Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment V4.1

Brian Bailey
October 17, 2004

1. Executive Summary

This analysis was conducted on three days worth of logs spanning from March 25th 2004 to March 27th 2004. These logs were generated by the university's Snort intrusion detection system. The intrusion detection system produced three types of log files, Alert, Scan and OOS (Out of Spec) files.

The three days worth of logs contained well over 70,000 alerts (not including alerts generated by Snort's port scan preprocessor). Due to the sheer volume of logs generated, only the most critical alerts were analyzed.

There are some serious issues concerning the university's approach to security. It would seem that there is simply no perimeter filtering or firewalls in place at this university. Scan and alert files show packets traveling to and from the university's network on a wide range of destination ports. By allowing these packets to pass into the university's network unhindered, attackers, worms, trojans, and other malicious traffic can pass to and from university resources at will. Most of the alerts and scans detected could have been easily dealt with by filtering the traffic at the perimeter.

The intrusion detection system generated so many alerts that it may be difficult, if not impossible, for an analyst to adequately evaluate log files on a regular basis. For example, there are certain rules that alert on any traffic to a specified host. Any packet sent to this host, including valid traffic, may trigger these types of rules. This creates a situation where an analyst must sort through all these events to find alerts that may be malicious. Over the three days worth of log files contained in this analysis, one such rule triggered 22,862 alerts. Reviewing every alert generated by this rule in search of malicious traffic is an exercise in futility. Instead these rules should be eliminated, and new rules should be implemented in order to reduce the amount of false positives.

Due to the lack of security measures, some hosts within the university's network have already been compromised. Other hosts remain at risk while the network is relatively undefended. It is the opinion of this analyst that all security policies should be reevaluated, and that the network infrastructure should be redesigned to employ more effective security measures.

2.1 Log Files Analyzed

The following log files were used in this analysis and were obtained from http://www.incidents.org/logs/

Alert Files	OOS Files	Scan Files
alert.040325	oos_report_040325	scans.040325
alert.040326	oos_report_040326	scans.040326
alert.040327	oos_report_040327	scans.040327

2.2 Overview of Alerts

The following table shows the top ten unique alerts detected within the alert logs, and is sorted by the amount of occurrences of each alert. Following this table, three of the more critical alerts are examined in depth.

Alert Name	Number of Occurrences	Unique Sources	Unique Destinations
MY.NET.30.3 activity	22862	132	1
MY.NET.30.4 activity	15942	194	1
connect to 515 from outside	13730	3	246
High port 65535 tcp - possible Red Worm - traffic	11367	77	100
SHELLCODE x86 NOOP	4461	454	460
SMB Name Wildcard	3662	85	301
Null scan!	1364	117	526
NMAP TCP ping!	510	129	55
FTP DoS ftpd globbing	347	15	2
Possible trojan server activity	286	31	36

2.2.1 Botnet Activity

Description of detect

A Botnet is a collection of hosts that have been compromised and are using attacker-installed Bots to connect to an IRC (Internet Relay Chat) server of the attacker's choosing, wherein the attacker can control these hosts through commands sent via IRC. The specific attack vector in this case is unknown, as the attacker can use a great many techniques in order to install and run the Bot on the remote machine.

The damage a Bot can do depends entirely on how the attacker designed it. Worse case scenario the attacker will have complete control over the infected machine. While Bots are a threat on their own, the dangers multiply exponentially when they come together to form a Botnet. One of the more common uses for Botnets is to facilitate DDoS (Distributed Denial of Service) attacks. Here an attacker can enter an IRC chat room where he can send a command to all of his Bots to perform a DoS attack against a target machine or network. Once that command is received, all the Bot hosts will begin to attack the target with SYN Floods, or excessive UDP or ICMP packets.

Reason this detect was selected

This series of detects were chosen due to the inherent danger in allowing an attacker to have control over a machine on the internal network. Allowing an internal host to be controlled by an attacker is dangerous to say the least, but those dangers increase dramatically in the event that Botnet exists on the internal network.

Detect was generated by

These detects were generated by the Snort intrusion detection system. The IRC related alerts being analyzed are illustrated in Figure 2. There are other IRC alerts, but these three stand out, and contain the particular Botnet being examined.

Alert Name	Number of alerts	Number of Sources	Number of Destinations
[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.	284	30	45
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	173	13	1
[UMBC NIDS IRC Alert] Possible drone command detected.	15	1	2

Figure2

These are not standard Snort alerts, but were created by university staff members. Therefore this analyst can only speculate on what those rules actually contain. One alert detects the IRC command */kill*. This command is used to kick users off of IRC servers. The volume of these alerts may be explained by the fact that those users getting kicked out of IRC are not normal users, but Bots. These Bots may have auto-connect features

built in that will reconnect them to the server once kicked out from the /*kill* command. The other two IRC related alerts mentioned here are more of a mystery, but suffice to say that they were triggered because of suspicious IRC activity. Examples of these alerts are illustrated in Figure 3, and show the Bots communicating with the IRC server containing the Botnet.

```
03/25-20:59:45.214150 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible
trojan. [**] 139.165.206.128:6666 -> MY.NET.97.209:3041

03/27-13:01:43.449994 [**] [UMBC NIDS IRC Alert] Possible sdbot floodnet detected
attempting to IRC [**] MY.NET.97.78:3668 -> 139.165.206.128:6666

03/26-21:50:18.462765 [**] [UMBC NIDS IRC Alert] Possible drone command
detected. [**] 139.165.206.128:6666 -> MY.NET.97.102:2514
```

Figure3

Probability the source address was spoofed

In order for the attacker to use Bots or a Botnet, they must communicate with the target hosts via TCP. Due to the requirement of the TCP three-way handshake prior to two-way communication, it makes it impossible for the attacker to use a spoofed IP address in this case.

It is possible, however, even likely, that the infected hosts will use spoofed IP addresses to perform DoS attacks triggered in response to the attacker's command to the Botnet.

Attack Mechanism

In order to truly understand the attack mechanism, the IRC Bot should be identified. In this case the most likely culprit is the Agobot/Gaobot worm. Some easily noticeable trends appeared as the infected machines' traffic patterns were analyzed. Almost every infected host on the Botnet attempted to propagate by means associated with Agobot. Thanks to the scan files, each host was observed attempting to connect to random addresses on four known ports. The ports and services targeted are illustrated in Figure 4. If one of those services were offered on the target machine, the worm would have most likely spread to that host. A snippet of the scan logs shows several of the infected machines attempting to make such connections as seen in Figure 5.

TCP Port	Service
80	IIS – WebDav Vulnerability
1025	Possibly LSASS or RPC DCOM vulnerability in MS Windows
2745	Backdoor created by the Bagle Worm
3127	Backdoor created by the MyDoom Worm
6129	Dameware – Buffer overflow vulnerability

Figure4

```
Mar 25 21:16:42 MY.NET.97.209:3495 -> 18.76.57.202:2745 SYN ******S*
Mar 25 21:16:43 MY.NET.97.209:3512 -> 18.76.57.202:3127 SYN ******S*
Mar 25 21:16:43 MY.NET.97.209:3515 -> 18.76.57.202:80 SYN ******S*
Mar 25 21:16:43 MY.NET.97.209:3513 -> 18.76.57.202:6129 SYN ******S*
Mar 25 21:16:42 MY.NET.97.209:4243 -> 128.30.182.247:2745 SYN ******S*
Mar 25 21:16:43 MY.NET.97.209:4247 -> 128.30.182.247:1025 SYN ******S*

Mar 26 09:03:46 MY.NET.97.50:1799 -> 130.146.102.23:6129 SYN ******S*
Mar 26 09:03:46 MY.NET.97.50:1796 -> 130.146.102.23:1025 SYN ******S*
Mar 26 09:03:46 MY.NET.97.50:1794 -> 130.146.102.23:2745 SYN ******S*
Mar 26 09:03:46 MY.NET.97.50:1802 -> 130.146.102.23:80 SYN ******S*
Mar 26 09:03:46 MY.NET.97.50:1798 -> 130.146.102.23:3127 SYN ******S*

Mar 27 22:26:00 MY.NET.97.92:4099 -> 130.33.25.114:80 SYN ******S*
Mar 27 22:26:00 MY.NET.97.92:4117 -> 130.8.127.128:1025 SYN ******S*
Mar 27 22:26:00 MY.NET.97.92:4566 -> 130.15.49.4:2745 SYN ******S*
Mar 27 22:26:01 MY.NET.97.92:3034 -> 130.30.133.32:1025 SYN ******S*
Mar 27 22:26:01 MY.NET.97.92:3161 -> 130.30.133.32:6129 SYN ******S*
Mar 27 22:26:02 MY.NET.97.92:3112 -> 130.30.133.32:3127 SYN ******S*
```

Figure5

Unfortunately, it is unclear as to how the worm spread to the infected university hosts. There simply is not enough information to draw any accurate conclusions. It is also worth mentioning that the source of this traffic is *most likely* the Agobot worm. Concrete answers cannot be given without further analysis of the infected machines, or examining packet captures from the infected machines.

It is also unclear as to what the purpose of this Botnet was. As seen in the alerts, the majority of traffic from the IRC server was the /kill command. This may have been a channel operator attempting to get rid of the Bots. It may also have been the owner of the Bots, but this is just speculation. Along with the /kill commands, then IRC server triggered the “Possible drone command detected” alert. Without the ability to view the packet traces, it is impossible to guess what that command may have been.

No significant traffic was seen from these infected hosts other than the propagation traffic shown in Figure 5. Thankfully, It does not appear that these Bots were used in any form of DoS attacks. A complete list of infected machines within this Botnet is illustrated in Figure 6.

MY.NET.111.51	MY.NET.97.164	MY.NET.97.78
MY.NET.98.49	MY.NET.97.124	MY.NET.97.50
MY.NET.97.235	MY.NET.97.113	MY.NET.97.25
MY.NET.97.209	MY.NET.97.102	
MY.NET.97.185	MY.NET.97.92	

Figure6

Correlations

Andrew Evans discusses some similar alerts regarding IRC Bots in his GCIA paper.
http://www.giac.org/practical/GCIA/Andrew_Evans_GCIA.pdf

Danny Schales talks about the Agobot worm in this email from the DShield mailing list.
http://lists.sans.org/pipermail/list/2004-April/047574.html

Evidence of active targeting

The infected hosts were not victims of active targeting. If the source of the infection is in fact the Agobot worm, then chances are the addresses of the targeted machines were randomly generated from an already infected host. Certain services and ports were actively targeted in propagation of the worm, but the hosts themselves were not.

Severity

Severity = (Criticality + Lethality) – (System countermeasures + Network countermeasures)

Criticality = 4
There is no evidence that the infected machines are critical to the operations of the university's network. These machines are, however, under complete control from an outside source. This can lead to a great many issues, including theft of confidential information, further attacks against the university's network resources from an inside source, and even attacks against external networks that originate from the university network. Based on these risks, and the amount of hosts infected, the criticality level is raised.

Lethality = 5
If successful, this attack allows an attacker to take complete control over the targeted machine. This is the worse possible type of compromise.

System countermeasures = 1
Even the most basic system countermeasures do not appear to have been used on the infected machines. A host-based firewall would have prevented this worm from infecting the computers, and an up to date antivirus would have detected and removed the worm.

Network countermeasures = 2
Even the most rudimentary perimeter filtering would have stopped this attack. There is no reason the packets destined for these hosts with the destination ports used by Agobot should have gotten past the perimeter. There is an intrusion detection system in place in this network though which does count for something. Without it, university staff would have been completely unaware of any malicious traffic.

Based on the previously mentioned formula, the severity level for this incident is six.

2.2.2 EXPLOIT x86 NOPS

Description of attack

This rule is triggered in the event that a packet contains a substantial amount of x90 characters. This character is known as a NOP, which is an x86 machine code for no-operation. These are most often seen in buffer overflows. There have been known to be false positives in the event of large file transfers. This, however, is not the case for this particular alert. Since this is a relatively generic alert to catch excessive NOPs, an accurate description of the alert cannot be made unless the attack using the buffer overflow is identified.

The packets that triggered these alerts have two very interesting characteristics to them. The source port is always 4000, and the destination port seems to be a totally random high port. This evidence, coupled with the presence of excessive NOPS within these packets point to one particularly nasty worm known as "Witty". This worm was released on March 20th, and appeared in the logs analyzed on March 25th.

The Witty worm targets a specific flaw in ISS products such as BlackICE and RealSecure. From http://xforce.iss.net/xforce/alerts/id/166, "*The flaw relates to incorrect parsing of the ICQ protocol which may lead to a buffer overflow condition*." The Witty worm attempts to take advantage of this flaw to unleash its extremely destructive payload. After infection, the worm, like all others, will attempt to propagate itself through mechanisms detailed later in this analysis. The real harm comes from its next step, and that is to overwrite a random 128 sectors of the hard drive on the target host. At this point the cycle is repeated. It will attempt to propagate, and will overwrite 128 more sectors of the hard drive with data from memory.

This not only causes system instability (and eventual failure), but can potentially destroy business critical data as well. This payload makes this worm a significant threat to the university's infrastructure.

Reason this detect was selected

This detect was selected due to the potential threat it posed to the university network. If the university uses affected ISS products on their network then the risks associated with this worm are staggering. If the use of these ISS products is a standard, then it is quite possible the worm could cripple the university's entire network.

Detect was generated by

This detect was generated by the Snort intrusion detection system. The rule that triggered these alerts seemed to be a variant of the arachNIDS rule "IDS362 SHELLCODE-X86-NOPS-UDP". This rule is designed to alert on excessive use of the x86 machine code NOP. It is a relatively generic rule that is used to detect buffer overflow attempts. In this case the rule was successful in identifying this threat. Several examples of the alert are shown in Figure 7. Figure 7 also displays, in red, the source port of 4000 used in these scans. Details on the arachNIDS rule can be found here: http://www.whitehats.com/info/IDS362.

03/25-16:14:34.463394 [**] EXPLOIT x86 NOPS [**] 61.175.223.60:4000 -> MY.NET.130.145:46025
03/25-16:25:33.570850 [**] EXPLOIT x86 NOOP [**] 61.175.223.60:4000 -> MY.NET.84.157:8290
03/25-16:26:38.215835 [**] EXPLOIT x86 NOPS [**] 61.175.223.60:4000 -> MY.NET.147.158:30844
03/25-16:27:54.575238 [**] EXPLOIT x86 NOPS [**] 61.175.223.60:4000 -> MY.NET.20.230:23088
03/25-16:28:08.569920 [**] EXPLOIT x86 NOPS [**] 61.175.223.60:4000 -> MY.NET.24.95:58792

Figure7

The attacking host shown in Figure 7 is 61.175.223.60. Over the course of the three days analyzed, traffic from this host was only seen on March 25th between 16:14 and 19:49. During this time the attacking hosts targeted 173 hosts on the university's network. Unfortunately, the scans themselves were spread far enough apart so Snort's preprocessor spp_portscan did not log these events. Other hosts scanned the university network as well, but 61.175.223.60 was the most active host.

The arachNIDS rule alerts on External to Internal packets. If one of targeted hosts were infected, it would begin to propagate in a manner consistent with the Witty worm. The IDS would not alert on this traffic since the attack would be from Internal to External. Therefore no conclusions can be made as to whether or not the attack was successful without examining the target hosts.

Probability the host address was spoofed

This is a UDP based attack. This allows the worm to spoof its source IP address, while still being able to execute the attack. This analyst has found no evidence however, that the Witty worm uses this technique during propagation.

Attack Mechanism

As previously discussed, the Witty worm uses a flaw found in certain ISS products in order to exploit and infect target hosts. The flaw relates to incorrect parsing of the ICQ protocol. The key in this attack is the source port of 4000. The worm is pretending to be a valid ICQ packet. Once that packet is received and analyzed by the affected ISS products, the buffer overflow is executed, and the worm infects the host. Since the only criterion used is the source port of 4000, the worm can send malicious packets to any random destination port.

Once a target host is infected, the worm does two things. It sends itself out to 20,000 randomly generated IP addresses with a UDP source port of 4000, and a random destination port. Then it executes its payload that overwrites 128 sectors of the hard drive with data from memory. The cycle then repeats.

As stated previously, it is unclear as to whether or not this attack was successful. If the spp_portscan preprocessor were able to identify these scans it may have been possible to locate infected hosts. The worm spreads the scans far enough apart to effectively elude detection. The rule may also be addressing external scans only. If the rule were modified to detect excessive NOPs from internal hosts as well as external hosts, the IDS may be able to detect infected machines.

Correlations

ISS released two alerts dealing with this issue. The first relates to the flaw itself, while the other deals with the Witty Worm.
http://xforce.iss.net/xforce/alerts/id/166
http://xforce.iss.net/xforce/alerts/id/167

Symantec provides more information on the Witty Worm as well
http://securityresponse.symantec.com/avcenter/venc/data/w32.witty.worm.html

Analysis of the worm by LURHQ provides a correlation with this analysis.
http://www.lurhq.com/witty.html

Evidence of active targeting

The worm did not actively target the hosts attacked on the university's network. During this analysis it was shown that the worm *randomly* generates 20,000 IP addresses to use during propagation. The worm did actively target specific ISS products that were vulnerable to a buffer overflow. The names and versions of the actively targeted software are shown in Figure 8.

BlackICE™ Agent for Server 3.6 ebz, ecb, ecd, ece, ecf
BlackICE PC Protection 3.6 cbz, ccb, ccd, ccf
BlackICE Server Protection 3.6 cbz, ccb, ccd, ccf
RealSecure® Network 7.0, XPU 22.4 and 22.10
RealSecure Server Sensor 7.0 XPU 22.4 and 22.10
RealSecure Desktop 7.0 ebf, ebj, ebk, ebl
RealSecure Desktop 3.6 ebz, ecb, ecd, ece, ecf
RealSecure Guard 3.6 ebz, ecb, ecd, ece, ecf
RealSecure Sentry 3.6 ebz, ecb, ecd, ece, ecf

Figure8

Severity

Severity = (Criticality + Lethality) – (System countermeasures + Network countermeasures)

Criticality = 4
This threat can cause severe damage to the network if hosts run the affected versions of the ISS software. It is possible that the use of this software is a standard at the university, and thus the criticality level must be raised. This threat could potentially affect the entire network in such a manner that could cripple network resources.

Lethality = 5
This worm has the capacity to destroy the use of the hosts it infects. Loss of data and functionality are extremely lethal to any network resource. Therefore, the highest level of lethality must be assigned.

System countermeasures = 3
It is unknown what countermeasures may be employed on these hosts. If they are using the affected ISS products then the very countermeasures being used could put the hosts at higher risk. Antivirus may be installed on these hosts, but that is unknown as well. Due to the lack of information a conservative rating is given for system countermeasures.

Network countermeasures = 2
This threat could have been dealt with at the perimeter by filtering these packets. The fact that these packets even made it to the hosts reduces the value of network countermeasures. An intrusion detection system is in place, and did alert on these malicious packets. This slightly raises the score for network countermeasures.

Based on the previously mentioned formula, the severity level for this incident is 4.

2.2.3 Excessive amount of false positives

Description of detect

This detect is not a conventional attack like other detects. This detect deals with a large amount of false positives from an incorrectly configured IDS. The amount of false positives creates a situation where an analyst may not be able to perform regular analysis of the intrusion detection system's log files. Two alerts generated by the university's IDS stand out. The alerts, "MY.NET.30.3 activity" and "MY.NET.30.4 activity", produced a total of 38,804 alerts. These two alerts count for 50.5% of the alerts during the three-day's worth of log files analyzed (this does not include portscan alerts).

Of those 38,804 alerts, it is difficult to ascertain what alerts are constitute valid traffic, what are false positives, and what are alerts worth looking into. Having rules that generate so much noise is not an efficient way to manage an IDS or an analyst's time. These types of rules create situations where an analyst spends so much time sorting

through valid traffic or false positives that they may miss out on alerts that require attention. These rules essentially defeat the purpose of having an intrusion detection system in place to begin with.

Reason this detect was selected

This detect was selected because it severely hinders to process of intrusion detection at the university. The amount of false positives and alerts on valid traffic could cause analysts to miss an opportunity to identify malicious traffic. This is an extremely critical issue that could cause an attacker's activity to go unnoticed.

Detect was generated by

This detect deals with two particular alerts that are extremely prone to false positives. So much so, that they generated half of the alerts within this analysis. The two rules in question and the amount of alerts they generated are illustrated in Figure 9.

Alert Name	Number of occurrences	Number of Sources	Number of Destinations
MY.NET.30.3 activity	22862	132	1
MY.NET.30.4 activity	15942	194	1

Figure9

The rules MY.NET.30.3 and MY.NET.30.4 are very similar. Both appear to alert on any activity destined to these addresses from an outside source. The intention is most likely to ensure that all data to these hosts is captured. It may sound good in theory, but in reality, all this accomplishes is alert logs filled with alerts that require no attention. The MY.NET.30.3 rule for example contains 22,862 alerts. 13,447 of those alerts are from external sources connecting to port 524. Port 524 is used for the Netware Core Protocol used in Novell implementations. In this case, this very well might be students or staff members connecting to the university network to work from home.

The rule, MY.NET.30.4, contains over 3,000 alerts for connections made to port 524 as well. Along with port 524, the top port accessed on this server is port 51443, which is used in a Novell implementation of an SSL enabled webserver. This traffic was responsible for over 8,000 alerts.

Between these two rules, alerts generated from access to port 524 and 51443 total over 24,000. This is a massive amount of alerts that do no need to be logged by the intrusion detection system. While it may be important to log activity on these servers to those ports, it could be done on the hosts systems themselves, or offloaded to another system for the strict purpose of monitoring connections to this host.

By keeping these rules in place, malicious traffic may become harder to detect due to the sheer volume of alerts an analyst must review. These rules are examples of what not to do with an intrusion detection system.

Probability the source address was spoofed

Since this detect is not formatted by conventional means, no data is relevant for this section. Since both of these rules alert on any traffic to the host, it could be assumed that if source addresses were spoofed then that traffic would still be logged by the intrusion detection system.

Attack Mechanism

The attack mechanism is brought on by the functionality of the rules. As previously discussed, these rules generate so many alerts that it becomes more and more difficult for an analyst to review and assess log files in a timely manner. This has the potential to allow malicious traffic to go unnoticed while the analyst is busy sifting through alerts generated by false positives or valid traffic.

Correlations

Pete Storm discusses the MY.NET.30.4 and MY.NET.30.3 alerts in his GCIA Practical Exam. Therein he describes most of the alerts as benign, and questions the purpose of these rules.
http://www.giac.org/practical/GCIA/Pete_Storm_GCIA.pdf

David Barroso identified traffic for these alerts as predominantly Novell related in his GCIA Practical Exam.
http://www.giac.org/practical/GCIA/David_Barroso_GCIA.pdf

Severity

Severity = (Criticality + Lethality) – (System countermeasures + Network countermeasures)

Criticality = 4
While this does not impact network resources directly, it does damage the ability of university staff members to provide regular analysis of the network's intrusion detection logs. Since this analysis is critical to business operations at the university, an elevated value is assigned here.

Lethality = 4
Again, this does not have an effect on the network or its operations directly, but produces an environment that makes it more difficult for an analyst to identify threats based of the intrusion detection logs. If this situation does occur, an attacker may go unnoticed in their efforts to compromise the university's network resources.

System Countermeasures = 3
The particular system in question is the IDS. The rules of the IDS are being evaluated in this case. Since some of the rules are valid and produce accurate alerts, while others produce far too many false positives, a value of three is assigned here.

Network Countermeasures = 3
Once again the IDS and its rules are under the microscope. Because the IDS is part of the network's countermeasures it is difficult to accurately assign a value here. The IDS does function as an effective countermeasure, but not at its full capacity based on the amount of false positives examined during this analysis. Therefore a score of three is assigned.

Based on the previously mentioned formula, the severity level assigned for this issue is two.

2.3 Network Topology and Link Graph

To understand the significance of intrusion logs and how they relate to the network, a network topology must be envisioned. In this case the topology is a simple one. After analysis of all three types of log files it would appear that all machines on the university network are publicly accessible. Since very little packet data is available for analysis, it is difficult to ascertain much of the network topology. It is unclear if subnets are segmented, or where the perimeter actually is. Furthermore, it is difficult to determine the placement of the intrusion detection system. Is it placed on a core switch, or are there several sensors placed throughout the network? Based on the information gathered, it is not possible to conclude this information. Due to the apparent lack of filtering it may be assumed that there is no firewall, and no filtering done on the border routers. It is also impossible to tell if valued hosts, such as MY.NET.30.3 and MY.NET.30.4, are placed in a DMZ or an otherwise separated network.

Perhaps if more information were gathered by the IDS it would be possible to draw an accurate network diagram. Raw packet captures would be an outstanding source of information as TTL values can play a significant part in determining router or gateway placement.

A link graph is provided in Figure 10. This graph details the relationships between the attackers and the target hosts from the three previous in-depth analyses.

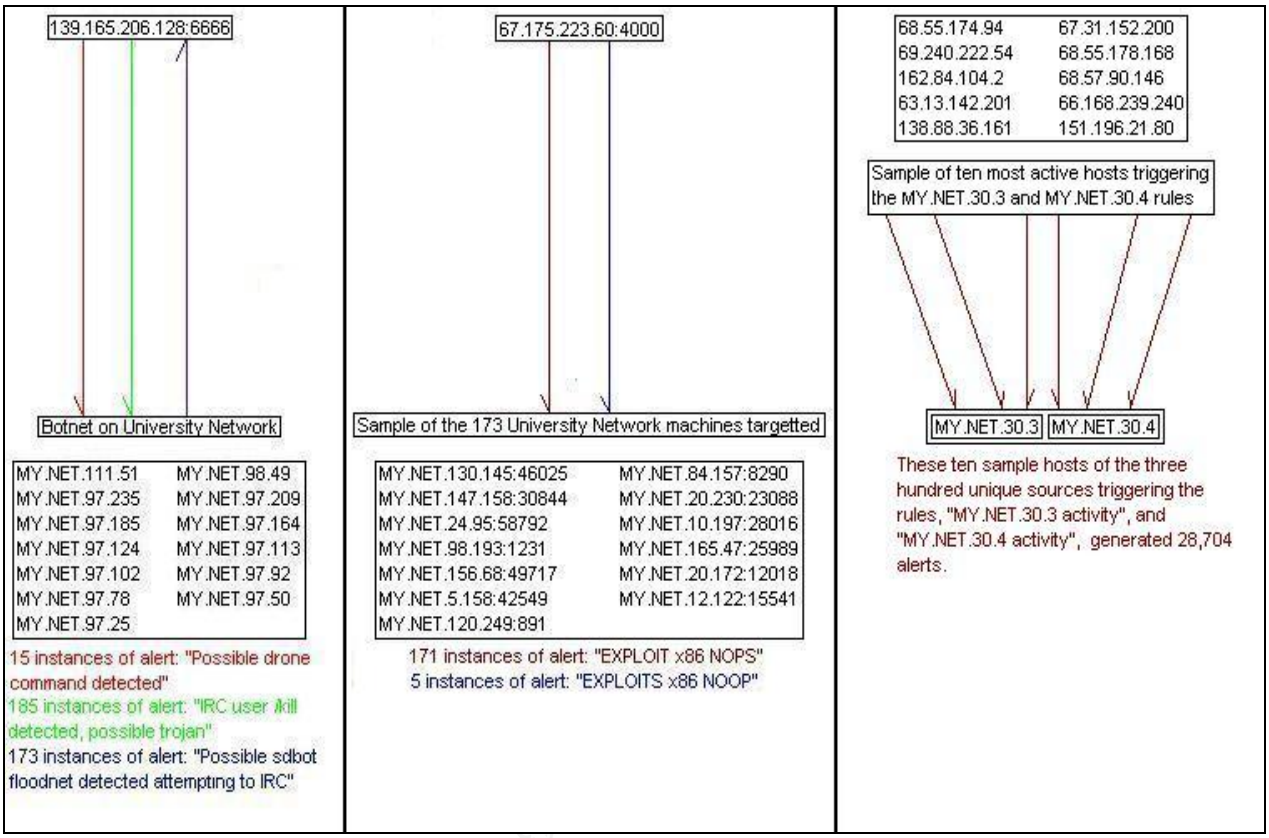

Figure10

The first section deals with the Botnet activity analyzed in the first detect. It shows the infected internal machines connecting to the IRC server hosting the Botnet. Directional arrows detail the traffic, and subsequent alerts generated by the hosts involved.

The second graph shows the machine responsible for the attempted propagation of the Witty worm that was detailed in the second in-depth analysis. Only a sample of the attacked hosts from the university network is shown here. It's important to note here that five instances of the "NOOP" alerts are shown. The "EXPLOIT x86 NOOP" alert is a variation of the "NOPS" alert that was discussed during the second analysis.

The final graph shows a sample of the over three hundred hosts that triggered the rules, "MY.NET.30.3 activity" and "MY.NET.30.4 activity". These ten hosts alone are responsible for 28,704 alerts, most of which are false positives or valid traffic. This supports the results found in the third in-depth analysis, which concluded that these two rules trigger far too many alerts.

2.4.1 Network Statistics

Different network statistics for the three-days worth of log files analyzed was presented in several ways. First being a "Top Five Talkers" list. Here the most active hosts in terms of traffic are shown. Since TCP and UDP are quite different, two tables are presented showing the top five hosts in terms of traffic generated per each protocol. The analysis is further expanded to show top internal and external hosts separately. All data was obtained from the scan logs. The following tables display this information.

Top Five External TCP Sources	
Host IP address	**Number of packets**
213.180.193.68	145,092
67.31.152.200	59,146
210.139.118.246	44,153
68.66.247.59	36,051
80.203.201.148	34,085

Top Five Internal TCP Sources	
Host IP Address	**Number of packets**
MY.NET.190.92	2,820,176
MY.NET.97.209	1,036,675
MY.NET.34.14	192,492
MY.NET.84.235	100,758
MY.NET.97.242	62,020

Top Five External UDP Sources	
Host IP Address	**Number of packets**
69.27.160.145	866
66.218.70.45	259
66.47.54.6	247
211.181.212.185	242
220.164.170.59	236

Top Five Internal UDP Sources	
Host IP Address	**Number of packets**
MY.NET.1.3	2,661,743
MY.NET.1.4	667,950
MY.NET.97.52	271,928
MY.NET.84.235	209,038
MY.NET.70.229	152,928

The following tables deal with top five services targeted. Again this data is broken into TCP and UDP, and internal and external hosts.

Top Five Targeted Services from External Sources - TCP	
Port / Service	**Number of packets**
6129 – Dameware	223,180
20168 – LovGate worm backdoor	149,224
80 – HTTP	99,286
4899 – Remote admin	66,980
4000 – Possibly ICQ	63,288

Top Five Targeted Services from Internal Sources - TCP	
Port / Service	**Number of packets**
135 – MS Remote Procedure Call Service	1,408,985
445 – MS Directory Services	1,403,921
2745 – Bagle worm backdoor	465,725
25 – SMTP	360,040
1025 - MS Remote Procedure Call Service	289,896

Top Five Targeted Services from External Sources - UDP	
Port / Service	**Number of packets**
137 – NetBIOS Name Service	2,917
5000 – Possibly Sockets de Trois Trojan	259
0 – Scan or DoS attempt? Reserved port	183
53 – DNS (Domain Name Service)	169
1608 – Smart Corp. License Manager?	162

Top Five Targeted Services from Internal Sources - UDP	
Port / Service	**Number of packets**
53 – DNS (Domain Name Service)	3,304,614*
4672 – eMule P2P	267,850
4673 – eMule P2P	75,626
41170 – Piolet P2P	36,043
123 – NTP (Network Time Protocol)	20,429

*This figure may be misleading since almost all of the packets logged came from two internal servers (MY.NET.1.3 and MY.NET.1.4). Most destinations seemed to be valid DNS servers as well. Therefore, this appears to be legitimate DNS traffic from the university's DNS servers.

2.4.2 Suspicious External Hosts

The three most suspicious external hosts have been selected, and require special attention. Their activity is so suspicious in fact that it would be appropriate to filter packets from these sources at the perimeter devices.

139.165.206.128

This host was the Botnet IRC server from the first in-depth analysis. It would be advised that all packets to and from this host be filtered at perimeter devices. Internal hosts are connecting to this IRC server in a manner consistent with Botnet activity. More information on this host is displayed in the following registration information.

```
OrgName:    RIPE Network Coordination Centre
OrgID:      RIPE
Address:    Singel 258
Address:    1016 AB
City:       Amsterdam
StateProv:
PostalCode:
Country:    NL

ReferralServer: whois://whois.ripe.net:43

NetRange:   139.164.0.0 - 139.166.255.255
CIDR:       139.164.0.0/15, 139.166.0.0/16
NetName:    RIPE-ERX-139-164-0-0
NetHandle:  NET-139-164-0-0-1
Parent:     NET-139-0-0-0-0
NetType:    Early Registrations, Transferred to RIPE NCC
Comment:    These addresses have been further assigned to users
in
Comment:    the RIPE NCC region.  Contact information can be
found in
Comment:    the RIPE database at http://www.ripe.net/whois
RegDate:    2004-03-03
Updated:    2004-03-03

# ARIN WHOIS database, last updated 2004-10-14 19:10
# Enter ? for additional hints on searching ARIN's WHOIS
database.
% This is the RIPE Whois secondary server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/db/copyright.html
inetnum:      139.165.0.0 - 139.165.255.255
netname:      UOFLIEGE-BE
descr:        Universite de Liege (ULg)
country:      BE
admin-c:      SU25-RIPE
tech-c:       SU25-RIPE
status:       ASSIGNED PI
remarks:      ---------------------------------
remarks:      In case of abuse, please contact:
remarks:      abuse@ulg.ac.be
remarks:      ---------------------------------
mnt-by:       BELNET-MNT
changed:      pirard@vm1.ulg.ac.be 19910327
changed:      piet@cwi.nl 19910404
changed:      Stephan.Biesbroeck@belnet.be 19930915
changed:      Marc.Roger@belnet.be 19980721
changed:      ad.hm@belnet.be 20031024
changed:      er-transfer@ripe.net 20040303
changed:      ad.hm@belnet.b 20040315
source:       RIPE

route:        139.165.0.0/16
descr:        UOFLIEGE-BE
origin:       AS2611
mnt-by:       BELNET-MNT
changed:      stephan@belnet.be 19950831
changed:      Eric.Luyten@belnet.be 19960419
changed:      Marc.Roger@belnet.be 19980721
source:       RIPE

role:         SEGI ULG
address:      Service General d'Informatique
address:      Universite de Liege
address:      B26 Sart Tilman
address:      B-4000 Liege
address:      Belgium
phone:        +32 4 3664904
fax-no:       +32 4 3662920
e-mail:       ripe@segi.ulg.ac.be
trouble:      call
admin-c:      FB7-RIPE
admin-c:      DK1178-RIPE
tech-c:       MF2348-RIPE
nic-hdl:      SU25-RIPE
mnt-by:       BELNET-MNT
changed:      ad.hm@belnet.be 20031024
source:       RIPE
```

67.31.152.200

This host, which was part of the "Top Five Talkers" list for external TCP scans, performed extensive port scanning on the MY.NET.34.0/24 and MY.NET.30.0/24 networks. This type of activity is usually a precursor to more attacks. This is simply the reconnaissance phase for an attacker who may be planning to attack the university network. Following is the registration information for this host.

OrgName: Level 3 Communications, Inc. OrgID: LVLT Address: 1025 Eldorado Blvd. City: Broomfield StateProv: CO PostalCode: 80021 Country: US NetRange: 67.24.0.0 - 67.31.255.255 CIDR: 67.24.0.0/13 NetName: LC-ORG-ARIN-BLK3 NetHandle: NET-67-24-0-0-1 Parent: NET-67-0-0-0-0 NetType: Direct Allocation NameServer: NS1.LEVEL3.NET NameServer: NS2.LEVEL3.NET Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE RegDate: 2001-11-07 Updated: 2002-08-08	TechHandle: LC-ORG-ARIN TechName: level Communications TechPhone: +1-877-453-8353 TechEmail: ipaddressing@level3.com OrgAbuseHandle: APL8-ARIN OrgAbuseName: Abuse POC LVLT OrgAbusePhone: +1-877-453-8353 OrgAbuseEmail: abuse@level3.com OrgTechHandle: TPL1-ARIN OrgTechName: Tech POC LVLT OrgTechPhone: +1-877-453-8353 OrgTechEmail: ipaddressing@level3.com OrgTechHandle: ARINC4-ARIN OrgTechName: ARIN Contact OrgTechPhone: +1-800-436-8489 OrgTechEmail: arin-contact@genuity.com # ARIN WHOIS database, last updated 2004-10-14 19:10

213.180.193.68

This host is also a port scanner. It only targeted two hosts on the university network, but was much more thorough in port scanning these hosts than the previous suspicious host. This suggests the attacker isn't targeting the university network, but these two hosts for some reason. Since port scanning is a precursor to attack, it would be advised that this host be filtered at the perimeter. It may also be prudent to investigate these two hosts, and find out what made them such a target for this individual. The targeted hosts were, MY.NET.25.68 and MY.NET.25.10. Following is the registration information for this suspicious host.

```
OrgName:    RIPE Network Coordination Centre
OrgID:      RIPE
Address:    Singel 258
Address:    1016 AB
City:       Amsterdam
StateProv:
PostalCode:
Country:    NL

ReferralServer: whois://whois.ripe.net:43

NetRange:   213.0.0.0 - 213.255.255.255
CIDR:       213.0.0.0/8
NetName:    RIPE-213
NetHandle:  NET-213-0-0-0-1
Parent:
NetType:    Allocated to RIPE NCC
NameServer: NS-PRI.RIPE.NET
NameServer: NS3.NIC.FR
NameServer: SUNIC.SUNET.SE
NameServer: AUTH00.NS.UU.NET
NameServer: SEC1.APNIC.NET
NameServer: SEC3.APNIC.NET
NameServer: TINNIE.ARIN.NET
Comment:    These addresses have been further assigned to users
in
Comment:    the RIPE NCC region. Contact information can be
found in
Comment:    the RIPE database at http://www.ripe.net/whois
RegDate:
Updated:    2004-03-16

# ARIN WHOIS database, last updated 2004-10-14 19:10
# Enter ? for additional hints on searching ARIN's WHOIS
database.
% This is the RIPE Whois secondary server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/db/copyright.html
```

```
inetnum:      213.180.192.0 - 213.180.193.255
netname:      COMPTEK-NET1
descr:        CompTek International/Yandex LLC
descr:        3, Gubkina str., Moscow, 117809
country:      RU
admin-c:      YNDX1-RIPE
tech-c:       YNDX1-RIPE
status:       ASSIGNED PA
notify:       noc@yandex.net
mnt-by:       YANDEX-MNT
changed:      wawa@comptek.ru 20020607
changed:      gvs@yandex-team.ru 20040625
source:       RIPE
route:        213.180.192.0/20
descr:        Yandex enterprise network
origin:       AS13238
notify:       noc@yandex.net
mnt-by:       YANDEX-MNT
changed:      wawa@comptek.ru 20010123
changed:      gvs@yandex-team.ru 20040625
source:       RIPE

role:         Yandex LLC Network Operations
address:      Yandex LLC
address:      40A Vavilova st.
address:      117333, Moscow, Russia
phone:        +7 095 9743555
fax-no:       +7 095 9743565
e-mail:       noc@yandex.net
trouble:      ------------------------------------------------------
trouble:      Points of contact for Yandex LLC Network
Operations
trouble:      ------------------------------------------------------
trouble:      Routing and peering issues:  noc@yandex.net
trouble:      SPAM issues:                 abuse@yandex.ru
trouble:      Network security issues:     abuse@yandex.ru
trouble:      Mail issues:                 postmaster@yandex.ru
trouble:      General information:         info@yandex.ru
trouble:      ------------------------------------------------------
admin-c:      VLI1-RIPE
admin-c:      GVS-RIPE
tech-c:       KBG2-RIPE
notify:       noc@yandex.net
nic-hdl:      YNDX1-RIPE
mnt-by:       YANDEX-MNT
changed:      gvs@yandex-team.ru 20040625
source:       RIPE
```


2.5 Possible compromised internal hosts

The first in-depth analysis turned up quite a few hosts on the university network that have most likely been compromised. Further analysis shows quite a few hosts on the university network that may have been compromised.

Looking at the table, “Top Five Targeted Services from Internal Sources – TCP”, shows signs that some hosts on the internal network have been compromised. Take special note of the almost three million scans to TCP ports 135 and 445. There is no valid reason why internal hosts should be accessing external hosts on these ports. These scans are most likely the result of worm or virus infections. Many of the recent MS Windows exploits have targeted these ports. At the time the culprits included worms like Blaster and Nachi.

The host that was responsible for this type of traffic is MY.NET.190.92. Over the three-days worth of log files, this host scanned a wide array of external machines on TCP port 135 and 445.There were 1,408,986 scans on port 135, and 1,403,923 scans on port 445 to be exact. This type of activity proves this host has been compromised.

Another from the Top Five Talkers list is almost assuredly compromised. The host, MY.NET.97.209, was responsible for one million scans during the three-day period. There is a definite pattern to this host's scanning. Figure 11 shows this activity from the scan logs.

```
Mar 25 21:22:37 MY.NET.97.209:3685 -> 128.30.69.22:80 SYN ******S*
Mar 25 21:22:37 MY.NET.97.209:3682 -> 128.30.69.22:3127 SYN ******S*
Mar 25 21:22:37 MY.NET.97.209:3678 -> 128.30.69.22:2745 SYN ******S*
Mar 25 21:22:37 MY.NET.97.209:3683 -> 128.30.69.22:6129 SYN ******S*
Mar 25 21:22:37 MY.NET.97.209:3680 -> 128.30.69.22:1025 SYN ******S*
```

Figure11

This type of activity is most likely associated with the Agobot worm. The scans in Figure 11 show the worm in its propagation stage as it attempts to infect other hosts via those ports seen in the scan. This host has definitely fallen victim to this worm, and should be investigated.

The Top Talkers list has proven to be invaluable in locating internal compromised hosts. This final host is responsible for nearly 200,000 scans to a multitude of external servers via SMTP. From DNS information about the university it does not appear that this host is a valid SMTP server for the university. This leads one to question why it's sending so many packets via TCP port 25. This is most likely the case of a mass mailing worm using its own SMTP engine. Worm propagation of this type would account for the massive amount of connections to other servers via SMTP. It can be assumed that this host, MY.NET.34.14, has been compromised, and requires investigation. Scan logs show this host connecting to external servers in Figure 12.

```
Mar 25 22:11:10 MY.NET.34.14:54833 -> 128.52.33.12:25 SYN ******S*
Mar 25 22:11:10 MY.NET.34.14:54835 -> 64.251.8.4:25 SYN ******S*
Mar 25 22:11:10 MY.NET.34.14:54837 -> 130.94.132.33:25 SYN ******S*
Mar 25 22:11:11 MY.NET.34.14:54942 -> 209.133.28.19:25 SYN ******S*
Mar 25 22:11:11 MY.NET.34.14:54935 -> 64.26.62.254:25 SYN ******S*
```

Figure12

2.6 Defensive Recommendations

The preceding analysis would be worthless if defensive recommendations were not provided. Defensive recommendations for the university's network are done in four parts. The first three are immediate response recommendations for the alerts analyzed during this analysis. The last section is advice on general security practices the university should take in order to maintain a more secure network.

Botnet activity

It is imperative that this Botnet is dismantled as soon as possible for reasons already stated in this analysis. If possible, these machines should be taken offline for further investigation, and possible rebuilding of the host's operating system. Once rebuilt these hosts should be brought up to appropriate patch levels according to vendor security updates. Users of these machines should then be educated on how to avoid infection from these types of Trojans. This may include a short class in safe Internet use.

Traffic to and from the host IRC server for the Botnet should also be blocked at border devices. This could help prevent further compromise since no internal machines could connect to that IRC server. This would also assist in locating other compromised internal hosts who may be attempting to connect to that IRC server since firewall or router log files would show these packets being dropped.

EXPLOIT x86 NOPS (Witty Worm)

If any of the affected ISS products are installed on the network, they should be patched according to ISS as soon as possible. While patching and upgrading is being completed, it would be prudent to filter all inbound packets with a source port of 4000/UDP. While this may disrupt normal ICQ operations, it is this analyst's opinion that it would be necessary while the threat of infection remains. Once patching and upgrading of affected ISS products is complete then those ICQ ports may be reopened.

Excessive amount of false positives

As discussed, the amount of false positives created by the rules, "MY.NET.30.3 activity" and "MY.NET.30.4 activity", are a severe disruption to IDS functionality. These rules should be reevaluated as soon as possible. If logging all connections to these hosts is necessary, then that duty should be offloaded to a dedicated monitoring device, or the hosts themselves. During this process it may be a good idea to evaluate all IDS rules, and ensure that these rules report only meaningful alerts. This will streamline the intrusion detection process, and create more significant log files.

General Security Recommendations

A device filtering packets at the perimeter could have stopped many of the packets triggering alerts analyzed over the three-day period. A firewall or router with appropriate access-lists would go a long way in securing the university's network. This is one of the most basic forms of security measures an organization can take to protect their network. Relevant personnel should discuss perimeter filtering, and a plan to incorporate this into the network should be made a priority.

The network topology should be reassessed as well. Perhaps it is not in the university's best interest to have an entire network that is publicly accessible. Instead of public IP addresses, NAT could be deployed in order to add to security, as well as cut costs; all those IP addresses cost money.

The network could also be segmented in order to keep more sensitive hosts or networks separate from more "general use" hosts or networks. Servers that need to be made public should be put into a DMZ where they could remain publicly accessible, but also remain separate from the internal university network.

It would also appear that some internal machines fell victim to viruses and worms. Perhaps relevant personnel could create a type of incident response package that could be made available to the users of the university's network resources. In case of a new vulnerability or worm, a notice would be sent out to all users. This would give them a head start on looking out for the new issue, and perhaps avoid infection.

These are some basic steps to take toward creating a more secure environment. Staff members should discuss these suggestions to see if they could find a way to integrate them into the university network.

3. Analysis Process

One of the more challenging issues that had to be overcome was being able to manage the sheer volume of information within the three-days worth of log files. Each type of log file presented its own unique challenge as well. Thankfully, some very interesting tools are available that allow an analyst to tackle these issues. Some are freely available scripts, such as Snortsnarf, while other techniques were drawn upon from other GCIA practical exams.

This analysis was done primarily on a FreeBSD 4.10 machine. It had a 1.3 GHz Pentium 4 processor and had 256 MB of RAM.

The alert files were generated from an older version of the Snort intrusion detection system. The three alert files were first combined into one large alert file for analysis. This was accomplished using the cat command as seen here: *cat alert.040325 alert.040326 alert.040327 >> alert_full.* The combined alert file was approximately 70MB. It was this analyst's intention to use a popular Perl script called Snortsnarf to parse and examine the alert file. Snortsnarf is designed to take Snort alert files, and generate HTML pages based on the alert file's contents. This allows an analyst to examine the alerts, and the relationships between the hosts in an easy to read format. The alert file did require some editing since the first two octets of the university IP addresses were replaced with "MY.NET". If left as they were, Snortsnarf would not be able to parse those IP addresses correctly. The first task was to replace the "MY.NET" portions with numerical values so as to match an IP address. A quick Perl script was written to take care of that. This script, simply called *replace.pl* is in the appendix.

At this point the alert file was 70MB, and due to memory limitations, Snortsnarf was unable to parse such a large file. Since most of the alerts were related to the spp_portscan preprocessor, they could be stripped from the alert file. Remember that all the data from the portscan alerts is found the scan files. The portscan alerts were stripped from the alert file using another Perl script very similar to *replace.pl*. This analyst, who is renown for his creativity, called this script *replace2.pl*. This resulted in two alert files, one containing all the scan alerts, and the other containing every other alert. Without all the portscan alerts the new alert file totaled only 7MB. Snortsnarf was able to deal with this file size, and generated some nice HTML pages based on the alert files. This created a manageable way to analyze the alert files.

The Out of Spec files proved to be quite a challenge until Ricky Smith's GCIA practical came into view. He wrote an outstanding Perl script that changes the format of the OOS files into ones similar to the alert files. Thanks Ricky! Once that was done, Snortsnarf was used again to create a nice HTML report based on the OOS files. The three OOS files were joined into one larger OOS using the same technique as with the alert files prior to parsing them through Snortsnarf. OOS files also contained obfuscated IP addresses for the university. The same technique to replace "MY.NET" with numerical values that was used with the alert file was used with the OOS file.

Finally the monstrous scan files had to be dealt with. Combining them using the cat command created one scan file of immense proportions. The resulting scan file was 651MB! Dealing with a 651MB log file is a difficult task, one that is made easier though if that file is split into more meaningful files. Using two Perl scripts, (pull.pl and pull2.pl) the scan file was broken down into four different files. It was split in two based on TCP

and UDP scans. Those two were then split based on source, one file for external sources, and one for internal. This technique resulted in four smaller, and much more manageable files.

The actual process of analysis for the alert and OOS files was quite simple since they were presented in easy to read HTML pages. Snortsnarf proved to be an invaluable tool for alert and OOS file analysis. The HTML pages are easy to read, and are linked together in such a fashion that allows an analyst to easily gauge the relationships between the alerts, attackers, and targeted hosts.

The scan files were a different story, but were easy to manage and analyze once a technique was ironed out. No applications or Perl scripts were required here, just the use of some common Unix based commands. These commands were cat, grep, egrep, cut, sort, and uniq. Using these in combination provided a quick way to sort through the scan files to find interesting hosts, and traffic trends. Some samples of the usage of these commands, and their results, are detailed in the appendix.

Passive reconnaissance was used in order to verify certain servers on the university network were or were not valid SMTP or DNS servers. Unlike alert and OOS files, the scan files had the university's real IP addresses instead of the "MY.NET" string. This allows an analyst to query DNS servers for records, such as MX and NS, relating to the university. This provides a bit more insight into actual hosts within the university network. Keeping with the standard of obfuscated IP addresses for the university, the details of the DNS queries will not be shown in this analysis.

Correlations and References

1. Snort - http://www.snort.org
2. SnortSnarf - http://www.snort.org/dl/contrib/data_analysis/snortsnarf/
3. IRC /kill command - http://library.n0i.net/irc/irchelp/irc_help4.html
4. Agobot - http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?id=37776
5. arachNIDS NOPS rule - http://www.whitehats.com/info/IDS362
6. ISS - http://xforce.iss.net/xforce/alerts/id/166 , http://xforce.iss.net/xforce/alerts/id/167
7. Symantec - http://securityresponse.symantec.com/avcenter/venc/data/w32.witty.worm.html
8. Witty Worm - http://www.lurhq.com/witty.html
9. http://www.giac.org/practical/GCIA/David_Barroso_GCIA.pdf
10. http://www.giac.org/practical/GCIA/Andrew_Evans_GCIA.pdf
11. http://www.giac.org/practical/GCIA/Donald_Parker_GCIA.pdf
12. http://www.giac.org/practical/GCIA/Ricky_Smith_GCIA.pdf
13. http://www.giac.org/practical/GCIA/Pete_Storm_GCIA.pdf

Appendix

replace.pl – Designed to read a file and replace "MY.NET" strings with a numerical value.
Usage: replace.pl <logfile>

```
#!/usr/bin/perl
open LOG,(">>./alert_full.log");

while (<>) {
      if (/MY\.NET/) {
            s/MY\.NET/10.10/g;
              print LOG;
      }
}
close LOG;
```

replace2.pl – Designed to write two logfiles from combined alert file. One file contains the portscan alerts; the other contains all other alerts.
Usage: replace2.pl <logfile>

```
#!/user/bin/perl
open SCAN, (">>./alert_scan.log");
open NOSCAN, (">>./alert_noscan.log");
while (<>) {
      if (/spp_portscan/) {
            print SCAN;
      } else {
            print NOSCAN;
      }
}
close SCAN;
close NOSCAN;
```

pull.pl – Designed to write two logfiles from combined scan files. One file contains UDP scans while the other contains TCP scans.
Usage: pull.pl <logfile>

```
#!/usr/bin/perl
open TCP,(">>./tcp_scans.log");
open UDP,(">>./udp_scans.log");

while (<>) {
      if (/UDP/) {
            print UDP;
      } else {
            print TCP;
      }
}
close TCP;
close UDP;
```

pull2.pl – Designed to separate TCP and UDP scans files into internal and external scans. This script can be used for both TCP and UDP files. Just make sure to change the logfile output.
Usage: pull2.pl <logfile>

```
#!/usr/bin/perl
open INT,(">>./int_tcp_scans.log");
open EXT,(">>./ext_tcp_scans.log");
```

```
while (<>) {
      if (/10\.10\.\d{1,3}\.\d{1,3}:\d{1,5} -/) {  # replace the tens with the university IP address
            print INT;
      } else {
            print EXT;
      }
}
close INT;
close EXT;
```

Parsing scan files using cat, grep, egrep, cut, sort, uniq

Sample output from scan file:
Mar 25 00:11:41 65.147.113.162:3655 -> MY.NET.190.159:135 SYN ******S*

Searching scan files for a particular IP address is done so using grep (self-explanatory)

Searching for 65.147.113.162 as source:
egrep '65.147.113.162:[0-9]{1,5} -' scans.log
Output:
Mar 25 00:11:42 65.147.113.162:3734 -> MY.NET.190.238:135 SYN ******S*

Searching for 66.7.128.123 as destination:
grep '> 66.7.128.123' scans.log
Output:
Mar 25 21:38:28 MY.NET.34.14:52702 -> 66.7.128.123:25 SYN ******S*

Finding unique destinations from source 213.180.193.68, and sorting by number of occurrences:
egrep '213.180.193.68:[0-9]{1,5} -' scans.log | cut -d ' ' -f 6 | cut -d : -f 1 | sort | uniq -c | sort -n
Output:
71018 MY.NET.25.68
74074 MY.NET.25.10

Finding unique destination ports from source 68.66.247.59:
egrep '68.66.247.59:[0-9]{1,5} -' scans.log | cut -d ' ' -f 6 | cut -d : -f 2 | sort | uniq -c | sort –n
Output:
11993 10080
12019 1080
12035 3128

Using cut, sort and uniq allows an analyst to perform a great many queries on the scan files. To understand this concept, it's easiest to go step by step. In the last example, egrep displays all lines that have 68.66.247.59 as the source IP address. That output is then piped through the first cut. The d switch tells the delimiter to use in the cut is a space. The field to cut is number 6, which is the destination IP address and port. Just count the spaces to the field required. The next cut uses a delimiter of a colon. Here the field choice of one will be the IP address while the field of two will be the port number. Whichever is chosen, the next step is to sort. This is used since only sorted data can be piped to the uniq command. The uniq command will display unique occurrences of the data piped from the sort command. The c switch is used to count how many times that unique occurrence was made. Finally, another sort command is used, this time with the n switch, to sort the new unique data numerically. This sorts the data by how often it appeared, and is a great way of determining trends in traffic. The above examples are only a few of the ways to use those commands. A wide variety of combinations can be used in order to obtain a wide variety of data. Consult the man pages for exact usage of these commands.