
OrgName:    Internet Assigned Numbers Authority
OrgID:      IANA
Address:    4676 Admiralty Way, Suite 330
City:       Marina del Rey
StateProv:  CA
PostalCode: 90292-6695
Country:    US

NetRange:   169.254.0.0 - 169.254.255.255
CIDR:       169.254.0.0/16
NetName:    LINKLOCAL
NetHandle:  NET-169-254-0-0-1
Parent:     NET-169-0-0-0-0
NetType:    IANA Special Use
NameServer: BLACKHOLE-1.IANA.ORG
NameServer: BLACKHOLE-2.IANA.ORG
Comment:    Please see RFC 3330 for additional information.
RegDate:    1998-01-27
Updated:    2002-10-14

OrgAbuseHandle: IANA-IP-ARIN
OrgAbuseName:   Internet Corporation for Assigned Names and Number
OrgAbusePhone:  +1-310-301-5820
OrgAbuseEmail:  abuse@iana.org

OrgTechHandle: IANA-IP-ARIN
OrgTechName:   Internet Corporation for Assigned Names and Number
OrgTechPhone:  +1-310-301-5820
OrgTechEmail:  abuse@iana.org

# ARIN WHOIS database, last updated 2004-09-08 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Other NetBIOS traffic to IP addresses assigned to IANA are IP addresses: MY.NET.11.5, MY.NET.11.6, and MY.NET.11.7. I assume that these hosts are running DHCP since the IP addresses assigned to IANA are invalid IP addresses for the Internet, according to RFC 3330.[26] RFC 3330 states, “This is the "link local" block. It is allocated for communication between hosts on a single link.

[26] Faqs.org

Hosts obtain these addresses by auto-configuration, such as when a DHCP server may not be found."

However, there are several hosts that do not fit what is "normal" NetBIOS traffic. Three of the top sources of this are the following:

Source Address	# of Alerts
MY.NET.150.44	632
MY.NET.150.198	435
MY.NET.43.14	220

Timestamp	Alert	SrcIP	SrcPort	DstIP	DstPort
04/20-14:13:47.325546	SMB Name Wildcard	MY.NET.150.44	1058	200.81.6.20	137
04/20-14:14:01.825863	SMB Name Wildcard	MY.NET.150.44	1058	200.81.6.20	137
04/20-14:14:25.388745	SMB Name Wildcard	MY.NET.150.44	1058	200.81.6.20	137
04/20-14:14:59.827571	SMB Name Wildcard	MY.NET.150.44	1058	200.81.6.20	137
04/20-14:19:35.139906	SMB Name Wildcard	MY.NET.150.44	1058	216.118.120.35	137
04/20-14:19:54.035203	SMB Name Wildcard	MY.NET.150.44	1058	216.118.120.35	137
04/20-14:20:52.036370	SMB Name Wildcard	MY.NET.150.44	1058	216.118.120.35	137

I assumed that this traffic was the same traffic that Peter Storm described in his practical, but what I did not see were entries in the scan logs to correlate the data.

Further analysis showed that the "SMB Name Wildcard" alert is possibly the result of stimulus from external sources.

Timestamp	Alert	SrcIP	SrcPort	DstIP	DstPort
04/20-22:55:24.352259	SMB Name Wildcard	MY.NET.150.44	1058	218.154.73.202	137
04/20-22:55:24.659901	EXPLOIT x86 NOOP	218.154.73.202	1160	MY.NET.150.44	80
04/20-22:55:24.909182	EXPLOIT x86 NOOP	218.154.73.202	1160	MY.NET.150.44	80
04/20-22:55:24.910383	EXPLOIT x86 NOOP	218.154.73.202	1160	MY.NET.150.44	80
04/20-22:55:24.911381	EXPLOIT x86 NOOP	218.154.73.202	1160	MY.NET.150.44	80
04/20-22:55:25.406728	SMB Name Wildcard	MY.NET.150.44	137	218.154.73.202	137
04/20-22:55:25.411404	EXPLOIT x86 NOOP	218.154.73.202	1160	MY.NET.150.44	80
04/20-22:55:25.413471	EXPLOIT x86 NOOP	218.154.73.202	1160	MY.NET.150.44	80
04/20-22:55:25.416618	EXPLOIT x86 NOOP	218.154.73.202	1160	MY.NET.150.44	80
04/20-22:55:25.417727	EXPLOIT x86 NOOP	218.154.73.202	1160	MY.NET.150.44	80
04/20-22:55:25.659325	EXPLOIT x86 NOOP	218.154.73.202	1160	MY.NET.150.44	80
04/20-22:55:25.660389	EXPLOIT x86 NOOP	218.154.73.202	1160	MY.NET.150.44	80
04/20-22:55:25.661926	EXPLOIT x86 NOOP	218.154.73.202	1160	MY.NET.150.44	80
04/20-22:55:25.663104	EXPLOIT x86 NOOP	218.154.73.202	1160	MY.NET.150.44	80
04/20-22:55:28.276067	SMB Name Wildcard	MY.NET.150.44	1058	218.154.73.202	137
04/20-22:55:28.400723	SMB Name Wildcard	MY.NET.150.44	137	218.154.73.202	137

Timestamp	Alert	SrcIP	SrcPort	DstIP	DstPort
04/20-22:55:42.776302	SMB Name Wildcard	MY.NET.150.44	1058	218.154.73.202	137
04/20-22:56:42.589804	SMB Name Wildcard	MY.NET.150.44	1058	218.154.73.202	137

It appears that this traffic might after all be "normal." I assume that the IP address MY.NET.150.44 is a web server for research purposes that is used for file sharing. This address resolved to illiad.lib.university.edu. MY.NET.150.198, which was characteristic of the same traffic, resolves to pharos2.lib.university.edu.

Recommendation: This rule should be modified to look for inbound SMB traffic. Routers and firewalls should also be configured with ACLs and firewall rules to deny any outbound NetBIOS traffic. There is no legitimate reason for NetBIOS traffic to leave the boundaries of the network.

Tiny Fragments – Possible Hostile Activity

This alert is caused by the minifrag preprocessor: 128. The minfrag option checks the size of IP fragments. If a fragment is smaller than a set threshold value, an alert is generated.[27] Fragmentation small enough to trigger the preprocessor should not occur normally. However, it appears that with correlation of the scan logs, this is mostly the result of port scanning attempts using crafted packets by the IP address 209.164.32.205, which in this five-day period accounted for the third most alerts.

```
OrgName:    XO Communications
OrgID:      XOXO
Address:    Corporate Headquarters
Address:    11111 Sunset Hills Road
City:       Reston
StateProv:  VA
PostalCode: 20190-5339
Country:    US

ReferralServer: rwhois://rwhois.eng.xo.com:4321/

NetRange:   209.164.0.0 - 209.164.63.255
CIDR:       209.164.0.0/18
NetName:    XOXO-BLK-18
NetHandle:  NET-209-164-0-0-1
Parent:     NET-209-0-0-0-0
NetType:    Direct Allocation
NameServer: NAMESERVER.CONCENTRIC.NET
NameServer: NAMESERVER1.CONCENTRIC.NET
NameServer: NAMESERVER2.CONCENTRIC.NET
NameServer: NAMESERVER3.CONCENTRIC.NET
Comment:    For best results, please send all spam and worm reports only to abuse@xo.com.
RegDate:    1997-11-14
Updated:    2003-08-08

OrgAbuseHandle: XCNV-ARIN
OrgAbuseName:   XO Communications, Network Violations
OrgAbusePhone:  +1-866-285-6208
OrgAbuseEmail:  abuse@xo.com
```

[27] Roesch, Marty

```
OrgTechHandle: XCIA-ARIN
OrgTechName:   XO Communications, IP Administrator
OrgTechPhone:  +1-703-547-2000
OrgTechEmail:  ipadmin@eng.xo.com

# ARIN WHOIS database, last updated 2004-09-07 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

This alert was also generated by IP addresses 212.41.93.144, 61.19.223.227, 61.216.77.135, and 200.221.157.175. Since 61.216.77.135 was one of the two major attackers, the following Whois results are listed. The other IP addresses were from an address pool out of Switzerland and a pool from Latin American and Caribbean Addresses.

```
% [whois.apnic.net node-1]
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html

inetnum:      61.216.0.0 - 61.219.255.255
netname:      HINET-TW
descr:        CHTD, Chunghwa Telecom Co.,Ltd.
descr:        Data-Bldg.6F, No.21, Sec.21, Hsin-Yi Rd.
descr:        Taipei Taiwan 100
country:      TW
admin-c:      HN27-AP
tech-c:       HN28-AP
remarks:      Delegated to HiNet for ADSL subscriber.
remarks:      This information has been partially mirrored by APNIC from
remarks:      TWNIC. To obtain more specific information, please use the
remarks:      TWNIC whois server at whois.twnic.net.
mnt-by:       MAINT-TW-TWNIC
changed:      hostmaster@twnic.net 20010117
status:       ALLOCATED PORTABLE
source:       APNIC

person:       HINET Network-Adm
address:      CHTD, Chunghwa Telecom Co., Ltd.
address:      Data-Bldg. 6F,  No. 21, Sec. 21, Hsin-Yi Rd.,
address:      Taipei Taiwan 100
country:      TW
phone:        +886 2 2322 3495
phone:        +886 2 2322 3442
phone:        +886 2 2344 3007
fax-no:       +886 2 2344 2513
fax-no:       +886 2 2395 5671
e-mail:       network-adm@hinet.net
nic-hdl:      HN27-AP
remarks:      same as TWNIC nic-handle HN184-TW
mnt-by:       MAINT-TW-TWNIC
changed:      hostmaster@twnic.net 20000721
source:       APNIC

person:       HINET Network-Center
address:      CHTD, Chunghwa Telecom Co., Ltd.
address:      Data-Bldg. 6F,  No. 21, Sec. 21, Hsin-Yi Rd.,
address:      Taipei Taiwan 100
country:      TW
phone:        +886 2 2322 3495
phone:        +886 2 2322 3442
phone:        +886 2 2344 3007
fax-no:       +886 2 2344 2513
fax-no:       +886 2 2395 5671
e-mail:       network-center@hinet.net
nic-hdl:      HN28-AP
remarks:      same as TWNIC nic-handle HN185-TW
mnt-by:       MAINT-TW-TWNIC
changed:      hostmaster@twnic.net 20000721
source:       APNIC
```

```
inetnum:      61.216.0.0 - 61.216.255.255
netname:      HINET-NET
descr:        CHTD, Chunghwa Telecom Co., Ltd.
descr:        Data-Bldg. 6F,  No. 21, Sec. 21, Hsin-Yi Rd.,
descr:        Taipei Taiwan
country:      TW
admin-c:      CYK-TW
tech-c:       CYK-TW
mnt-by:       MAINT-TW-TWNIC
remarks:      This information has been partially mirrored by APNIC from
remarks:      TWNIC. To obtain more specific information, please use the
remarks:      TWNIC whois server at whois.twnic.net.
changed:      fkchung@ms1.hinet.net 20010117
status:       ASSIGNED NON-PORTABLE
source:       TWNIC

person:       Chung Yung Kang
address:      Chunghwa Telecom Data communication Business Group
address:      No.21, Hsin-Yi Rd., sec. 1
address:      Taipei Taiwan
country:      TW
phone:        +886-2-2322-3442
fax-no:       +886-2-2344-2513
e-mail:       cykang@ms1.hinet.net
nic-hdl:      CYK-TW
remarks:      This information has been partially mirrored by APNIC from
remarks:      TWNIC. To obtain more specific information, please use the
remarks:      TWNIC whois server at whois.twnic.net.
changed:      hostmaster@twnic.net 19990924
source:       TWNIC
```

Recommendation: Based on the assumption that these are malicious attempts to gain access to the network, IP addresses 209.164.32.205 and 61.216.77.135 should be blocked at the border by the use of ACLs and denied by the firewall. Since the remaining IP addresses only accounted for a couple of alerts each and destined for one IP address, these should be monitored for any other attempts to access the network or blocked. The machines that were targeted should be removed from the network and scanned for signs of compromise.

RFB – Possible WinVNC – 010708 – 1

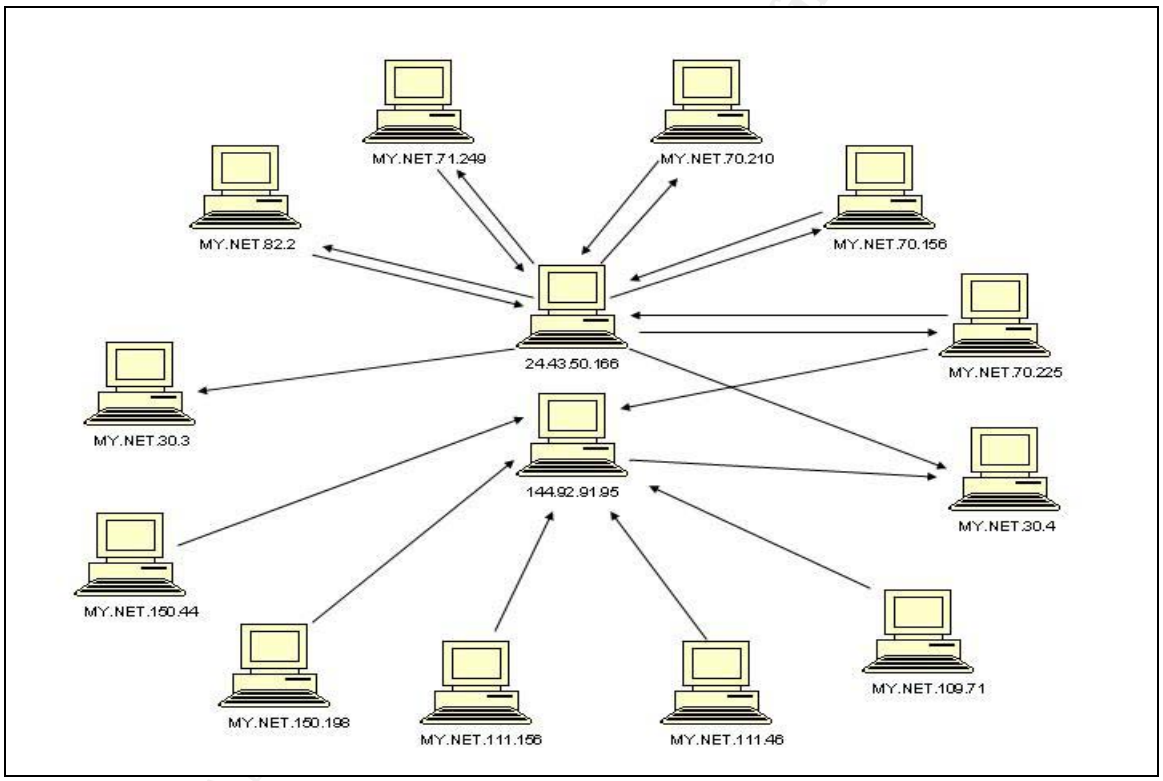
VNC stands for Virtual Network Computing. Basically, WinVNC is a remote desktop application that allows you to view a remote desktop from anywhere on the Internet.

This alert was generated 2355 times by several hosts. Analysis shows that these hosts are running as VNC servers to external hosts. The University might allow this application for educational purposes, but there are several security issues with WinVNC.

WinVNC provides encrypted authentication at login; however, data transfer after login is unencrypted. Everything that I could find on WinVNC, assures the user that local logon is a must in order for a client to login. I would be interested in confirming this. There is a known vulnerability that will allow remote access via a

buffer overflow to a client. Also, there is another vulnerability that will allow someone with local access to exploit a buffer overflow to add new users.[28]

The following link graph shows an example of the possible WinVNC traffic that was captured during this five-day period. MY.NET.30.3 and MY.NET.30.4 traffic appear to attempts by the external hosts to connect to these machines. I did not see and traffic originating from these internal hosts. MY.NET.150.44 and MY.NET.150.198 traffic is an "SMB Name Wildcard" alert in response to scans from IP address 144.92.91.95. All the traffic back and forth appears to be associated with 24.43.50.166. Only four alerts were associated with 144.92.91.95, which are most likely a response to an extensive scan of the entire network by this host.



IP address 24.43.50.166 resolves to the following:

```
Rogers Cable Inc. ROGERS-CAB-3 (NET-24-42-0-0-1)
                                24.42.0.0 - 24.43.255.255
Rogers Cable Inc. Lndn ON-ROG-LDN-18 (NET-24-43-48-0-1)
                                24.43.48.0 - 24.43.51.255

# ARIN WHOIS database, last updated 2004-09-08 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Recommendation: Continue to monitor this type of traffic. If this is indeed WinVNC traffic and is authorized, consider tuning this rule to log rather than alert.

[28] ICAT

Other Snort rules should alert to malicious activity. Also, like with remote management, consider the use of encrypted channels to prevent unauthorized access. For example, a SSH tunnel to a bastion host that will only allow authorized clients to the server.[29] A good example of how to secure WinVNC through the use of SSH is documented on the Bitvise website. If this is not authorized activity, remove those hosts from the network and remove WinVNC services. Scan for other signs of compromise before connecting back to the network.

Scans

Nearly 14 million scan alerts were analyzed for the five-day period. Of the scan alerts, 99.90% of those are associated with SYN Scans and UDP Scans. The major concern is that 93% of all scans were from internal hosts.

Scan Summary

Scan Type	# of Alerts	% of All Scan Alerts
SYN Scan	8259149	59.78%
UDP Scan	5542620	40.12%
FIN Scan	9888	0.07%
UNKNOWN Scan	1974	0.014%
NULL Scan	929	0.007%
INVALIDACK Scan	660	0.005%
NOACK Scan	630	0.005%
VECNA Scan	276	0.002%
XMAS Scan	43	0.0003%
FULLXMAS Scan	20	0.0001%
SPAU Scan	12	0.00006%
SYNFIN Scan	8	0.00005%
NMAPID Scan	7	0.00005%

The scan logs were used extensively in the analysis of the alert logs. Since the SYN Scans and UDP Scans were responsible for the majority of the scans, I will analyze them in further detail.

SYN Scans

Internal hosts generated approximately 91% of the SYN scan alerts, with the majority of the destinations to external hosts. The following table is a list of the top five sources of the SYN Scan.

Source Address	# of SYN Scan Alerts	Destination Ports
MY.NET.17.45	1179172	2745,135,1025,445,3127,6129,139, 80
MY.NET.80.224	910226	445,3127,6129,139,3410,5000,2745,135,1025
MY.NET.112.189	713578	135

[29] Norton, Ed

MY.NET.81.39	694877	135
MY.NET.111.51	553549	445,139, 1025, 6129, 135,2745, 3410, 3127

The most probable cause of this traffic is that the machines are infected with a virus and it is trying to propagate. The top two and number five all show signs of propagation of the Backdoor.Agobot. The Agobot worm is an IRC-controlled backdoor with network spreading capabilities that allow its authors to gain control over computers and link them into P2P networks. These networks, in turn, can be used to send large amounts of spam e-mail messages or to flood Web sites with data.[30]

The other two are characteristic of the W32.Blaster worm that propagates rapidly via port 135 to exploit RPC DCOM on unpatched systems.

Recommendation: The University should disconnect these machines from the network and scan them with antivirus software with the current DAT files. Completely disinfect the machines before connecting the machines to the network. A complete review of all hosts to look for signs of infection is recommended. Additionally, the University should block all unnecessary ports except for those that are needed.

UDP Scans

Again, approximately 90% of the UDP Scans were generated by internal sources. Although the top five sources only account for 38% of the alerts, there were several more sources that generated around 100000 alerts each. With the exception of MY.NET.1.3 and MY.NET.1.4, the majority of the traffic was what appeared to be the result of more P2P file sharing by internal hosts. The following table shows the top five sources.

Source Address	# of UDP Scan Alerts	Destination Ports
MY.NET.1.3	3629991	53
MY.NET.1.4	1072082	53
MY.NET.69.232	221506	Numerous ephemeral
MY.NET.69.214	167296	Numerous ephemeral
MY.NET.53.225	117977	Numerous ephemeral

Recommendation: The University should consider modifying the preprocessor options to a higher threshold. This will reduce the amount of alerts generated if file sharing is accepted practice on the network. Also, pass statements can be used for the DNS servers. DNS servers generate a large amount of scans when querying for name resolution. The pass statement will reduce the amount of alerts generated by this normal DNS traffic.

[30] eTrust PestPatrol

Out of Spec (OOS) Logs

There were only a total of 5098 OOS logs for the five day period analyzed. I am not sure if this is because of insufficient logging or due to prior recommendations in previous practical assignments. However, out of those logs 99.3% were generated by external sources. These alerts were generated by illegal flags or ECN bits set in the TCP packets. The following table lists the top five destination hosts with associated ports.

Destination Address	# of OOS Alerts	Destination Ports
MY.NET.6.7	1685	80, 110
MY.NET.12.6	1352	25
MY.NET.24.44	453	80
MY.NET.5.67	156	8080
MY.NET.24.34	92	80

A large amount of the OOS alerts were generated by P2P traffic. Several common file-sharing ports were noted, for example 6881-6888. The majority of these alerts were generated because of what Snort sees as illegal bits, which are now known as ECN and CWR bits. This is a new concept that is explained in RFC 2481 as a type of flow control to prevent bottlenecks.[31]

It appears for the most part, this is legitimate traffic to the above IP addresses based on the fact that these machines are web servers. MY.NET.12.6 is most likely the SMTP server for the network.

Analysis Process

I began the analysis process be reading several practical assignments to determine the best approach. After reading a couple of documents, I determined that I would not try my hand at SnortSnarf or ACID based on the problems documented by these individuals.

The alert logs were concatenated into one large file to add continuity to the alerts. I comma delimited the file using a perl script often recommended and borrowed from Tod Beardsley. I also used the other script that was posted by Tod Beardsley to generate an alert summary of the data.

I found another useful set of scripts that were written by James Maher. The alert-reports.pl script generated some very nice reports to perform early analysis. This script produces lists like: Hits per Source Address, Hits per Destination Address, and Attack/Scan Types. There is also a nice set of options to tailor the output.

I must say that it would have been hard to get started had it not been for the summaries provided by the excellent written scripts of the aforementioned gentlemen. I have included these scripts in Appendix A. Once I had all the data

[31] IETF

I wanted, I began the brutal task of querying the Access database for more information.

I followed the same process for the scan logs. I concatenated the scan logs into one large file. Did I say large? Once again, I used Tod Beardsley's csv.pl script to parse the file. As soon as I did that, I realized that I was going to have trouble transferring the file to CD to import it into the Access database. I went back and parsed each log separately and then imported it into Access. I lost several files doing that, but I still had plenty to play with.

I couldn't find a script to parse the oos logs that worked so I resorted to manually removing parts of the logs in order to import them into an Excel spreadsheet. I used several sorting options and functions to count in order to analyze these logs.

I must mention here that along with Internet sources, I used several previously submitted practical assignments to correlate what I saw in the analysis. If they were not specifically mentioned in the text of this report, they have been mentioned in the references.

Equipment Used

Computer 1
HP Laptop
1800+
512K RAM
Windows XP
Microsoft Access XP
Microsoft Excel XP
Microsoft Word XP
SuperScan 4 (FoundStone tool that simplifies Whois queries and IP Lookups)

Computer 2
Toshiba Laptop
1.8 GHz
512K RAM
SuSE 9.1 Linux
Snort v2.1.3

References (Assignment #2)

CERT. "CERT® Advisory CA-2001-30 Multiple Vulnerabilities in lpd." URL: http://www.cert.org/advisories/CA-2001-30.html (July 14, 2004)

CERT. “CERT® Advisory CA-2001-32 HP-UX Line Printer Daemon Vulnerable to Directory Traversal.” URL: http://www.cert.org/advisories/CA-2001-32.html (July 14, 2004)

CERT. “CERT® Incident Note IN-2004-01.” URL: http://www.cert.org/incident_notes/IN-2004-01.html. (July 16, 2004)

Common Vulnerabilities and Exposures. “CVE-1999-0009 Inverse query buffer overflow in BIND 4.9 and BIND 8.” URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0009 (July 13, 2004)

Common Vulnerabilities and Exposures. “CVE-1999-0833 Buffer overflow in BIND 8.2 via NXT records.” URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0833 (July 14, 2004)

Common Vulnerabilities and Exposures. “CVE-2001-0010 Buffer overflow in transaction signature (TSIG) handling code in BIND 8 allows remote attackers to gain root privileges.” URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0010 (July 13, 2004)

Common Vulnerabilities and Exposures. “CVE-2001-0011 Buffer overflow in nslookupComplain function in BIND 4 allows remote attackers to gain root privileges.” URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0011 (July 13, 2004)

Common Vulnerabilities and Exposures. “CVE-2002-0068” URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0068 (July 17, 2004)

Common Vulnerabilities and Exposures. “CAN-2004-0315” URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0315 (July 16, 2004)

IETF. “Transmission Control Protocol.” Sept. 1981. URL: http://www.ietf.org/rfc/rfc793.txt (Aug. 4, 2004)

LinkLogger. “TCP Port 3127” April 30, 2004 URL: http://www.linklogger.com/TCP3127.htm (Sept. 2, 2004)

Network Associates, Inc. “Backdoor-Q” McAfee. URL: http://vil.nai.com/vil/content/v_100468.htm (July 15, 2004)

Riddell, Trent. “Detect #1 – Broadcast to Printer Port.” GIAC Certification Practical Assignment version 3.0. Sans.org. URL: http://www.giac.org/practical/Trenton_Riddell_GCIA.doc (July 14, 2004)

Robertson, Paul D. “Re: Probe from 255.255.255.255?” The Shmoo Group. Jun 1, 2004. URL: http://www.shmoo.com/mail/firewalls/jun01/msg00006.shtml (July 15, 2004)

Snort Signature Database. “DNS named version attempt.” Snort.org URL: http://www.snort.org/snort-db/sid.html?sid=257 (July 13, 2004)

Symantec Security Response. “W32.Mydoom.A@mm.” July 27, 2004 URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.novarg.a@mm.html (Aug. 10, 2004)

Symantec Security Response. “W32.Mydoom.B@mm.” July 27, 2004 URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.mydoom.b@mm.html (Aug. 10, 2004)

Viner, Dennis. “[unisog] Increased Port 3128/1080 Scanning.” Sans.org. Mar. 5, 2004. URL: http://lists.sans.org/pipermail/unisog/2004-March/006955.php (July 17, 2004)

Zirkle, Laurie. “May 31, 2001 Probes.” DShield.org. Jun 1, 2001 URL: http://www.dshield.org/pipermail/intrusions/2001-June/000492.php (Aug. 10, 2004)

References (Assignment #3)

Ackerman, Richard. “TCP/IP Ports” URL: http://www.chebucto.ns.ca/~rakerman/port-table.html (Sept. 11, 2004)

Beardsley, Tod. “Intrusion Detection and Analysis: Theory, Techniques, and Tools.” GIAC.org. Mar 15, 2002. URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc (Aug. 31, 2004)

BitTorrent. “BitTorrent protocol specifications.” URL: http://bitconjurer.org/BitTorrent/protocol.html (Sept. 6, 2004)

Bitvise. “Securing WinVNC Using SSH Connection Tunneling” URL: http://www.bitvise.com/winvnc.html (Sept. 5, 2004)

Dshield. “Port Report.” Port 25. URL: http://www.dshield.org/port_report.php?port=25&recax=1&tarax=2&srcax=2&percent=N&days=40 (Sept. 3, 2004)

Dshield. “Port Report.” Port 113. URL: http://www.dshield.org/port_report.php?port=113&recax=1&tarax=2&srcax=2&percent=N&days=40 (Sept. 6, 2004)

Dshield. “Port Report.” Port 135. URL: http://www.dshield.org/port_report.php?port=135&recax=1&tarax=2&srcax=2&percent=N&days=40 (Sept. 7, 2004)

Dshield. “Port Report.” Port 3127. URL: http://www.dshield.org/port_report.php?port=3127&recax=1&tarax=2&srcax=2&percent=N&days=40 (Sept. 7, 2004)

Dshield. “Port Report.” Port 6129. URL: http://www.dshield.org/port_report.php?port=6129&recax=1&tarax=2&srcax=2&percent=N&days=40 (Sept. 7, 2004)

Dshield. “Port Report.” Port 6881. URL: http://www.dshield.org/port_report.php?port=6881&recax=1&tarax=2&srcax=2&percent=N&days=40 (Sept. 6, 2004)

eTrust PestPatrol – Pest Encyclopedia. “Backdoor.Agobot” URL: http://www.pestpatrol.com/pestinfo/b/backdoor_agobot.asp (Sept. 8, 2004)

Evans, Eric. “Intrusion Detection In-Depth: GCIA Practical Assignment, v3.4.” Dec. 11, 2003. URL: http://www.giac.org/practical/GCIA/Eric_Evans_GCIA.pdf (Sept. 10, 2004)

Extreme Networks. “Recommendations for Virus Protection from Extreme Networks®.” URL: http://www.extremenetworks.com/services/news/article4.asp (Sept. 8, 2004)

Faske, Mark. “GCIA Certified Intrusion Analyst (GCIA) Practical Assignment.” GIAC.org. Dec. 7, 2003. URL: http://www.giac.org/practical/GCIA/Mark_Faske_GCIA.pdf (Sept. 11, 2004)

Faqs.org. “RFC 3330.” Sept. 2002. URL: http://www.faqs.org/rfcs/rfc3330.html (Sept. 9, 2004)

FedoraForum.org. “Azureus BitTorrent, port 6881 security question” URL: http://www.fedoraforum.org/forum/archive/index.php/t-20720.html (Sept. 6, 2004)

Gladiator Security Forum. “Emergency Processing Report for Sasser Worm and its variants.” May 12, 2004. URL: http://forum.gladiator-antivirus.com/index.php?showtopic=14534 (Sept. 9, 2004)

Hughes, Timothy. “Re: 1501?” ADSM.org. Jul 7, 2001. URL: http://msgs.adsm.org/cgi-bin/get/adsm0407/19/1.html (Sept. 8, 2004)

ICAT Metabase. “CAN-2001-0167: Buffer overflow in AT&T WinVNC.” NIST. May 5, 2001. URL: http://icat.nist.gov/icat.cfm?cvename=CAN-2001-0167 (Sept. 7, 2004)

ICAT Metabase. “CAN-2002-0971: Vulnerability in VNC, TightVNC, and TridiaVNC.” NIST. Sept. 24,2002. http://icat.nist.gov/icat.cfm?cvename=CAN-2002-0971 (Sept. 7, 2004)

Internet Storm Center. “Port History.” April 20, 2004. SANS.org. URL: http://isc.sans.org/port_report.php?l=20&a=0&s=records&d=desc&date_month=04&date_day=20&date_year=2004 (Sept. 10, 2004)

Microsoft. “Windows Systems Resource Manager.” URL: http://www.microsoft.com/windowsserver2003/downloads/wsrm.mspx (Sept. 8, 2004)

Montcalm, Erik. “GCIA Practical Assignment Version 3.3.” Nov. 12, 2003. URL: http://www.giac.org./practical/GCIA/Erik_Montcalm_GCIA.pdf (Sept. 10, 2004)

Norton, Ed. “Safe® WinVNC.” OIT Security and Assurance. Sept. 12, 2002. URL: http://www1.umn.edu/oit/img/assets/5630/SafeWinVNC.pdf (Sept. 7, 2004)

Novell. “Port Number Assignments.” URL: http://www.novell.com/documentation/nw6p/index.html?page=/documentation/lg/nw6p/adminenu/data/aclkn27.html (Sept. 9, 2004)

NSA. “Windows 2000 Security Recommendation Guides.” Jan. 16, 2004. URL: http://nsa2.www.conxion.com/win2k/download.htm (Sept. 5, 2004)

Que. “Novell NetWare 6.5 Management Tools.” URL: http://www.quepublishing.com/articles/article.asp?p=102274&seqNum=12 (Sept. 8, 2004)

Ramakrishnan, K. and S. Floyd. “A Proposal to add Explicit Congestion Notification (ECN) to IP.” IETF. Jan. 1999 URL: http://www.ietf.org/rfc/rfc2481.txt?number=2481 (Sept. 8, 2004)

Randier, Sylvain. “GCIA Practical Assignment.” URL: http://www.giac.org/practical/GCIA/Sylvain_Randier_GCIA.pdf (Sept. 10, 2004)

Roesch, Marty. “Re: [snort] Tiny Fragments.” May 14, 2000 URL: http://archives.neohapsis.com/archives/snort/2000-05/0103.html (Sept. 6, 2004)

SANS. “Adore Worm” Version 0.8 April 12, 2001. Sans.org. URL: http://www.sans.org/y2k/adore.htm (Sept. 7, 2004)

Schultz, Greg. “GCIA Practical Assignment.” Dec. 18, 2003. URL: http://www.giac.org/practical/GCIA/Greg_Schultz_GCIA.pdf (Sept. 10, 2004)

Snort.org. “Snort Rules Database.” URL: http://www.snort.org/snort-db/ (Sept. 10, 2004)

Snort.org. “Snort Signature Database.” URL: http://www.snort.org/snort-db/sid.html?sid=648 (Sept. 9, 2004)

Snort.org. “Snort Signature Database.” URL: http://www.snort.org/snort-db/sid.html?sid=1394 (Sept. 9, 2004)

StillSecure. “Safe Access.” URL: http://www.stillsecure.com/products/sa/ (Sept. 8, 2004)

StillSecure. “VAM.” URL: http://www.stillsecure.com/products/vam/ (Sept. 8, 2004)

Storm, Peter H. “GIAC Certification Practical Assignment.” SANS.org. Nov. 15, 2003. URL: http://www.giac.org/practical/GCIA/Pete_Storm_GCIA.pdf (Sept. 11, 2004)

Sun Microsystems. “ Solaris 9 Resource Manager Data Sheet.” URL: http://wwws.sun.com/software/solaris/ds/ds-srm/ (Sept. 8, 2004)

Symantec Security Response. “Backdoor.Gaobot” Aug. 6, 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/backdoor.gaobot.html (Sept. 7, 2004)

Symantec Security Response. “Linux.Adore.Worm” April 15, 2001. URL: http://securityresponse.symantec.com/avcenter/venc/data/linux.adore.worm.html (Sept. 7, 2004)

Symantec Security Response. “W32.Blaster.worm” Aug. 11, 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html (Sept. 7, 2004)

Symantec Security Response. “W32.Beagle.J@mm” Mar. 2, 2004. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.beagle.j@mm.html (Sept. 7, 2004)

Symantec Security Response. “W32.Opaserv.Worm” July 29, 2004. URL:

http://securityresponse.symantec.com/avcenter/venc/data/w32.opaserv.worm.html (Sept. 8, 2004)

Whitehats Network Security Resource. URL: http://www.whitehats.com/ids/ (Sept. 9, 2004)

Whitehats Network Security Resource. "IDS177 "*NETBIOS-NAME-QUERY*"." arachnids. URL: http://www.whitehats.com/info/IDS177 (Sept. 9, 2004)

Whitehats Network Security Resource. "IDS457 "*LPRNG-REDHAT7-OVERFLOW-SECURITY.IS*"." arachnids. URL: http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids457&view=signatures (Sept. 9, 2004)

Appendix A – Perl Scripts

csv.pl – A perl script used to format alert logs into .csv files for importing into a database. (Courtesy of Tod Beardsley)

```
#!/cygdrive/c/Perl/bin/perl.exe -w

# Name: csv.pl

# Reads in a Snort -A Fast style alert log which for some
# reason wasn't generated as CSV, and make it as such.
#
# Usage: csv.pl infile [outfile]

unless ($ARGV[0]) {
  print "Need an input file!\n";
  die "(Hint: go to http://www.research.umbc.edu/~andy and get one)\n";
}

unless ($ARGV[1]) {
  $outfile = "$ARGV[0].csv";
} else {
  $outfile = "$ARGV[1]";
}

open(INFILE,"$ARGV[0]") || die "Can't open $ARGV[0] for reading!\n";
open(OUTFILE,">$outfile") || die "Can't open $ARGV[1] for writing!\n";

print "Transforming $ARGV[0] into $outfile.\n";
print "Just a moment.";

@calendar=qw(Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec);



while (<INFILE>) {
  next unless /(\w{1,3}\.){2}(\d{1,3}\.\d{1,3})/;    # Skip lines missing IPv4 IPs.
  next if /spp_portscan/;                            # Skip portscan notifications.
  chomp;
  if (/ \[\*\*\] /) {                                # Alert report.

  ($date_and_time,$alert,$src_and_dst) = split(/\s+\[\*\*\]\s/);
  ($date,$time) = split(/-/,$date_and_time);
  ($month_number,$day) = split(/\//,$date);
```

```
  $month = $calendar[$month_number-1];
  ($src,$dst) = split(/\s-\>\s/,$src_and_dst);
  ($src_ip,$src_port) = split(/:/,$src);
  ($dst_ip,$dst_port) = split(/:/,$dst);
  $snort_entry="ALERT" ;

  } else {                                          # Scan report.
  ($month,$day,$time,$src,$arrow,$dst,$alert,$flags) = split;
  undef $arrow;
  ($src_ip,$src_port) = split(/:/,$src);
  $alert = "$alert scan (Internally-based)" if $src_ip =~ /^MY\.NET/;
  $alert = "$alert scan (Externally-based)" unless $src_ip =~ /^MY\.NET/;
  ($dst_ip,$dst_port) = split(/:/,$dst);
  $snort_entry="SCAN" ;
}

  print OUTFILE "$snort_entry,";
  print OUTFILE "$month,$day,$time,$alert,";
  print OUTFILE "$src_ip,";
  print OUTFILE "$src_port" if $src_port;
  print OUTFILE "None" unless $src_port;
  print OUTFILE ",";
  print OUTFILE "$dst_ip";
  print OUTFILE ",";
  print OUTFILE "$dst_port" if $dst_port;
  print OUTFILE "," if $flags;
  print OUTFILE "None," unless $dst_port;
  print OUTFILE "$flags" if $flags;
  print OUTFILE "\n";

    $happydots++;
    print "." if $happydots % 100 == 0; # if $happydots == 100;
    print "Just a moment." if $happydots % 46600 == 0;
}
```

summarize.pl – A perl script used to generate an alert summary of the data. (Courtesy of Tod Beardsley)

```
#!/cygdrive/c/Perl/bin/perl.exe

# Name: summarize.pl

# Take a source file (generated by csv.pl) and summarize the contents,
# grouping alerts in a variety of ways we care about. This code absolutely
# could be and should be optomized by a real perl hacker.

# Usage: summarize.pl infile [outfile]

unless ($ARGV[0]) {
  print "Need an input file!\n";
  print "(Hint: go to http://www.research.umbc.edu/~andy and get one)\n";
  die "(Hint2: Don't forget to turn it into CSV and drop the portscans.)\n";
}

unless ($ARGV[1]) {                  # Check for a specified output file.
  if ($ARGV[0] =~ /\.csv$/ ) {       # If it's *.csv, autogenerate the output
     $outfile = $`."-summary.txt";   # filename. (Could be seen as unfriendly.)
  }
  } else {
  $outfile = "$ARGV[1]";
}

open(INFILE,"$ARGV[0]")  || die "Can't open $ARGV[0] for reading!\n";
open(OUTFILE,">$outfile") || die "Can't open $outfile for writing!\n";

print "Counting up all the Events of Interest in $ARGV[0].\nJust a moment.";

while (<INFILE>) {
chomp;
if ( (split(/\,/,$_))[0] eq "ALERT") {
```

```
 ($snort_type,$month,$day,$time,$alert,
  $src_ip,$src_port,$dst_ip,$dst_port) = (split(/\,/,$_));
 $date = "$month/$day";
} else {
 ($snort_type,$month,$day,$time,$alert,
  $src_ip,$src_port,$dst_ip,$dst_port,$flags) = (split(/\,/,$_));
 $date = "$month/$day";
}

# Frequency analysis on all that junk up there.


$date_counter{"$date"}++;
$alert_counter{"$alert"}++;


  if ($src_ip =~ "^MY\.NET") {
        $internal_src_ip_counter{"$src_ip"}++;
        $internal_src_port_counter{"$src_port"}++;

        if ($dst_ip =~ "^MY\.NET") {
                $internal_internal_relationship_counter{"$src_ip"."->"."$dst_ip"}++;
        } else {
                $internal_external_relationship_counter{"$src_ip"."->"."$dst_ip"}++;
        }
  } else {
        $external_src_ip_counter{"$src_ip"}++;
        $external_src_port_counter{"$src_port"}++;
        if ($dst_ip =~ "^MY\.NET") {
                $external_internal_relationship_counter{"$src_ip"."->"."$dst_ip"}++;
        } else {
                $external_external_relationship_counter{"$src_ip"."->"."$dst_ip"}++;
                # Hopefully, this case never happens.
        }
  }

if ($dst_ip =~ "^MY\.NET") {
        $internal_dst_ip_counter{"$dst_ip"}++;
        $internal_dst_port_counter{"$dst_port"}++;
} else {
        $external_dst_ip_counter{"$dst_ip"}++;
        $external_dst_port_counter{"$dst_port"}++;
}

# Assure the user that something's happening, and we're not hung.

    $happydots++;
    print "." if $happydots % 100 == 0; # if $happydots == 100;
    print "Just a moment." if $happydots % 46600 == 0;
}

foreach $key ( keys(%date_counter) ) {
        push (@dates, "$date_counter{$key},$key");
}
foreach $key ( keys(%alert_counter) ) {
        push (@alerts, "$alert_counter{$key},$key");
}
foreach $key ( keys(%internal_src_ip_counter) ) {
        push (@internal_src_ips, "$internal_src_ip_counter{$key},$key");
}
foreach $key ( keys(%internal_src_port_counter) ) {
        push (@internal_src_ports, "$internal_src_port_counter{$key},$key");
}
foreach $key ( keys(%internal_dst_port_counter) ) {
        push (@internal_dst_ports, "$internal_dst_port_counter{$key},$key");
}
foreach $key ( keys(%internal_dst_ip_counter) ) {
        push (@internal_dst_ips, "$internal_dst_ip_counter{$key},$key");
}
foreach $key ( keys(%external_src_ip_counter) ) {
```

```
        push (@external_src_ips, "$external_src_ip_counter{$key},$key");
}
foreach $key ( keys(%external_src_port_counter) ) {
        push (@external_src_ports, "$external_src_port_counter{$key},$key");
}
foreach $key ( keys(%external_dst_ip_counter) ) {
        push (@external_dst_ips, "$external_dst_ip_counter{$key},$key");
}
foreach $key ( keys(%external_dst_port_counter) ) {
        push (@external_dst_ports, "$external_dst_port_counter{$key},$key");
}
foreach $key ( keys(%internal_internal_relationship_counter) ) {
        push (@internal_internal_relationships,
"$internal_internal_relationship_counter{$key},$key");
}
foreach $key ( keys(%internal_external_relationship_counter) ) {
        push (@internal_external_relationships,
"$internal_external_relationship_counter{$key},$key");
}
foreach $key ( keys(%external_internal_relationship_counter) ) {
        push (@external_internal_relationships,
"$external_internal_relationship_counter{$key},$key");
}
foreach $key ( keys(%external_external_relationship_counter) ) {
        push (@external_external_relationships,
"$external_external_relationship_counter{$key},$key");
}

# Group everything up in a sensible order:

@things_we_care_about = (
        [@dates],
        [@alerts],
        [@external_src_ips],
        [@external_src_ports],
        [@external_internal_relationships],
        [@external_external_relationships],
        [@internal_src_ips],
        [@internal_src_ports],
        [@internal_internal_relationships],
        [@internal_external_relationships],
        [@internal_dst_ips],
        [@internal_dst_ports],
        [@external_dst_ips],
        [@external_dst_ports],
        );

# Write it all down.

print "\nWriting the report to $outfile.";
undef $happydots;

foreach $report_item (@things_we_care_about) {

# print OUTFILE "\n\@$report_item\n"; # Uncomment this for light debugging

if ($report_item eq @things_we_care_about[0]) {
        $title = "EOIs by Date";
   } elsif ($report_item eq @things_we_care_about[1]) {
        $title = "EOIs by Alert Message";
   } elsif ($report_item eq @things_we_care_about[2]) {
        $title = "EOIs by Source IP (External Only)";
   } elsif ($report_item eq @things_we_care_about[3]) {
        $title = "EOIs by Source Port (External Only)";
   } elsif ($report_item eq @things_we_care_about[4]) {
        $title = "EOIs by Relationship (External->Internal Only)";
   } elsif ($report_item eq @things_we_care_about[5]) {
        $title = "EOIs by Relationship (External->External Only)";
   } elsif ($report_item eq @things_we_care_about[6]) {
        $title = "EOIs by Source IP (Internal Only)";
   } elsif ($report_item eq @things_we_care_about[7]) {
```

```
        $title = "EOIs by Source Port (Internal Only)";
    } elsif ($report_item eq @things_we_care_about[8]) {
        $title = "EOIs by Relationship (Internal->Internal Only)";
    } elsif ($report_item eq @things_we_care_about[9]) {
        $title = "EOIs by Relationship (Internal->External Only)";
    } elsif ($report_item eq @things_we_care_about[10]) {
        $title = "EOIs by Destination IP (Internal Only)";
    } elsif ($report_item eq @things_we_care_about[11]) {
        $title = "EOIs by Destination Port (Internal Only)";
    } elsif ($report_item eq @things_we_care_about[12]) {
        $title = "EOIs by Destination IP (External Only)";
    } elsif ($report_item eq @things_we_care_about[13]) {
        $title = "EOIs by Destination Port (External Only)";

}

print OUTFILE "    ";
for ($i = -1; $i <= length($title); $i++) {print OUTFILE "_" ; }

print OUTFILE "\n";
print OUTFILE " __/ $title \\";
for ($i = 0; $i+8+length($title) <= 70; $i++) { print OUTFILE "_" ; }
print OUTFILE "\n";
printf OUTFILE "| %-68s|\n";

undef $eoi_unique_count;
undef $eoi_total_count;
unless (@$report_item) {
       printf OUTFILE "| %-68s|\n","No events of interest for this category (usually a
Good Thing)" ;
}

foreach $item ( reverse(sort{ $a <=> $b }(@$report_item))) {
        ($count,$entry) = split(/\,/,$item);

        # Assure the user we're doing stuff (ie, not hung or anything)...
            $happydots++;
            print "." and $happydots = 0 if $happydots == 100;

        $eoi_unique_count++;
        $eoi_total_count = $eoi_total_count + $count;

        if (length($entry) <= 58) {
                printf OUTFILE "| %-8d %-58s |\n",$count,$entry;
        } elsif (length($entry) > 65 ) {
                printf OUTFILE "| %-8d %-55s... |\n",$count,substr($entry,0,55);
        }
}

printf OUTFILE "| %-68s|\n";
printf OUTFILE "| %-20s%8d%31s%8d |\n ",
        "Total Uniques: ",
        $eoi_unique_count,
        "Total EOIs: ",
        $eoi_total_count;

for ($i = 0; $i <= 68; $i++) { print OUTFILE "-" ; }
print OUTFILE "\n";

}

print "\nDone!\n";
```

parse-alerts-simple.pl – A perl script used to format alert logs into .csv files to be used with the alert-report.pl script. (Courtesy of James Maher)

```
#!/usr/bin/perl -w
##########################################
#Name:  parse-alerts-simple.pl
#
```

```
#Synopsis: parse-alerts-simple.pl[-d][-p]-f<alert-file>
#
#Description:
#       Simple script to generate a csv file from
#       a snort alert file which I may later import
#       into a DB.  Also having one line of data
#       makes mining it in correlation with
#       the other files easier.
#       Very similar to parse-alerts.pl, however this script only does the csv stuff and
#       Analysis is left to alert-report.pl.
#
#Author: James Maher <scouser@paradise.net.nz>
#
###############################################

use Getopt::Long;
#get the options
my $result = GetOptions("file=s"  =>\$alert_file,    #string
                        "ports"    =>\$ports,         #include portscan data in csv
                        "help"     =>\$help,       #print out command line options and die
                       "debug"    =>\$debug);         #debug flag

die "\t--ports         include portscan data in csv\n",
  "\t--debug           debug flag\n",
  "\t--file <name>     Name of the alert file we are reading in\n",
  "\t--help            print out command line options and die.\n",
  if $help;

die "usage: $0 [-d][-p] -f <alert-file>\n" if (! defined($alert_file));

#use a temp output csv file if we are in debug mode
$csv_file=$debug ?'output.csv':defined($ports)?"$alert_file+scans.csv":"$alert_file.csv";

open DATA,"$alert_file" or die "Failed to open $alert_file: $?\n";
open CSV,">$csv_file" or die "Failed to open CSV file: $?\n";

#Data structure for storing results
my %alert_list;
while(<DATA>)
{
  if((!/^\d/)|(/.*\[\*{2}\].*\[\*{2}\].*\[\*{2}\]/)){
    push (@matchless, $_) if(!/^\d/);
    next;
  }
  chomp $_;
  $count++;
  #better make sure there are no commas in the data
  $_=~s/,//g;

  my ($date_s,$desc,$src,$src_prt,$dst,$dst_prt);

  #some patterns to make the regexps easier to follow
  my $IP_CHRS='(?:[a-zA-Z0-9\.]+)';
  my $DLM='\[\*{2}\]';
  my $EOS='End of portscan from';
  my $TM='Total time\S';

  if($_=~/^(\S+)\s+\[\*{2}\]\s+(.*)\s+\[\*{2}\]\s+($IP_CHRS)(:(\d+))?(?:\s+-
>\s+($IP_CHRS)(:(\d+))?)?$/){
               ($date_s,$desc,$src)=($1,$2,$3);
               if(defined($8)){
                 ($src_prt,$dst,$dst_prt)=($5,$6,$8);
                 print CSV "$date_s,$desc,$src,$src_prt,$dst,$dst_prt\n";
               }
               elsif( defined($6))
               {
                 ($src_prt,$dst,$dst_prt)=('-',$6,'-');
                 print CSV "$date_s,$desc,$src,$src_prt,$dst,$dst_prt\n";
               }elsif( defined($5))
               {
                 ($src_prt,$dst,$dst_prt)=($5,'-','-');
```

```
                print CSV "$date_s,$desc,$src,$src_prt,$dst,$dst_prt\n";
              }
  }

elsif($_=~/^(\S+)\s+SDLM\s+(\S+):$EOS\s+($IP_CHRS\.\d+):\s+$TM(\d+s)\S\s+hosts\S(\d+)\STC
P\S(\d+)\SUDP\S(\d+)\S\s+$DLM/){
              ($date_s,$desc,$src,$host_cnt)=($1,$2,$3,$5);
              $port_cnt=$6+$7;
              print CSV "$date_s,$desc,$src,$host_cnt,$port_cnt\n" if$ports;
  }
  elsif($_!~/^(\S+)\s+$DLM\s+spp_port/){
              push(@matchless,$_);
              print STDERR "\nNo Match:\t$_\n";
              next;
  }
}
close DATA or die "Failed to close data file!\n";
print "\n\t processed $count records\n\n";
```

alert-reports.pl – A perl script used to generated several alert reports based on the different options available. (Courtesy of James Maher)

```
################################################################################
#!/usr/bin/perl -w

############################################################
#Name: alert-reports.pl
#
#Synopsis: alert-reports.pl [-d][-s][-n][-c<max>][-l<min>]-f<alert-file>
#
#Description:
#   Read in the parsed alert.csv file
#   and generate a number of reports
#
#Author: James Maher <scouser@paradise.net.nz>
#$Id: alert-reporter.pl,v 1.4 2003/06/22 11:49:09 Exp scouser $
############################################################

use Getopt::Long;

#get the options
my $result = GetOptions("file=s" =>\$alert_file,   #Name of the alert CSV file we are
reading in
                 "portscans"   =>\$scans,        #should we include portscans?
                "ip-order"     =>\$ip_order,     #print IP summary ordered by IP not
frequency
                "summary"      =>\$summary,      #Summary mode only prints out totals of
hits per IP,
                                                 #not a line for each attack type per
host.
                "verbose"      =>\$verbose,      #always print out full lists rather
than totals
                "lowest"       =>\$floor,        #what is the cut off for printing out
data
                "ceiling=i"    =>\$ceiling,      #How many entries should I print out
per report(ie top ten)
                "no_attack"    =>\$no_attack,    #Do not generate an attack report
                "target"       =>\$dst_grp,      #Group by target address rather than
source address
                "help"         =>\$help,         #print out command line options and
die.
                "debug"        =>\$debug);       #debug flag

die "\t--ceiling <max> How many entries should I print out per report(ie top ten)\n",
  "\t--debug           debug flag\n",
  "\t--file <name>     Name of the alert CSV file we are reading in\n",
  "\t--help            print out command line options and die.\n",
  "\t--ip-order        print IP summary ordered by IP not frequency\n",
  "\t--lowest <min>    what is the cut off for printing out data\n",
```

```
  "\t--no_attack       Do not generate an attack report\n",
  "\t--summary         Summary mode only prints out totals of hits per IP,\n",
  "\t                  not a line for each attack type per host.\n",
  "\t--target          Group by target address rather than source address\n",
  "\t--verbose         always print out full lists rather than totals\n",
  "\t--help            print out command line options and die.\n",
  if $help;

die "usage: $0 [-d][-s][-n][-c <max>][-l <min>]-f <alert-file>\n" if(!
defined($alert_file));
die "Alert file not a .csv file, aborting...\n" if $alert_file !~/csv$/;

#Set things up before we start.
#Data structure for storing results
my %alert_list;
my $rpt_dir='/home/cashmoney/GIAC/reports';
&initialise();

&read_in_data();

&gen_ip_report();

&gen_attack_report() if!$no_attack;

print "\nFinished analysis, reports written to:\n$address_report\n";
print "$atck_rpt\n" if!$no_attack;
print "\n";

#Subroutines
sub gen_ip_report(){
#Lets see what we have got;-)
#and print out to a file
  print "\nGenerating IP report... -";
  open TARGET,">$address_report" or die "Could not open IP address report file\n";
  print TARGET $rpt_heading;
  $cnt =0;
  foreach $target(sort ip_sort keys(%{$alert_list{'hosts'}}))
  {
              $cnt++
              &progress(200);
              local *STDOUT=*TARGET;
              next if $target =~/^-/;
              last if(($floor)&&($alert_list{'hosts'}->{$target}->{'count'}<$floor));
              last if(($ceiling)&&($cnt>=$ceiling));
              if($summary){
                print "$target\t ".$alert_list{'hosts'}->{$target}->{'count'}."\t\t";

                my @alerts=keys(%{$alert_list{'hosts'}->{$target}});
                my $num_alrts=$#alerts;
                print " $num_alrts\t\t";
                my @hosts=(keys(%{$alert_list{'hosts'}->{$target}->{'hosts'}}));
                my $count=($#hosts+1);
                print " $count\t";
                my @ports=(keys(%{$alert_list{'hosts'}->{$target}->{'dprts'}}));
                $count=($#ports + 1);
                print " $count\n";

              }
              else
              {
                foreach $alert_desc(sort keys(%{$alert_list{'hosts'}->{$target}}))
                {
                            #save count till after -)
                            next if $alert_desc =~/^(count)|(hosts)|(sprts)|(dprts)/;

                            my $alrt= $alert_list{'hosts'}->{$target}->{$alert_desc};
                            print "$target,$alert_desc,";
                            print $alrt->{"count"}.",";

                            my @hosts=(keys(%{$alrt->{'hosts'}}));
                            my $num_hosts=$#hosts+1;
```

```
                                if(($num_hosts < 3)||($verbose)){
                                  print "H: ";
                                  my $last=pop @hosts;
                                  print "$_," foreach (@hosts);
                                  print "$last";
                                }else{
                                  print $num_hosts;
                                }
                                print "\t";

                                my @ports=(keys(%{$alrt->{"ports"}}));
                                my $num_ports=$#ports+1;
                                if($num_ports==0){
                                  print "\n";
                                }elsif(($num_ports<2)||($verbose)){
                                  print "P: ";
                                  my $last=pop @ports;
                                  print "$_," foreach @ports;
                                  print "$last\n";
                                }
                                else{
                                  print "$num_ports\n";
                                }

                  }
                  print "\t".$alert_list{'hosts'}->{$target}->{'count'}."total hits
($target)\n";
                }
  }
  close TARGET ||die "Could not close target tally output file\n";
}

sub gen_attack_report(){
#now for the report by attack type
#pretty similar to above
  print "\nGenerating attack report... -";
  open ATTACK,">$atck_rpt" or die "Could not open tally output file\n";
  print ATTACK $atck_heading;
  $cnt=0;
  foreach $attack(sort atck_sort keys(%{$alert_list{'Attacks'}}))
  {
              $cnt++;
              last if(($ceiling)&&($cnt>=$ceiling));
              &progress(5);
              local *STDOUT=*ATTACK;
              last if(($floor)&&($alert_list{'Attacks'}->{$attack}->{'count'}<$floor));
              my $alrt= $alert_list{'Attacks'}->{$attack};
              print "$attack\t".$alrt->{'count'}."\t";

              foreach $hst('src','dst'){
                my @hosts=(keys(%{$alrt->{$hst}}));
                my $num_hosts=$#hosts+1;

                if(($num_hosts<3)||($verbose)){
                                print "H: ";
                                my $last=pop @hosts;
                                print "$_," foreach(@hosts);
                                print "$lasts";
                }else{
                                print "$num_hosts hosts";
                }
                print "\t";
              }

              my @ports=(keys(%{$alrt->{"ports"}}));
              my $num_ports=$#ports+1;
              if($num_ports==0){
                print "\n";
              }elsif(($num_ports<3)||($verbose)){
                print "P:";
                my $last=pop @ports;
```

```
                print "$_," foreach @ports;
                print "$last\n";
              }else{
                print "$num_ports ports\n";
              }
  }
  close ATTACK ||die "\nCould not close target tally output file\n";
}

#sort attacks by frequency
sub atck_sort(){
  return($alert_list{'Attacks'}->{$b}->{'count'}<=>$alert_list{'Attacks'}->{$a}-
>{'count'});
}

#sort IP addresses
sub ip_sort{
  if(!$ip_order){
              return-1 if(!defined($alert_list{'hosts'}->{$b}-
>{'count'}))||(!defined($alert_list{'hosts'}->{$a}->{'count'}));
              return($alert_list{'hosts'}->{$b}->{'count'}<=>$alert_list{'hosts'}->{$a}-
>{'count'});
  }

  #else we have to do sort by IP address ;-(
  my($a1,$a2,$a3,$a4)=split(/\./,$a);
  my($b1,$b2,$b3,$b4)=split(/\./,$b);

  if(($a1 =~/^MY/)||($b1 =~/^MY/)){
              if(($a1 =~/^MY/)&&($b1 =~/^MY/)){
                if($a3 ne $b3){ return($a3<=>$b3);}
                if($a4 ne $b4){ return($a4<=>$b4);}
              }
              return-1 if($a1 =~/^MY/);
              return 1;
  }

  if($a1 ne $b1){ return($a1<=>$b1);}
  if($a2 ne $b2){ return($a2<=>$b2);}
  if($a3 ne $b3){ return($a3<=>$b3);}
  if($a4 ne $b4){ return($a4<=>$b4);}

  return($a<=>$b);
}

#Initialise variables based on command line args etc.
sub initialise(){
  die "WARNING: Incompatible options \'--scans\' and \'--target\'-tCan't group portscan
alerts by destination port, exiting.\n" if(($dst_grp)&&($scans));

  print "--portscans flag not implemented, ignoring...\n" if $scans;
  undef $scans;

  $line_cnt=0;
  print "\ninitialising report variables...\n";
  $ad_sfx=defined($summary)?'summary':defined($verbose)?'verbose':'normal';
  $ad_sfx.="-$floor" if defined($floor);
  $ad_sfx.="-$ceiling" if $ceiling;

$address_report=defined($dst_grp)?"$rpt_dir/dst_IP_report.$ad_sfx":"$rpt_dir/src_IP_repor
t.$ad_sfx";

  $at_sfx=defined($verbose)?'verbose':'normal';
  $at_sfx.="-$floor" if defined($floor);
  $at_sfx.="-$ceiling" if $ceiling;
  $atck_rpt="$rpt_dir/attack_report.$at_sfx";;

  $rpt_heading="Hits per Destination Address\n=======================\n" if $dst_grp;
  $rpt_heading="Hits per Source Address\n=======================\n" if !$dst_grp;
  $rpt_heading.="(including portscans)\n" if $scans;
  $rpt_heading.="Verbose Mode\n" if $verbose;
```

```
  $rpt_heading.="\n Address \tTot-Hits \tDistinct \tHosts \t Ports\n" if $summary;

  $atck_heading="Attack/Scan Types\n=================\n";
  $atck_heading.="(including portscans)\n\n" if $scans;
  $atck_heading.="Verbose Mode\n" if $verbose;
  $atck_heading.="\nDescription\t\tHits\tSrc IPs\t\tDst IPs\t\tDst Ports\n";

  print "Ceiling set to $ceiling\n" if $ceiling;
  #just for fun, and to stop you from going crazy while the reports run ;-)
  @display =('-','\\','|','/');
}

#
sub read_in_data(){
  print STDERR "reading in csv file...-";
  open DATA, "$alert_file" or die "Failed to open $alert_file: $?\n";
  while(<DATA>){
        &progress(1000);
        chomp;

         #no need to analyse if it is an spp portscan line unless forced to by command
line flag
         my @data=split/,/;
         if($data[1]!~/^spp_portscan/){
                 &process_alert(@data);
                }
                elsif($scans){
                   #&process_scan(@data); #Not implemented - Handling portscans separately
now.
                }
                else{print "DBG:$_\n" if $debug;}
        }
  close DATA or die "Failed to close data file!\n";
}

sub process_alert(){
  my ($date_s,$desc,$src,$src_prt,$dst,$dst_prt)=@_;
  #decide who to log this under Will use the source address for now.
  if(!defined($dst)&&($dst_grp)){
                print STDERR "Target grouping specified but no dst IP!! Ignoring this
alert.\n" if $debug;
                next;
  }
  my $alert_host=$dst_grp ?$dst:$src;
  my $scnd_host=$dst_grp ?$src:defined($dst) ?$dst:'-';
  $alert_list{'hosts'}->{$alert_host}->{$desc}={} if !$alert_lilst{'hosts'}-
>{$alert_host}->{$desc};
  my $alert=$alert_list{'hosts'}->{$alert_host}->{$desc};

  $alert->{'ports'}->{$dst_prt}++ if( defined($dst_prt)&&($dst_prt !~/-/));
  $alert->{'hosts'}->{$scnd_host}++ if $scnd_host !~/^-/;
  #print $alert->{'hosts'}->{$scnd_host}."\t" if $scnd_host !~/^-/;
  $alert->{'count'}++;

  #lets also count total alerts against a host
  $alert_list{'hosts'}->{$alert_host}->{'count'}++;
  #and the total number of ports
  $alert_list{'hosts'}->{$alert_host}->{'sprts'}->{$src_prt}++ if(
defined($src_prt)&&($src_prt !~/-/));
  $alert_list{'hosts'}->{$alert_host}->{'dprts'}->{$dst_prt}++ if(
defined($dst_prt)&&($dst_prt !~/-/));
  $alert_list{'hosts'}->{$alert_host}->{'hosts'}->{$scnd_host}++ if(
defined($scnd_host)&&($scnd_host !~/-/));

  #lets also collate totals for different attacks
  $alert_list{'Attacks'}->{$desc}={} if !$alert_list{'Attacks'}->{$desc};
  local *signature=$alert_list{'Attacks'}->{$desc};
  $signature{'count'}++;
  $signature{'src'}->{$alert_host}++;
  $signature{'dst'}->{$scnd_host}++ if $scnd_host !~/^-/;
  $signature{'ports'}->{$dst_prt}++ if( defined($dst_prt)&&($dst_prt !~/-/));
```

```
}

sub progress(){
  $mod=shift;
  if($line_cnt%$mod==0){
               $char=shift @display;
               system("echo -n");
               print "\b\b$char";
               push @display,$char;
  }
  $line_cnt++;
}
```

Appendix B – Complete List of Alerts

Alert	Hits	Src IPs	Dst IPs	Ports
EXPLOIT x86 NOOP	38903	2084	916	65
MY.NET.30.4 activity	35289	313		23
High port 65535 tcp – possible Red Worm – traffic	19471	119	154	86
MY.NET.30.3 activity	15900	197		23
SMB Name Wildcard	8627	61	639	137
Tiny Fragments – Possible Hostile Activity	4412	5	18	
RFB – Possible WinVNC – 010708-1	2355	15	17	1099
Null Scan!	1936	89	54	152
NMAP TCP ping!	869	218	67	46
Possible trojan server activity	419	48	52	9
SUNRPC highport access!	254	25	25	1 (32771)
connect to 515 from outside	250			515
TCP SMTP Source Port traffic	191	4	4	25
DDOS shaft client to handler	147	3	1	1 (20432)
[UMBC NIDS IRC Alert] IRC user /kill detected possible trojan.	138	61	49	111
Incomplete Packet Fragments Discarded	127	40	28	1 (0)
High port 65535 udp – possible Red Worm – traffic	122	34	28	29
TCP SRC and DST outside network	80	23	34	15
SMB C access	75	19	5	1 (139)
FTP passwd attempt	69	44	1	1 (21)
[UMBC NIDS IRC ALERT] Possible sdbot floodnet detected attempting to IRC	65	12	7	1 (7000)
ICMP SRC and DST outside network	43	21	37	
TFTP – Internal UDP connection to external tftp server	41	12	11	6
IDS552/web-iis_IIS ISAPI Overflow ida Internal nosize	38	1	21	1 (80)
[UMBC NIDS IRC Alert] Possible drone command detected.	34	6	5	19
NIMDA – Attempt to execute cmd from campus host	34	11	23	1 (80)
EXPLOIT x86 setuid 0	28	22	16	19
EXPLOIT x86 setgid 0	26	22	20	24
Attempted Sun RPC high port access	25	6	21	1 (32771)
External RPC call 23	23	1	4	1 (111)

[UMBC NIDS IRC Alert] SDCC client detected attempting to IRC	23	1	4	2 (7000, 6667)
[UMBC NIDS] External MiMail alert	17	12		25
TFTP – External TCP connection to internal tftp server	17	4	5	2 (69, 48386)
FTP DoS ftpd globbing	15			21
EXPLOIT NTPDX buffer overflow	14	7	6	1 (123)
DDOS mstream client to handler	14	4	3	2 (15104, 12754)
[UMBC NIDS] Internal MiMail alert	11	4	10	1 (25)
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	10	1		1 (34078, 32795)
IRC evil – running XDCC	7			6663
TFTP – Intenal TCP connection to external tftp server	6	3	3	3
EXPLOIT x86 stealth noop	5	4	4	4
DDOS mstream handler to client	4		1	2 (1164, 4662)
connect to 515 from inside	4			1 (515)
Traffic from port 53 to port 123	3			1 (123)
SYN-FIN scan!	3	1	1	3
Probable NMAP fingerprint attempt	2	1	1	2 (0, 6131)
External FTP to HelpDesk MY.NET.70.49	2	1		1 (21)
EXPLOIT x86 NOPS	2			1 (7001)
NETBIOS NT NULL session	2	1		1 (139)
NIMDA – Attempt to execute root fro campus host	2		1	1 (80)
HelpDesk MY.NET.70.49 to External FTP	2			1 (21)
TFTP – External UDP connection to internal tftp server	1			1 (69)
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	1			1 (1642)
Back Orifice	1			1 (31337)
External FTP to HelpDesk MY.NET.53.29	1			1 (21)
External FTP to HelpDesk MY.NET.70.50	1			1 (21)