Global Information Assurance Certification Paper

Intrusion Report for SANS University

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment Version 4.1

Gaspar Modelo Howard

SANS Washington D.C. 2004

Table of Contents

Part I – Executive Summary

We were commissioned to analyze the log files created from the intrusion detection system (IDS) of SANS University, between March 7 and March 10, 2004. This document presents a summary of the findings. It is our recommendation to promptly implement several security controls, based on the malicious activity found and possible compromised of several hosts inside the network.

The activity registered in the log files shows a very active University network with many assets to protect, services to maintain and users to support. The logs showed that this network is continuously under attack, by malicious users who want to probe, compromise and take illegal advantage of the resources available. It is very important that the university has a complete security strategy to protect its assets.

We found several hosts that we believe are compromised. Unauthorized software that allowed the remote control of the hosts or generated big amounts of traffic are part of the evidence. Our analysis shows that most of these hosts were in networks used by regular users like students and administrative personnel: MY.NET.42.0, MY.NET.53.0, MY.NET.70.0 and MY.NET.80.0. We recommend doing a complete cleaning of these hosts, hardening their operating systems and installing anti virus software to prevent future compromises.

The IDS, although provided valuable information, should be tuned to prevent from reporting false positives. Examples were real mail traffic being reported as worms, showing legal remote access to Novell servers as thousands of alerts and good web requests to University's websites as traffic caused by hosts infected with Red Worm. The IDS should also be configured to run with tools that will automate the analysis process of the logs. Hundreds of megabytes are created daily and to process it manually requires too many people or time.

Another recommendation is to turn off unneeded services such as Windows neighborhood protocol, NetBios. The existence of the Netware servers allow for the university to provide centralized file management systems instead of allowing end users to create shared directories in every computer they use. Also, it is a good practice to block unnecessary ports with a firewall at the Internet connection point. Although University policies might want to keep networks as open as possible, a small search done by us on the Internet found out that similar institutions have been blocking unnecessary ports for quite some time. In their opinion this has not created commotion or problems among faculty and students.

The remaining of the document will provide a detailed analysis of the logs, our findings and more detailed recommendations. The report has been sanitized to protect the identity of the University. We appreciate the opportunity to perform this task and look forward to future engagements.

Part II – Detailed Analysis

The analysis is based on log files from a university network, captured by a Snort-based IDS for a period of four (4) days from March 7 to March 11, 2004. Three different files were generated for each day: alert, scans and out-of-specifications (OOS). The alert files are logs produced with the fast alert option (-A), the scans files refer to the snort portscan log files and the OOS files include the malformed packets, based on the RFC793 [ISI81] specifications.

The log files' names include their creation date, making it easy to determine the log files necessary to analyze a certain period of time. Nevertheless, there was an error on the OOS files. Although their names refer to a certain date, the actual logs inside the file refer to a date four days later. To correct this situation, we chose to work on the OOS files named for the period of March 3 to March 6, 2004. The logged packets on those files actually refer to the same period of the alert and scans files. It was not assumed that this situation was the result of lack of time synchronization on the IDS computers.

Following is a list of the log files analyzed along with their sizes and MD5 checksums:

File Type	File	Size (MB)	MD5 Checksum
Alert (Fast)	alert.040307	12.56	af13d2aa7098e08b88ee554be2cc7eae
Alert (Fast)	alert.040308	9.32	7cf61b162618def7282e6d031e4b76cf
Alert (Fast)	alert.040309	10.27	f991c87e430c8a889376307bfbad88ea
Alert (Fast)	alert.040310	9.70	368f1f25e8a15d09c81656c08890a3da
Out-of-Spec	oos_report_040303	0.20	4d5a0c0ad79c852bb2020d7a9ed7457f
Out-of-Spec	oos_report_040304	0.23	952921e2f849c2a738085137ebc53ab9
Out-of-Spec	oos_report_040305	0.16	9f3c7c7fd0e1ef707d072ebf99e67bf0
Out-of-Spec	oos_report_040306	0.14	016e0b855a554c9112eb4db896027a7c
Portscan	scans.040307	51.43	4a6aff8ffd3f3c989908d82cacd2a07d
Portscan	scans.040308	35.36	7a79e14b3e13b70554a0cd93f6e8c8a1
Portscan	scans.040309	47.50	bac194fe02e5015573276d900e82525b
Portscan	scans.040310	37.76	1ef0b693cc8ed8796935f8e263eb7c34

The scan files were not sanitized, which allowed us to determine the complete IP address space of the university network. This probed to be helpful since it allowed us to perform DNS queries to determine MY.NET. hostnames. Some of our conclusions are based on this information. Still, throughout this report we have kept hidden the identity of the university to protect its assets.

1. Network Topology

The university's network is composed of thousand of hosts, the vast majority using a

public Internet address on class-B network registered to the university. As expected in a university such as this one, there are many common services and hosts available: domain servers, web servers, file servers (FTP, directory-based systems), mail servers, network management servers, remote access servers and departmental servers. There were also subnets assigned to hosts for administration personnel and to laboratories for different departments.

The IDS logs showed an interesting farm of mail servers, based on traffic to ports related to mail protocols (for example 25/TCP, 110/TCP, and 143/TCP). A quick DNS name query for each of the IP addresses confirmed the existence of mail-related services on those servers. Examples of hostnames found were: mailserver-ng (MY.NET.34.14), imap (MY.NET.34.14), mxin (MY.NET.12.6), mx1del (MY.NET.6.47), smtp (MY.NET.12.2), mx1in (MY.NET.25.66), mx2in (MY.NET.25.66), and mx8in (MY.NET.25.73). Similar DNS queries helped to determine the hostname for machines possibly running the FTP service: ftp (MY.NET.6.63) and ftp1 (MY.NET.24.47). The logs showed that those servers were accessed on port 21/TCP.

A usual target of hackers and malicious code is a web server and there are many at this university's network, for example: www (MY.NET.24.34), my (MY.NET.24.33), userpages (MY.NET.24.44), www.gl ((MY.NET.12.12), webauth, (MY.NET.12.7) and linux3.gl ((MY.NET.60.39). The web servers are used to publish information, administer resources and also for tunneling other protocols or services. Although such abundance of web-based services should attract malicious traffic, the IDS logs showed no evidence of web attacks, nor could we determine the use of IDS rules for these situations.

The DNS servers are hosts MY.NET.1.3, MY.NET.1.4 and MY.NET.1.5. Several subnetworks could have been for shared computers, as in a laboratory or at students' room quarters. Examples of such subnets are MY.NET.80.0/24, MY.NET.70.0/24 and MY.NET.53.0/24. Also, a server with hostname oncampus.vpn (MY.NET.16.106) might be a VPN concentrator for remote users.

The snort IDS sensor that generated the logs for this project is probably located at a Internet gateway point, between the Internet and the university's internal network. All traffic logged by the sensor is either from an internal host to and Internet host or vice versa. No internal only traffic was detected with the IDS. A couple of possible external only packets were found, the majority had the source address from the AmericaOnLine address space (172.168.0.0 – 172.191.255.255) or the 192.168.0.0/16 address space reserved for private networks. The rule set of the IDS includes a collection of rules developed in-house along with several from the Snort [SNO05] and Arachnids [ARC05] database. A more detailed explanation of the rules is included in the Appendix A of this document.

2. Link Graph

The following link graph shows several connections between internal and external hosts. All have in common that are most probably communicating using IRC protocol (ports 6660/TCP to 7000/TCP). I consider the logs analyzed to have a good amount of traffic on this type of communication. The link graph below shows what could be two different IRC clients or bots, SDbot and XDCC. This means that different MY.NET. hosts are being used separately for different purposes. The SDbot is usually found in compromised hosts and allow the attacker to control remotely the host. The XDCC bot is used for transfer of copyright material and allows the attacker to use disk space for storage. It also allows detecting new possible hosts for more storage and transfer of files.

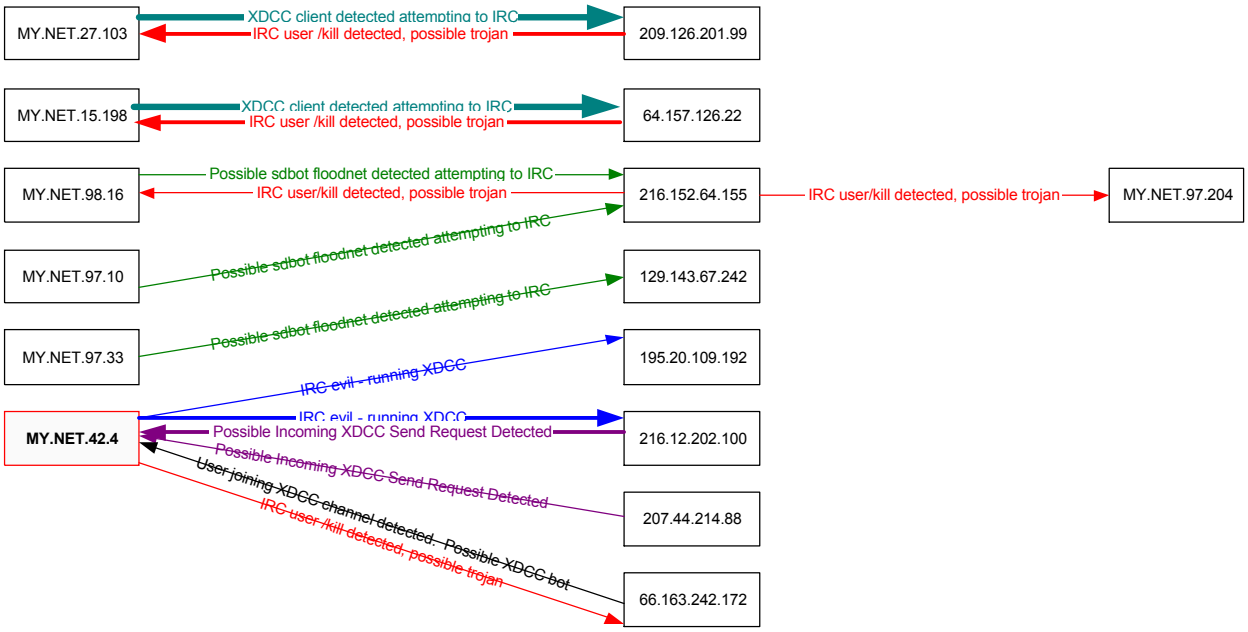


Figure 1. Link graph showing the IRC traffic generated between external hosts and selected MY.NET. network hosts. Such activity shows it is very likely that several internal hosts were compromised and/or had IRC bots installed.

The MY.NET.42.4 host is an example of the many hosts found in the MY.NET.42.0 network that exhibit the same traffic. These hosts are members of an IRC channel and probably have installed an XDCC bot. They probably have already been used for transfer of files and need to be cleaned by administrators. It is a good exercise to come up with thesis on how the attacker first compromises these systems. I have decided to select this detects for an in-depth analysis, later in this document.

3. Overview of Detects Identified and Number of Occurrences of Each Detect

49 different detects were logged on the files analyzed and generated 37674 alerts. Two situations creating more than 69% of those alerts: IRC traffic and traffic to servers

MY.NET.30.3 and 4. The IRC situation shows the special attention given by IDS administrators to IRC traffic, seven different alerts were created and accounted for approximately 15% of all alerts. Traffic to MY.NET.30.3 and MY.NET.30.4 servers generated more than 54% of all alerts. Still we believe that the majority of that traffic was false positive since both servers offered services to remote users and the possible IDS rules used to detect this traffic were too generic.

The rest of the detects showed suspicious activity related to different services: SunRPC, SMB, FTP, VNC, SMTP, TFTP and printer spooler (515/TCP). The FTP alerts showed several attempts to compromised or make unauthorized access to those servers. Still, we found no evidence of actual compromise. There were also some false positives, such as alerts for FTP traffic possibly generated by downloading software from an anti-virus provider FTP site.

The VNC traffic was interesting and could probably mean a backdoor installed after a server was compromised. Although VNC is a valid and helpful management tool for administrators, the internal addresses involved seem to be part of shared-use PC networks such as laboratories: MY.NET.70.0/24, MY.NET.149.0/24 and MY.NET.150.0/24. These networks usually have weaker security policies than more controlled environments such as the universities' data centers.

Below is a list of the top ten detects according to the number of alerts generated. The list also includes the number of hosts that generated the traffic (sources) and hosts that received the traffic (destinations):

SIGNATURE	NO. OF ALERTS	NO. OF SOURCES	NO. OF DESTS
130.85.30.4 activity	12103	225	1
130.85.30.3 activity	8357	120	1
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	5537	5	4
High port 65535 tcp - possible Red Worm - traffic	3631	81	96
OOS	2317	395	55
SMB Name Wildcard	2040	155	362
Null scan!	761	53	35
NMAP TCP ping!	513	123	47
EXPLOIT x86 NOOP	447	180	77
SUNRPC highport access!	303	25	66

The portscan files were not included in the list above but certainly were an important part of our analysis. Those files showed a mix of false positives and possible malicious traffic. We believe the false positives were the result of a possible pre-processor configuration that needs to be tightened. We found a portscan alert from twelve (12) packets generated during a period of over thirteen (13) minutes. Any server that would receive more than twelve service requests in a shorter period could trigger the pre-processor. For example, traffic that involved the MY.NET.30.3 and 4 servers were quite possibly false positives. Those two servers were in the top three of

machines generating portscan alerts.

The portscan files also showed possible malicious traffic from internal hosts. The majority comes from dialup and pooled networks such as MY.NET.97.0/24, MY.NET.82.0/24, MY.NET.80.0/24, MY.NET.98.0/24, MY. NET.70.0/24, and MY.NET.60.0/24. The other sources of portscan were external hosts, the top five machines were: 213.157.171.109 (Romania), 129.22.166.233 (Case Western Reserve University, US), 67.170.105.177 (AT&T, US), 134.2.78.155 (Germany), and 141.85.252.94 (Romania).

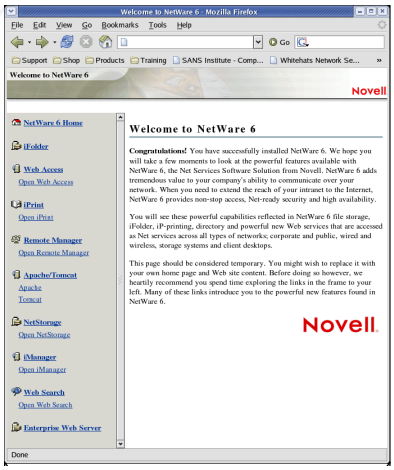
4. Three Critical Detects with In-Depth Analysis

Below there are three detects that we considered worth to analyzed in-depth. Alerts were grouped according to the activity they were designed to detect, so a detect will include more than one alert in two of the three detects chosen.

4.1 Detect No. 1: MY.NET.30.3 activity and MY.NET.30.4 activity

4.1.1 **Description of detect:** There are several detects under the ”MY.NET.30.3 activity” and ”MY.NET.30.4 activity” alerts. Most are related to Novell Netware 6 services [NOV01] such as a web server, Remote Manager and NetStorage. Both hosts are running Netware 6 as the operating system and seem to be the foundation for a network directory based on the NDS protocol. We also believe that several if not the majority of the alerts could be false positives. Still, the IDS should be tuned to detect non-authorized requests to these hosts because they offer administration services that could impact not only themselves but also the rest of the network directory.

The alerts show that one of the destination ports for both hosts was 80/TCP, commonly used for web servers. Using a web browser, we could find out that the web service is active and are most likely running Novell 6 on both hosts.



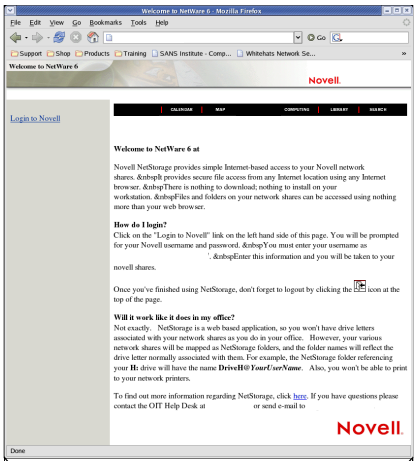


Figure 2. Start page for web servers found in MY.NET.30.3 (left) and MY.NET.30.4 (right) hosts. Both pages allow a visitor to perform a reconnaissance of possible services available on the hosts.

Below is a summary of the four most popular destination ports used on the MY.NET.30.3 and .4 hosts for the four days analyzed. They accounted for 99% of the "MY.NET.30.3 activity" and "MY.NET.30.4 activity" alerts:

PORT (TCP)	NETWARE DESCRIPTION	MY.NET.30.3 ALERTS	MY.NET.30.4 ALERTS
524	NDS protocol, client requests. Source ports should be a high port, above 1024.	7216	2382
51443	Novell Netware 6 secure (SSL) web server as found on MY.NET.30.3 and .4 hosts. Also allows the transmission of Netware services, such as NetStorage, with SSL	1	8818
8009	Netware Remote Manager, allows to secure (SSL) and remotely access Netware servers for management duties.	993	171
80	Apache web server or Novell own (Enterprise) web server, included with Netware 6.	96	678

The alert logs show that all alerts were triggered by external hosts, 225 sources for MY.NET.30.4 and 120 for MY.NET.30.3. There were no alert started from internal hosts. Although there could be some attacks alerted, we believe the majority could be from authorized hosts for remote access. We assume that the hosts are used for network operation purposes and not as a honeypot or decoy system. If the hosts are part of a honeypot, we provide more details in section 4.1.5 (Attack Mechanism) of this document.

Almost all of the top ten source hosts that triggered the alerts seem to be located around the state of Maryland, where the university is located. The MY.NET.30.3 and .4 hosts seem to be used for remote administration or access to the resources of the university's network so could be regular or administrative users accessing network resources through these servers. A whois query showed that source hosts came from Internet providers located in Washington, DC., Coudersport, PA, and Baltimore, MD. There were two providers registered in Denver, CO and Austin, TX but they also provide access in the Maryland area.

We found no indications on the Internet of worms or malicious code that could attack the Novell NDS or services only found in Netware products such as iPrint, iFolder or NetStorage. Therefore, some of the detects could only have been triggered by worms aimed at other systems that also offer services in same ports. One example could be the Nimda worm that targets port 80/TCP of windows boxes. The lack of worms for Netware could be the result of malicious code writers being more interested on more popular operating systems such as Windows and Linux.

4.1.2 **Reason this detect was selected:** Curiosity proved to be the main catalyst to perform an in-depth analysis on both alerts. Since the alert message for both signatures were very similar, I decided to do the analysis as one. The curiosity was based on the high number of alerts found, over 54% of all alerts reported by Snort fall in these two categories. Therefore, these detects were very noisy.

Another reason to choose these alerts is that they were consistently found on each of the four days analyzed, which could mean that the threat was permanent. If vulnerability were found on either of the ports probed, scanned or used by the threat, MY.NET.30.3 and MY.NET.30.4 hosts would be compromised promptly. This possibility was worrisome since having an alert exclusive for such hosts is a sign of its importance or role inside the network.

Finally, the destination ports used on both hosts were a mix of common and interesting ports such as 80/TCP and 1080/TCP and odd ports such as 524/TCP, 8009/TCP and 3128/TCP. Our purpose was to determine if most or all ports were used as part of a common attack or just the result of multiple, not related attacks.

4.1.3 **Detect was generated by:** Both detects were generated by the Snort intrusion detection system, using two rules created internally by people responsible for the configuration or administration of the IDS. The alerts were generated by signatures similar to the following:

- `alert tcp $EXTERNAL_NET any -> MY.NET.30.3 any (msg: "MY.NET.30.3 activity";)`

- `alert tcp $EXTERNAL_NET any -> MY.NET.30.4 any (msg: "MY.NET.30.4 activity";)`

4.1.4 **Probability the source address was spoofed:** Low or unlikely. To the majority of source hosts seemed necessary to receive packets back. First, there could be authorized accesses on the logs that would definitely not spoof their addresses. Also, the existence of web servers on both

hosts allow an attacker to list possible services available and the operating system running on hosts. This could make the attacker to probe or scan the hosts to determine and receive more information about the victims. Finally, all scans packets use TCP that is not commonly used for attacks where source addresses are spoofed.

4.1.5 **Attack mechanism:** The use of the Netware Directory Services requires the prior authentication of a user. Therefore, those web-based services could be subject to a dictionary or brute-force attack. An example is the NetStorage service, available through port 51443/TCP. Several alerts targeted at this port could be the result of such attack.

Using a tool such as Brutus [BRU01], an attacker could use the NetStorage authentication process and try to brute force the password for a directory account. Brutus works with HTTP (Basic Authentication) and HTTP (HTLM Form/CGI) authentication types. Still, the attack process could be long since Netware allows to define a number of tries to authenticate before an account locks for a determine amount of time. Therefore, an attacker might lock an account while brute forcing its password with Brutus.

There were more than 9000 alerts by external hosts trying to communicate to port 524/TCP. It was not determined if the NDS protocol (524/TCP) was publicly available from the Internet. Such availability could mean an opportunity for an attacker to probe a Netware network in the same way as a Windows network is probed through the SMB protocol. There are many tools to query and administer a Netware directory, such as NDS Snoop [NOV02] and Visual Click Software's DSRazor but they all require an account.

4.1.6 **Correlations:** There are many SANS practicals reporting on the "MY.NET.30.3 activity" and "MY.NET.30.4 activity" alerts, available at www.giac.org. We specifically analyzed and compared the information presented by Eric Montcalm [MON01] and Vilaiporn Taweelappontong [TAW01].

Incidents.org shows little activity regarding port 524/TCP for the period between March 7 and 10, 2004. Approximately a month later, the activity started to grow and has been above the levels seen in March 2004 ever since. There is even less activity reported for port 51443/TCP during the same period. This is a good example as to why every network is distinct from any other or could also mean that the IDS should be tuned.

No CVE entry associated with this Netware activity was found. Some of the alerts showed that destination port was 6129/TCP. CAN-2003-1030 mentions about a buffer overflow in DameWare, a remote administration

software, that uses the port. DameWare only works on Windows systems so this server could not be vulnerable.

4.1.7 **Evidence of active targeting:** The logs show mostly evidence of active targeting and some general scans.

4.1.8 **Severity:**
(5 + 4) – (3 + 4) = 2

Target Criticality: 5. The systems targeted are administration tools to remotely control resources on a network directory.

Attack Lethality: 4. If this attack is successful, an attacker would be able to login in as an authorized user of the directory and have access to the resources available to the user.

System Countermeasures: 3. User must login prior to accessing resources. Identification of the server is done with digital certificates. Start page on web server lets attacker find out operating system and possible services available.

Network Countermeasures: 4. There is no evidence that traffic is allowed through port 524/TCP. Also, communication channel is encrypted (SSL).

4.2 Detect No. 2: [UMBC NIDS] External MiMail alert

4.2.1 **Description of detect:** W32/MiMail@MM is a worm that attacks windows machines for end users, spreading by email and looking to infect the computers. This is the behavior of a mass-mailing worm. MiMail tries to harvest any email address found in the computer and sends itself to those addresses, inside an infected file attached to the mesage. The worm tries to contact the destination address domain SMTP server to send the infected message. If no DNS server is found, to solve for the SMTP server name, it might use the 212.5.86.163 address and send mails through the list.ru server.

4.2.2 **Reason this detect was selected:** The university presents an interesting mail infrastructure with many message transfer agents (MTA), so this type of threats must be a serious issue to them. A worm loose on their network could mean thousand of machines infected in a short amount of time.

Another reason is that the MiMail worm has more than fifteen variants so I wanted to know how could such situation be handled with the help of an IDS. Finally, the alert message mentions that it is an inbound traffic so it is detecting external hosts that might be infected. It is interesting to see

such configuration in an environment where some people might argue that you have full hands just to manage the network's hosts.

4.2.3 **Detect was generated by:** All detects were generated by the Snort intrusion detection system, using a rule probably created internally by people responsible for the configuration or administration of the IDS. Snort IDS. It is not clear as to what the rule includes, except for the source and destination addresses and ports:

- `alert tcp $EXTERNAL_NET any -> MY.NET.12.6 25 (msg: "[UMBC NIDS] External MiMail alert";)`

We believe the signature must be inspecting the content of the packets or it would have generated many more alerts. The destination address is (very likely) a SMTP server so definitely it is going to receive many packets to port 25/TCP. It is difficult to predict the content defined by the IDS administrators to inspect for since the MiMail worm has more than fifteen variants. The worm also resembles similar features to other worms such as MyDoom and Downloader-CY.

Appendix B has a list of rules found on www.bleedingsnort.com that shows possible content definitions.

4.2.4 **Probability the source address was spoofed:** Low. Logs show established connections between an internal and external address. Also, the amount of alerts logged suggests that the IDS signature is also inspecting for some content in the payload of the packets. If the signature had been broader, like not defining a content to search for, a lot more alerts would have been generated. Such behavior suggests that the IDS is detecting possible attacks that require a connection to the victim host.

4.2.5 **Attack mechanism:** The attack is successful after the worm finds an end-user windows host that downloads the infected message and has vulnerable versions of Microsoft Internet Explorer and Outlook Express. Such vulnerabilities are the MHTML [MSB14] and the codebase exploits [MSB15] reported by Microsoft.

The message includes a compressed zip file that contains an HTML file. The HTML file creates another file, an executable one, which finish the installation of the worm by creating registry keys and files in the operating system directory.

The worm collects email addresses by searching through files stored in the computer, saving them to a file located in the TEMP directory. The worm also captures text from specific windows and sends it by email.

Explain how works the MiMail alert… looks more like a file transfer with a P2P software. One machine on Japan (Yahoo Broadband ADSL) and the other one part of a computer laboratory (ecs021pc25.ucslab.umbc.edu)

4.2.6 **Correlations:**
Our findings and IDS rule theories were compared against the practical assignments of the following GCIA: David Perez Cónde, Billy Smity and Glenn Larratt. Information about the worm was also confirmed at the Symantec Security Response site [SYM04] and the McAfee Virus Information Library site [MCA05].

4.2.7 **Evidence of active targeting:** The logs show evidence of active targeting.

4.2.8 **Severity:** (3 + 2) – (1 + 2) = 2

Target Criticality: 3. It attacks end-user hosts so primary impact would be down time, no critical information should be stored on those hosts. Still, this scenario might impact more important resources so should not be underestimated.

Attack Lethality: 2. Successful attack is more an annoyance to administrators than a real compromise of valuable assets. The information stole from the hosts are mainly email addresses. A negative impact would be the unnecessary consumption of bandwidth, which might affect other network services, if a considerable amount of hosts are infected.

System Countermeasures: 1. No evidence of anti-virus mechanisms installed on end-user hosts or at the mail gateway.

Network Countermeasures: 2. An IDS is detecting external worm attempts to enter the network. There might be anti-virus mechanisms in place but we have no evidence of such.

4.3 Detect No. 3: IRC Alerts

4.3.1 **Description of detect:** There are seven alerts included in this detect from the logs analyzed, all related to the Internet Relay Chat (IRC):

- [UMBC NIDS IRC Alert] XDCC client detected attempting to IRC
- [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.
- [UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.
- [UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting

to IRC
- [UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot
- [UMBC NIDS IRC Alert] K\:line'd user detected, possible trojan.
- IRC evil – running XDCC

IRC is a multi-user chat system. Using channels, people can talk, meet and share information. IRC works in a client/server architecture, people who want to talk together must use a client to connect to a server on the same network and choose the same channel.

A bot is generally an automated client and allows a person to keep control of a channel, thus becoming the operator. Such person can determine who has access to the channel. A bot can also be developed to accept commands from the operator, so the host where it resides could also be controlled or used by the operator.

Usage of IRC usually means problems to college networks. It can be used to launch denial of service (DoS) attacks or to download copyrighted material, while keeping the author(s) anonymously. Both activities are illegal. Attackers are attracted to the high bandwidth available and the less restricted controls usually implemented at those networks, since they are designed to support universities operations and foster research among teachers and students. Unfortunately, attackers are always looking for new ways to disrupt into such networks and misuse their resources.

The seven alerts can be divided into two groups: those that refer to XDCC in the messages from the ones that don't. XDCC refers to a file transfer system, pretty much in the same way as peer to peer (P2P) file systems such as Kazaa or Gnutella, but using the Direct Client-to-Client (DCC) protocol over IRC. There is another big difference, XDCC helps to keep its users anonymous. By compromising computers on the Internet and installing bots, it might become difficult to trace back the origin of the IRC sessions.

The “sdbot floodnet” alerts probably refer to a popular bot used to launch DoS attacks. For the last two years people have found many variants of this bot.

4.3.2 **Reason this detect was selected:** To understand the inner working details of IRC and the XDCC bot. The risks that transferring copyrighted material or creating a denial of service web of handlers and daemons by using college's hosts could produce to the university are worth the effort. IRC, including the XDCC bot, can be used for such purposes.

The fact that university's IDS administrators have edited the alert messages to point out IRC activity from the rest of the detects also is a sign of the importance given or risks associated to such service.

4.3.3 **Detect was generated by:** All detects were generated by the Snort intrusion detection system. The alerts were generated by signatures similar to the ones found at [PER01]:

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

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

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

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

- alert tcp $EXTERNAL_NET 6660:7000 -> $HOME_NET any\
(content: " 324 "; offset:5;\
content: "xdcc";\
msg: "User joining XDCC channel detected. Possible XDCC bot";\
classtype:misc-activity;)

- alert tcp $EXTERNAL_NET 6660:7000 -> $HOME_NET any\
(content: " 465 "; \
msg: "K\:line'd user detected, possible trojan.";\
classtype:misc-activity;)

- alert tcp $HOME_NET any -> $EXTERNAL_NET 6660:7000\
(content: “xdccbot”; nocase; hmsg: “IRC evil – running XDCC”;\
classtype:misc-activity;)

From the six signatures above, one can determine that ports 6660/TCP to 7000/TCP are commonly used between IRC servers and bots. 6665 to 6669/TCP are registered ports for IRC according to IANA. All six signatures also look for specific content inside the payload of the packet, searching for control information or strings that identify the IRC traffic.

The IRC packets' payload is analyzed using the “content” option and two modifiers: offset and nocase. The content option is used to detect one or

more character strings or byte values. The offset modifier defines a pointer from the beginning of the payload to start the search of content. The nocase allows detecting the strings or values regardless of the case sensitivity.

4.3.4 **Probability the source address was spoofed:** Low. Some alert logs show established connections between internal and external addresses: MY.NET.27.103 <> 209.126.201.99 and MY.NET.15.198 <> 64.157.246.22. Also, based on the rules signatures, Snort is looking for TCP packets with a payload usually found in two-way communication negotiations (for example, error and authentication messages).

4.3.5 **Attack mechanism:** How the IRC clients and bots are installed on the MY.NET. hosts have two possible explanations. The first and obvious one is an existing user who wants to install the IRC client or bot and has the rights necessary to do it. This is a probable situation since college students use chat channels and download copyright material. The second option involves system vulnerabilities and is explained below.

The installation of a bot (XDCC or Sdbot) could be the result of an attacker taken advantage of the misconfiguration or lack of security patch in a host. A search on the UNISOG mailing list, SANS hosted list intended for educational organizations and especially universities detected a thread between February and April of 2002 where several infosec specialists reported on hosts compromised that had the XDCC bot installed.

The first step for the system to get the XDCC bot installed was the lack of a strong password for an account with administrative privileges on a windows host, as reported by several administrators. Tools like X-Scan [XFO01] and Fluxay [FLU01] allow enumerating accounts and performing dictionary attacks on them and were found on compromised machines. The attack wasn't limited to the administrator's account since the hosts most likely answered to null session enumeration, allowing the attacker to determine other accounts with similar rights.

Once an administrative account was found, an attacker could have downloaded software to the compromised host by using tools such as Sysinternals' psexec [SYS01]. This would permit the attacker to install any bot, like GTbot or XDCC. After this, the bot would connect automatically to a predefined IRC channel and install or run any other software determined by the attacker.

An analysis from Allen Chang of Berkeley University determined that a telnet server is started on port 132/TCP, the backdoor client listens on 8888/TCP and an FTP server listens on 43958/TCP and 3112/TCP. The

FTP server file is named lsass.exe just like the real Windows service. Lsass.exe is CatSoft's Serve-U FTP server, installed as a service using the firedaemon.exe utility [FIR01]. The ir.con configuration file shows that the IRC bot is set up as an XDCC file serving bot.

Several files are copied to the All Users' Startup directory to guarantee that services will start once the hosts are rebooting. Finally, a hidden directory is created in C:\RECYCLYER as the upload dir for the XDCC bot.

4.3.6 **Correlations:** An excellent report on details of the X-DCC IRC bot is available from Dave Dittrich [DIT01] of Washington University. Credit also goes to Christopher Cramer of Duke University, who started the thread on the SANS Unisog mailing list. Full thread is available at [DIT02]. Other good sources for XDDCC are [TON01, WIK01].

David Love's GCIA Practical Assignment was compared and although is for a different set of logs, agrees on the general analysis to the detects related to IRC.

A reference to the sdbot floodnet can be found at [NAI01, CAP01].

There are several references to the CVE, based on the mechanisms used by attackers to compromised hosts and install the XDCC bot. As part of the XDCC suite of tools, X-Scan looks for easy to guess or null passwords and open shares on windows hosts:

- CAN-1999-0503: A Windows NT local user or administrator account has a guessable password.
- CAN-1999-0504: A Windows NT domain user or administrator account has a default, null, blank, or missing password.
- CAN-1999-0505: A Windows NT local user or administrator account has a guessable password.
- CAN-1999-0506: A Windows NT domain user or administrator account has a default, null, blank, or missing password.
- CAN-1999-0518: A Netbios/SMB share password is guessable.
- CAN-1999-0519: A Netbios/SMB share password is the default, null, or missing.
- CVE-2003-1200: Windows NT allows remote attackers to list all users in a domain by obtaining the domain SID.

4.3.7 **Evidence of active targeting:** The logs show evidence of active targeting.

4.3.8 **Severity:**
(3 + 4) – (1 + 2) = 4

Target Criticality: 3. According to DNS queries, most of the hosts involved could be end-user or computer lab hosts where there is no critical information stored. Still, the amount of internal hosts to be found was considered.

A couple of hosts named solaris1.gl and tfc-pplant should be further analyzed to determine if more valued resources could be compromised.

Attack Lethality: 4. Network hosts could be used to further compromise other systems, transfer copyrighted material or be part of a DoS attack. Legal vulnerabilities should be taken seriously since several organizations are trying to put an end to copyright infringements.

System Countermeasures: 1. Several internal hosts are source of alerts detected, meaning that a bot or client could have been installed. Hosts lack hardening configurations or may be administrative users are installing unauthorized software.

Network Countermeasures: 2. An ID is detecting IRC requests, but traffic on ports 6660/TCP to 7000/TCP could be allowed in both directions.

5. Network Statistics

5.1 Top Five Talkers

Below are the five noisiest hosts according to the traffic they generated and regardless of their location. After analyzing the log files with SnortSnarf, these are the top five talkers:

SOURCE IP	NO. OF ALERTS	NO. OF SIGNATURES	DESTINATIONS INVOLVED
MY.NET.27.103	5473	2 Signatures	169.254.45.176, 209.126.201.99
67.20.160.15	4405	1 signature	MY.NET.30.4
68.49.76.164	1348	2 Signatures	MY.NET.30.4, MY.NET.30.3
216.56.88.95	1311	2 Signatures	MY.NET.30.4, MY.NET.30.3
68.55.156.128	1173	2 Signatures	MY.NET.30.4, MY.NET.30.3

This table includes the information from the alert and OOS files. The noisiest host is MY.NET.27.103, which should be revised more in terms of worms or signs of being compromised. I am suspicious about false positives for the other four hosts since they only talked to MY.NET.30.4 and MY.NET.30.3 hosts. As was determined previously, these hosts offer remote access to network services. All whois lookups determined

that these hosts could be located in USA and some could even be not very far from the University's campus premises.

If the scan files were considered, all five top talkers would have been portscan sources: MY.NET.110.72, MY.NET.97.74, MY.NET.82.87, MY.NET.153.92 and MY.NET.112.151. These hosts performed SYN scans and generated lots of UDP packets.

5.2 **Top Five Targeted Services or Ports:**

The criteria to choose these services was based but not limited on the amount of alerts related to each port. The IDS needs to be tuned because there are many possible cases of false positives, where alerts are triggered by normal and not dangerous traffic. The following services are chosen as the top targeted but are presented in no particular order:

5.2.1 Simple Mail Transfer Protocol (SMTP, 25/TCP): Several types of alerts for this service. The university has many mail servers, they all received OOS packets. Also, external hosts used this port to perform NMAP TCP ping on MY.NET. hosts.

5.2.2 Internet Relay Chat (IRC, 6660 to 7000/TCP): Many hosts from the MY.NET. network were most probable compromised and had IRC bots installed such as XDCC and Sdbot. The bots allowed these hosts to be used to transfer big files and participate as members of a denial of service web.

5.2.3 Netware Core Protocol (NCP, 524/TCP): Although we did not have enough information to determine if traffic to this port was authorized or not, it is the language of the directory service. Since the service could allow controlling many resources in the network, should be a high-targeted port.

5.2.4 Netware NetStorage Web Access (51443, TCP): Same situation as NCP. The NetStorage File Management system allows a remote user to access files and directories located on the inside of the University's network. Default web pages on servers allowed attackers to easily identify this service.

5.2.5 Hyper Text Transfer Protocol (HTTP, 80/TCP): Many alerts detected on what is a very common and expected member of any network: shellcode attempts, NMAP TCP pings, and remote execution of system commands. The Novell Directory servers had web service enabled and many alerts were detected. Unfortunately IDS needs to be tuned to discard false positives from real attacks attempts.

5.3 **Top Three Most Suspicious External Source Addresses:**

5.3.1 **213.157.171.109:** Between March 9 and 10, SYN scanned about a third of the Class-B MY.NET. network, potentially detecting live hosts. A DNS query provides the 213-157-171-109.brasov.rdsnet.ro hostname. The registration information for this IP address is:

```
inetnum:        213.157.171.0 - 213.157.171.255
netname:        RDSNET
descr:          Romania Data Systems
country:        RO
admin-c:        RDS-RIPE
tech-c:         RDS-RIPE
status:         ASSIGNED PA
notify:         notify-ripe@rdsnet.ro
mnt-by:         AS8708-MNT
changed:        tim@rdsnet.ro 20011109
source:         RIPE

role:           Romania Data Systems NOC
address:        71-75 Dr. Staicovici
address:        Bucharest / ROMANIA
phone:          +40 21 30 10 888
fax-no:         +40 21 30 10 892
e-mail:         contact-tech@rdsnet.ro
admin-c:        AS1385-RIPE
tech-c:         BCD-RIPE
tech-c:         MIHV1-RIPE
tech-c:         GEPU1-RIPE
nic-hdl:        RDS-RIPE
```

5.3.2 **220.37.240.35:** For fifteen minutes on March 8, there was a established session with MY.NET.53.55 host, using port 65535/TCP. Worms commonly use this port. A DNS query provides the YahooBB220037240035.bbtec.net hostname. The registration information for this IP address is:

```
inetnum:        220.0.0.0 - 220.63.255.255
netname:        BBTECH
descr:          Japan nation-wide Network of SOFTBANK BB CORP
descr:          Tokyo, Japan
country:        JP
admin-c:        SA127-AP
tech-c:         SA127-AP
mnt-by:         APNIC-HM
mnt-lower:      MAINT-JP-BBTECH
changed:        hostmaster@apnic.net 20020412
changed:        hm-changed@apnic.net 20030616
status:         ALLOCATED PORTABLE
source:         APNIC

role:           SoftbankBB ABUSE
address:        24-1, Nihonbashi Hakozaki-Cho ,Chuo-Ku ,Tokyo
country:        JP
phone:          +81-0570-919-820
e-mail:         hostmaster@bbtec.net
trouble:        Please send spam report,virus alart
trouble:        or any other abuse report
trouble:        to  abuse@bbtec.net
trouble:        Any other Information, Notice,
trouble:        Please send to hostmaster@bbtec.net
admin-c:        TT123-AP
tech-c:         ST222-AP
```

```
nic-hdl:        SA127-AP
notify:           admin@bbtec.net
mnt-by:           MAINT-JP-BBTECH
changed:          stsuruma@softbank.co.jp 20030613
source:           APNIC
```

5.1.3 **209.126.201.99:** On March 7, MY.NET.27.103 attempted to IRC to this host. The following day another host, MY.NET.80.15, did the same. A common XDCC bot might have been configured to connect to this host. A DNS query provides the desire.of.hotgirlz.org hostname. The registration information for this IP address is:

```
OrgName:     California Regional Internet, Inc.
OrgID:       CALI
Address:     8929A COMPLEX DRIVE
City:        SAN DIEGO
StateProv:   CA
PostalCode:  92123
Country:     US

NetRange:    209.126.128.0 - 209.126.255.255
CIDR:        209.126.128.0/17
NetName:     CARI
NetHandle:   NET-209-126-128-0-1
Parent:      NET-209-0-0-0-0
NetType:     Direct Allocation
NameServer:  NS1.ASPADMIN.COM
NameServer:  NS2.ASPADMIN.COM
Comment:     ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:     1999-03-12
Updated:     2003-07-01
```

6. Correlations from Previous Students Practicals

The GIAC site proved to be a very valuable source of information, comments and opinions from previously certified analysts. Of particular interest and help to correlate detects were the following GCIA Practical Assignments:

- Ben Allen
- Billy Smith
- Chris Baker
- Chris Kuethe
- Dana McLaughlin
- David Love
- Diego Gonzalez
- Don Murdoch
- Donald Merchant
- Edward Peck
- Glenn Larratt
- John Hally
- Kam Hung Ng
- Matthew Richard
- Ricky Smith
- Tim Kroeger
- Vilaiporn Taweelappontong

7. Insights into Internal Machines that Might Be Compromised or Generating

Dangerous or Anomalous Activity

Several hosts could have been compromised or could be part of malicious, non-authorized activity:

7.1 MY.NET.98.16, MY.NET.97.10 and MY.NET.97.33 could have the IRC sdbot installed. The IDS detected traffic between these hosts and several external IRC servers, such as irc.belwue.de (129.143.67.242) or webmaster.ca.us.austnet.org (216.152.64.155).

7.2 Several internal hosts could be part of an IRC XDCC network. MY.NET.27.103, MY.NET.112.199, MY.NET.15.198 and MY.NET.80.15, attempted to connect to IRC channel and seemed to have an XDCC client installed. MY.NET.42.0 network hosts (1, 3, 4, 11 and 12), MY.NET.11.199 and MY.NET.82.79 reported to an IRC channel as XDCC bot. Their targets are also found as source hosts for "XDCC Send Request Detected".

7.3 MY.NET.97.95, MY.NET.97.76 and MY.NET.60.39 hosts should be further investigated. They all attempted to connect to port 515/TCP of external hosts.

7.4 MY.NET.84.235 could be a Denial of Service handler. Two alerts involved this host: DDOS mstream client to handler and DDOS shaft client to handler.

7.5 MY.NET.53.55, MY.NET.53.31, MY.NET.42.1, MY.NET.42.5, MY.NET.42.6, MY.NET.29.3, MY.NET.6.63, MY.NET.42.8 and MY.NET.112.152 should be inspected because they could have the Red Worm. Their activity involved sending packets to hosts on port 65535/TCP. Same diagnostic for MY.NET.53.55, MY.NET.112.151, MY.NET.6.62 and MY.NET.69.214 because of 65535/UDP traffic generated.

7.6 MY.NET.70.225, MY.NET.70.156, MY.NET.42.2, MY.NET.70.210, MY.NET.150.244 and MY.NET.111.46 could be running a non-authorized VNC server.

7.7 A few hosts could also be examined, although the suspicious activity logged is not enough to determine if they have been compromised or infected. Host MY.NET.112.226 sent a couple of packets with null ports and MY.NET.17.45 sent one packet, attempting to execute the cmd.exe on external host.

7.8 There are many hosts that should be examined to determine source of portscans detected. These hosts repeatedly send packets to big address spaces (in some cases, several class-B networks), send packets in short amount of time (hundreds in seconds) or use strange ports: MY.NET.110.72, MY.NET.97.74, MY.NET.82.87, MY.NET.153.92, MY.NET.112.151, MY.NET.80.224, MY.NET.82.15, MY.NET.112.216, and MY.NET.98.11. A general examination to the entire MY.NET.97.0/24 network might be worthwhile.

7.9 Finally, all source hosts for "SMB Name Wildcard" detect should be examined to

determine if some Windows worm infected them or are scanning for open shares or Windows hosts on the Internet. This could be a false positive because SMB ports don't seem to be blocked at firewall so there could be legitimate reasons for this traffic. Network users could be accessing external hosts with SMB protocol.

8. Defensive Recommendations

8.1 Configure the portscan preprocessor on Snort so the false positives can be reduced, possibly ignoring hosts like servers that create many connections in a short period of time. For example, the following line could be added to the snort configuration file to exclude traffic from MY.NET.30.3 and 4 hosts:

preprocessor portscan-ignorehosts: MY.NET.30.3/32 MY.NET.30.4/32

8.2 Also, the preprocessor time period should be tuned, to avoid generating false positives. For example, the threshold is currently at 12 connections in at least 787 seconds and could be changed to 12 connections in 60 seconds. The number of connections could also be used to tune the preprocessor.

8.3 Make sure the system where snort is running to be secure using a protocol like SSH for remote access, closing all unnecesary ports and keeping system patched. Snort sensor should also query an NTP server to keep time synchronize, specially among different logfiles to allow correlation.

8.4 Tune the Snort sensor so detects real attacks or attempts to MY.NET.30.3 and MY.NET.30.4 servers. At this time, looks like the signatures are too broad and detect anything, creating too many false positives. Also consider removing or at least tuning the SMB Name Wildcard signature.

8.5 Install and run some intrusion (logs) analysis tool frequently for better management of logs. The analysis tool should produce results at least on screen in (almost) real time. This network is very active in terms of possible malicious activity and its administrators need to be proactive about its protection.

8.6 Lockdown end-user hosts, install anti-virus software and update it periodically (for example daily). When situation is controlled and the majority of hosts have security measures implemented, tune again the IDS sensor. For example, trying to detect internal portscan might be too noisy. You can choose to disable such detection if several other measures are implemented.

8.7 Collect snort packets with link layer and application layer data. Producing the oos files from running snort in sniffing mode provides only header information for each packet that doesn't allow the administrator to fully determine risk.
Also, dates don't match the files names. 040307 has 03/11 packets.

8.8 Consider use of tagging feature in Snort to help determine possible real attacks versus false positives. Tagging is available on Snort version 1.8 or later.

8.9 Priority should be defined for each rule configured on Snort. This will help the administrators to quick determine if an alert should be handled immediately or can wait for later.

8.9 Consider the implementation of HTTP-based services alerts. There are many important servers that provide access to university's network and its resources through these services. They are a popular target for attackers so consider having a limited number of alerts to assess the threats and risks.

Part III – Analysis Process

The log files were analyzed on a DELL Inspiron 8600 laptop, with 1GB RAM and running Fedora Core R1. An additional external 120GB Hard Disk Drive (USB v2.0) was used to store the logs, for portability reasons.

Three tools were used to deal with the logs and produce analysis reports: SnortSnarf, the Snort_Stat perl script and the stream editor, sed. Each one contributed to understand and analyze the huge amount of logs for each day selected. It is not feasible to manually analyze the logs since they were over 200MB.

SnortSnarf allowed me to do a lot of correlation, between addresses and alerts. This tool produce HTML pages where you could find all alerts related to an specific address, acting as source or destination. Also useful, was the whois and DNS lookup links that where automatically generated for each host.

The log files had to be formatted and reduced in size for SnortSnarf to work. While testing with the tool, a non-edited alert file took more than four hours to process with SnortSnarf. This probed useless as the report only contained one day of alerts. To solve this problem I decided to remove the portscan preprocessor alerts, as was recommended by [SMT01]. The eight alert and OOS files took about a minute for SnortSnarf to crunch after the portscan alerts where removed. Also, the OOS logs were edited with a perl script provided by [SMT01].

Although SnortSnarf is a must use tool, it has its limitations. For example, it cannot answer you questions involving several alerts or detects. Also, if you want to determine ports probed or search for specific hosts between several detects, you might want to try another tool. This is where I decided to use the stream editor, sed. While SnortSnarf provided a good overall picture, sed allowed me to find answer to specific questions.

Sed also help me prepare the alert files for SnortSnarf. The alert files had been sanitized to protect university identity but SnortSnarf could not process this. With a simple sed command I was able to fix this and some missing new line characters:

```
sed –e 's/03\//\n03\//2' –e 's/MY.NET./224.244./g' alert_file >
                          output_file
```

Sed is a tool very simple to use. After a couple of hours I was able to come up with several alternatives to answer my questions. For example, to remove the spp_portscan lines from the alert files and prepare them for SnortSnarf:

```
sed '/^03.*spp_portscan:.*$/d'
```

To determine the destination ports of one detect, arrange in numeric order:

```
cat <source file> | grep "<unique string>" | sed
's/^03.*\:.*\://' | sort | uniq -c | sort –rn
```

To determine source hosts for one detect, arranged in numeric order:

```
cat <source file> | grep "<unique string>" | sed 's/^03.*].*] //'
| sed 's/\:.*$//' | sort | uniq -c
```

To determine destination hosts for one detect, arranged in numeric order:

```
cat <source file> | grep "<unique string>" |  sed 's/^03.*].*]
//' | sed 's/\:.*$//' | sort | uniq -c | sort -rn
```

To determine destination ports for one detect, arrange in numeric order:

```
cat <source file> | grep "<unique string>" | sed 's/^03.*->.*://'
| sort | uniq -c | sort -rn
```

To determine destination host and port for a detect, arrange in numeric order:

```
cat <source file> | grep "<unique string>" | sed 's/^03.*-> //' |
sort | uniq -c | sort -rn
```

To determine portscan sources and the number of times it was detected:

```
cat <source files> | sed ''s/ -> .*$//' -e 's/^Mar .*:[0-9]*:[0-
9]* //' -e 's/:.*$//' | sort | uniq -c | sort -rn
```

The third tool used was Snort_Stat, available from the Snort website. This tool allowed me to determine the distribution (percentage) of attack methods and more importantly to confirm the results I previously got from SnortSnarf and Sed. With a third tool showing the same numbers for each alert, I was more confident that was using the tools correctly.

Another important aspect to understand the logs, was to determine or have a good idea of the rules involved. The Snort Rules Database [SNO05] and the Whitehats ArachNIDS archive [ARC05] are an excellent source for information on rules. A similar source is the SANS Intrusion Detection Forum. I found answers to my question the files types, good references, reviews by peers, value of snort rules, and even opinions about tools you might want to use.

No port scans were performed for the preparation of this report. Web browsing and DNS single queries were done to determine possible host usage thru its name or to confirm if a web server was running. Problem with this method was that an IP address might have been reassigned between the time the logs were created and I did the queries. Still, I decided to accept this risk in order to be able to formulate more detailed theories.

Bibliography

[ARC05] Advanced Reference Archive of Current Heuristics for Network Intrusion Detection Systems (ARACHNIDS). Last accessed March 4, 2005 at http://www.whitehats.com/ids.

[BRU01] Hoobie. Brutus, remote password cracker. Retrieved January 3, 2005 from http://www.hoobie.net/brutus.

[CAP01] Computer Associates. Etrust PestPatrol Pest Encyclopedia: Backdoor/SDBot. Retrieved January 13, 2005 from http://www.pestpatrol.com/pestinfo/b/backdoor_sdbot.asp

[DIT01] Dittrich, D. (2002,May 3). World-wide distributed DoS and "warez" bot networks. Retrieved January 15, 2005 from http://staff.washington.edu/dittrich/talks/core02/xdcc-analysis.txt

[DIT02] Cramer, C. et al. (2002). Several threads regarding the X-DCC IRC bot. Retrieved January 15, 2005 from http://staff.washington.edu/dittrich/talks/core02/unisog-xdcc.txt

[FLU01] Anonymous. Fluxay windows scanner. Retrieved January 4, 2005 from http://www.netxeyes.com/down.html

[FIR01] Sublime Solutions. Firedaemon v1.7 GA (Build 2123). Retrieved January 2, 2005 from http://www.firedaemon.com

[ISI81] Information Sciences Institute. (1981). RFC793: Transmission Control Protocol, Darpa Internet Program Protocol Specification. Retrieved December 30, 2004, from http://www.isi.edu/in-notes/rfc793.txt

[NOV01] Novell Corporation. (2002). Netware 6 Documentation. Retrieved January 5, 2005 from http://www.novell.com/documentation/nw6p/index.html.

[NOV02] Bunnell, K. (2002). NDS Snoop, An NDS Prototyping Tool. Retrieved January 5, 2005 from http://www.novell.com/coolsolutions/feature/5338.html.

[MCA05] Network Associates Technologies, Inc. McAfee Virus Information Library: W32/MiMail@MM. Retrieved January 5, 2005 from http://vil.nai.com/vil/content/v_100523.htm.

[MON01] Montcalm, E. GCIA Practical Assignment. Retrieved December 28, 2004 from http://www.giac.org/practical/GCIA/Erik_Montcalm_GCIA.pdf

[MSB14] Microsoft Corporation. Microsoft Security Bulletin MS03-014: Cumulative Patch for Outlook Express (330994). Retrieved January 5, 2005 from http://www.microsoft.com/technet/security/Bulletin/MS03-014.mspx

[MSB15] Microsoft Corporation. Microsoft Security Bulletin MS02-015: 28 March 2002 Cumulative Patch for Internet Explorer. Retrieved January 5, 2005 from http://www.microsoft.com/technet/security/Bulletin/MS02-015.mspx

[NAI01] Network Associates. Virus Information Library: IRC-Sdbot. Retrieved January 13, 2005 from http://vil.nai.com/vil/content/v_99410.htm

[PER01] Snort IRC Rules. Retrieved January 15, 2005 from http://coders.meta.net.nz/perry/irc.rules/

[SMT01] Smith, R.D. GCIA Practical Assignment. Retrieved December 23, 2004 from http://www.giac.org/practical/GCIA/Ricky_Smith_GCIA.pdf

[SNO05] Snort Rules Website. Retrieved March 9, 2005 from http://www.snort.org/rules/

[SYM04] Symantec Corporation. Security Response: W32.Mimail.A@mm Information. Retrieved January 4, 2005 from http://securityresponse.symantec.com/avcenter/venc/data/w32.mimail.a@mm.html

[SYS01] Sysinternals. PsTools v2.11. Retrieved January 4, 2005 from http://www.sysinternals.com/ntw2k/freeware/pstools.shtml

[TON01] TonikGin. (2002, April 11). XDCC, An .EDU Admin's Nightmare. Retrieved December 15, 2004 from http://www.cs.rochester.edu/~bukys/host/tonikgin/EduHacking.html

[WIK01] WikiMedia. (2005). Wikipedia, The Free Encyclopedia: XDCC. Retrieved January 12, 2005 from http://en.wikipedia.org/wiki/XDCC

[XFO01] Xfocus Team. X-Scanv3.1-en.rar. Retrieved January 4, 2005 from http://www.xfocus.org/programs/200407/16.html

References

Baker, C. (2001). GCIA Practical Assignment. Retrieved December 31, 2004, from http://www.giac.org/practical/Chris_Baker_GCIA.zip

Beale, J., Baker, A., Caswell, B., & Poor, M. (2004). Snort 2.1. (2nd ed.). Rockland: Syngress Publishing.

Chuvakin, A. (2004, October 21). Five Mistakes of Log Analysis. ComputerWorld. Retrieved December 30, 2004, from http://www.computerworld.com/securitytopics/security/story/0,10801,96587,00.html?SKC=security-96587

Cooper, M., Northcutt, S., Fearnow, M., & Frederick, K. (2001). Intrusion Signatures and Analysis. Indianapolis: New Riders.

Gilly, D., & staff of O'Reilly & Associates, Inc. (1994). UNIX in a Nutshell, System V Edition (2nd ed.). Cambridge: O'Reilly & Associates.

Kuethe, C. (2001). GCIA Practical Assignment. Retrieved December 31, 2004, from http://www.giac.org/practical/chris_kuethe_gcia.html

McLaughlin, D. (2000). GCIA Practical Assignment. Retrieved December 31, 2004, from http://www.giac.org/practical/dana_mclaughlin_gcia.doc

Northcutt, S., & Novak, J. (2002). Network Intrusion Detection: An Analyst's Handbook (3rd ed.). Indianapolis: Sams Publishing.

Phillips, W. (2002). GCIA Practical Assignment. Retrieved December 31, 2004, from http://www.giac.org/practical/Bill_Phillips_GCIA.zip

Randier, S. (2003). GCIA Practical Assignment. Retrieved March 5, 2005, from http://www.giac.org/certified_professionals/practicals/gcia/0620.php

Roesch, M., & Poor, M. (2004). Intrusion Detection: Snort Style. The SANS Institute.

Stevens, W. R. (2002). TCP/IP Illustrated, Volume 1: The Protocols (21st ed.). Boston: Addison-Wesley.

Veeraraghavan, S. (1999). Teach Yourself Shell Programming in 24 Hours. Indianapolis: Sams Publishing.

Appendix A

The table below provides a summary of the activity detected with the Snort IDS located at the SANS University network. The logs were generated between March 7 and March 10, 2004. The table includes the amount (daily and total) of packets that triggered an alert, the message and description of such alert and the possible source of the rule.

S -> D means "Source to Destination" and can have at least one of two values:

- Int -> Ext: University or internal hosts to Non-University or external hosts and,
- Ext -> Int: Non-University or external hosts to University or internal hosts.

The Rule Source column has four possible choices:

- In house: Means that the Snort rule was created locally,
- Snort xxx: Refers to a value assigned to an specific Snort IDS rule. Value (xxx) should be between 100 and 1,000,000
- Arachnids yyy: Refers to a value assigned to an IDS rule on the Whitehats' ArachNIDS archive system.
- SPP: Means the alert was generated using a Snort PreProcessor.

S -> D	Alert	03/07/04	03/08/04	03/09/04	03/10/04	Total	Description	Rule Source
Ext -> Int	[UMBC NIDS IRC Alert] IRC user / kill detected possible trojan	122	61	17	68	268	Source ports are between 6660/TCP and 7000/TCP. Mainly targeted to MY.NET.27.103, while other MY.NET. hosts are targeted. Detects a kill command sent to a MY.NET. Host that is connected to an IRC channel	In house
Ext -> Int	[UMBC NIDS IRC Alert] K\:line'd user detected, possible trojan	0	0	1	0	1	From host 81.174.249.138:6881 to MY.NET.82.109:1361, the packet contained an instruction to prevent the internal host from connecting to an IRC server	In house

Ext -> Int	[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected	4	22	16	14	56	Source ports are 6660/TCP to 7000/TCP. Targeted to MY.NET.42.0 network hosts (1,3,4,11, and 12). Shows an XDCC send command by an external host, possibly looking to download files hosted in MY.NET. External source hosts are also found as destination hosts for detect “IRC evil -running XDCC”	In house
Int -> Ext	[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	0	12	0	7	19	MY.NET.98.16, MY.NET.97.10 and MY.NET.97.33 connecting to IRC servers 216.152.64.155 and 129.143.67.242. Destination ports are between 6660/TCP and 6666/TCP.	In house
Ext -> Int	[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	9	2	1	6	18	External hosts with source ports between 6660/TCP and 7000/TCP to MY.NET.42.0 network hosts (2,4,8,11 and 12). Packet could mean the action of a XDCC bot having joined a channel.	In house
Int -> Ext	[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	5469	7	0	60	5536	On March 07/04, MY.NET.27.103 tried to log to server 209.126.201.99. On March 08, MY.NET.112.199 did the same on server 216.10.29.62, while MY.NET.15.198 accessed 64.157.246.22 and MY.NET.80.15 accessed 209.126.201.99 (same server as for MY.NET.27.103). Finally, on March 10 MY.NET.15.198 accessed 64.157.246.22. Destination ports were between 6666/TCP and 7000/TCP.	In house
Ext -> Int	[UMBC NIDS] External MiMail alert	5	7	7	5	24	Multiple external hosts trying to access MY.NET.12.6 on port 25/TCP. Seems like a false positive since hostname is mxin (mail exchange inside).	In house

Ext -> Int	Attempted Sun RPC high port access	0	0	15	2	17	Mainly a port scan from host 142.165.212.10 to subnet MY.NET.70.0. Also attempts from 66.28.227.10 to some MY.NET.190.0 and MY.NET.109.0 network hosts. Destination port is always 32771/TCP. The scan is looking for SunOS hosts that have the portmapper listening on this port. Some firewalls forget to block high ports, allowing an attacker to access the portmapper even when port 111/TCP is blocked.	Snort 599 or Arachnids 26
Ext -> Int	Back Orifice	0	0	1	0	1	From host 142.165.212.10:1345 to MY.NET.190.203:31337, an information request could have been sent to a Back Orifice trojan.	Snort 116
Int -> Ext	Connect to 515 from inside	0	0	7	113	120	Internal hosts MY.NET.97.95, MY.NET.97.76 and MY.NET.60.39 attempt to connect to port 515/TCP of external hosts	In house
Ext -> Int	Connect to 515 from outside	27	0	7	1	35	142.165.212.10 and 82.82.67.46 connects to host MY.NET.6.15 and MY.NET.190.0 network hosts. Destination port is 515/TCP	In house
Ext -> Int	DDOS mstream client to handler	0	2	0	0	2	Client 209.246.126.236 is communicating to handler MY.NET.84.235 (CAN-2000-0138). Destination port is 12754/TCP. A string of “>” is included in the payload of the packet.	Snort 247

Int -> Ext	DDOS shaft client to handler	0	8	0	1	9	From hosts 210.65.0.24:80, 80.178.3.252:4662, 131.220.99.72:80 and 69.0.136.123:80 to MY.NET.84.235:20432. This event is generated when a DDoS Shaft client communicates with a Shaft handler. It is also possible that this event may be generated when any host attempts to discover or detect a Shaft handler. URL: http://security.royans.net/info/posts/bugtraq_ddos3.shtml)	Snort 230
Ext -> Int	EXPLOIT NTPDX buffer overflow	1	2	4	1	8	Buffer overflow attempts at ntpd daemon, sending packets greater than 128 bytes. Could be a port scan. March 8, 9 and 10: Host 66.250.188.23 to MY.NET.66.29. Also, one alert per destinations hosts MY.NET.16.106 and MY.NET.109.86. Destination port is 123/TCP. More info at http://www.digitaltrust.it/arachnids/IDS492/research.html	Snort 312 or Arachnids 492
Ext -> Int	EXPLOIT x86 NOOP	100	89	165	93	447	NOOP is a common string found in buffer overflow attacks. The logs show several well-known destination ports such as 80/TCP, 135/TCP, 6129/TCP, 119/TCP, and 445/TCP among others.	Snort 648 or Snort 1394
Ext -> Int	EXPLOIT x86 setgid 0	3	1	4	4	12	External hosts run code to attempt to change identity of group to root's. Source port is 80/TCP	Snort 649
Ext -> Int	EXPLOIT x86 setuid 0	6	2	5	9	22	Attacker attempts to gain administrative rights by running code to change identity of user to root. Root's setuid is zero. All source hosts are external and only are responsible for one alert.	Snort 650

Ext -> Int	EXPLOIT x86 stealth noop	1	0	0	0	1	From host 206.24.192.253:80 to MY.NET.97.41:3786. Attempting to run buffer overflow to gain access to internal host.	Snort 651
Ext -> Int	External FTP to Helpdesk MY.NET.53.29	1	0	0	0	1	From 131.130.170.146:1053 to 130.85.53.29:21	In house
Ext -> Int	External FTP to Helpdesk MY.NET.70.49	1	5	2	1	4	Three external hosts (134.2.78.155, 24.79.169.51 and 131.130.170.146) attempt to connect to MY.NET.70.49 on port 21/TCP	In house
Int -> Ext	HelpDesk MY.NET.70.49 to External FTP					5	Attempts FTP connection to hosts 205.227.137.57, 216.49.88.143, 63.209.221.236 and 208.184.139.99. At least three out of those servers respond as ftp.nai.com.	In house
Ext -> Int	External FTP to Helpdesk MY.NET.70.50	1	0	1	1	3	Three source hosts: 134.2.78.155, 67.100.216.5 and 131.130.170.146. All from college networks.	In house
Ext -> Int	External RPC call	5	0	8	155	168	MY.NET.190.0 network was heavy scanned for portmap daemon (111/TCP) on March 10. The objective is to gather port information for open RPC services and associated ports.	Snort 598
Ext -> Int	FTP DoS ftpd globbing	6	0	78	0	84	Heavy packet load to host MY.NET.153.79 on March 9 and lightly to host MY.NET.24.27 (ftp1) on March 7.	Arachnids 487
Ext -> Int	FTP passwd attempt	11	12	9	1	33	Attempts to retrieve a password file from MY.NET.24.47 (ftp1). Detected throughout the four days analyzed. Several source hosts.	Arachnids 213

Int -> Ext Ext -> Int	High port 65535 tcp – possible Red Worm – traffic	171	1929	151	1379	3630	Source or destination port is always 65535/TCP. Several internal hosts as source, including: MY.NET.53.55, MY.NET.53.31, MY.NET.24.44, MY.NET.42.1,MY.NET. 42.5 and MY.NET.42.6. Traffic from source port as 65535/TCP could be mail transfers and traffic from source port as 80/TCP could be HTTP. Both look like false positives.	In house
Int -> Ext Ext -> Int	High port 65535 udp – possible Red Worm – traffic	0	5	3	22	30	Source port is always 65535/UDP. Four internal hosts act as source: MY.NET.112.151, MY.NET.6.62, MY.NET.53.55 and MY.NET.69.214	In house
Int -> Ext Ext -> Int	Incomplete Packet Fragments Discarded	23	29	26	9	87	Source and dest ports are zero (0). Two alerts where triggered by MY.NET.112.226 as source host. Fourteen alerts from 218.164.134.94	In house
Int -> Ext	IRC evil – running XDCC	13	77	65	42	197	Hosts from MY.NET.42.0 network reporting as XDCC bot on an IRC channel. Same situation for hosts MY.NET.112.199 and MY.NET.82.79. All destination hosts are found as source hosts for detect “Possible incoming XDCC Send Request Detected.”	In house
Ext -> Int	MY.NET.30.3 activity	1558	2287	2815	1697	8357	Main source hosts are 216.56.88.95, 68.55.156.128 and 141.157.21.74. Targeted mainly to ports 8009 and 524, the server seems to be running Netware 6 and offering remote file management	In house
Ext -> Int	MY.NET.30.4 activity	2560	1631	1844	6066	12101	Accounts for 34.23% of all alerts. Main source hosts are 67.20.160.15, 68.49.76.164 and 68.50.102.64. Main dst ports are 51442, 524, 80 and 8009, targeting MY.NET.30.4. Majority of alerts seem to follow same pattern as MY.NET.30.3 activity.	In house

Ext -> Int	NETBIOS NT NULL session	2	0	0	3	5	External hosts sending blank userid and password to login to three MY.NET.190.0 network hosts(CVE-2000-0347)	Snort 530
Int -> Ext	NIMDA – Attempt to execute cmd from campus host	0	0	0	1	1	From MY.NET.17.45:3297 to 69.90.32.141:80	In house
Ext -> Int	NMAP TCP ping!	98	174	135	106	513	Top TCP ports pinged: 53, 80, 143, 25 and 3472. Probable Snort signature should be deleted, because use of stateful firewall is common practice nowadays.	Snort 628
Ext -> Int	Null scan!	43	451	97	170	761	A tcp packet showed up with no flags set (null scan). All packets come from external sources, trying to gather information or bypass possible firewall.	Snort 623
Int -> Ext Ext -> Int	Possible trojan server activity	16	32	63	115	226	Either destination or source port is 27374/TCP. Several internal hosts should be further analyze, including MY.NET.24.44, MY.NET.34.11, MY.NET.12.6, MY.NET.24.34, MY.NET.12.7, MY.NET.24.74, MY.NET.60.39 and MY.NET.12.4	In house
Ext -> Int	Probable NMAP fingerprint attempt	0	0	0	1	1	From host 66.218.71.233:80 to MY.NET.152.16:2587, trying to determine the OS	Arachnids 5
Int -> Ext Ext -> Int	RFB – Possible WinVNC – 010708-1	0	0	22	3	25	Port 5900/TCP involved on all alerts, related to possible VNC server installed on hosts. 5900 is for web based server. Important to check internal hosts that send packets: MY.NET.70.225, MY.NET.70.156, MY.NET.42.2, MY.NET.70.210, MY.NET.150.244 and MY.NET.149.37	In house
Ext -> Int	SMB C access	12	5	30	32	79	All traffic sent to five MY.NET.190.0 network hosts, by different public hosts. Looks like there are shares allowed and public on that network. Destination ports are 139/TCP	Snort 533

Int -> Ext	SMB Name Wildcard	350	653	685	352	2040	Dest port is 137, the MY.NET hosts most likely are infected or compromised, connecting to external hosts	Arachnids 177
Ext -> Int	SUNRPC highport access!	20	66	80	137	303	Mainly a port scan from host 142.165.212.10 to subnet MY.NET.70.0, but more source hosts than from detect “Attempted Sun RPC high port access”. Some look like false positives, such as listings.ebay.com (66.135.210.143) and www.mts.ru (213.87.4.1) which are web servers. Destination port is always 32771/TCP. The scan is looking for SunOS hosts that have the portmapper listening on this port. Some firewalls forget to block high ports, allowing an attacker to access the portmapper even when port 111/TCP is blocked.	Snort 599 or Arachnids 26
Ext -> Int	SYN-FIN scan!	0	1	0	3	4	High source and dest ports, targeted all but one to 130.85.42.6	Snort 624
Ext -> Int	TCP SMTP Source Port traffic	12	0	3	4	19	False alarm, source and destination hosts are most probably mail servers. Source port 25 to dest ports 110 or 25 (hostname of MY.NET.12.4 is mail and MY.NET.12.6's is mxin)	In house
Ext -> Ext	TCP SRC and DST outside network	19	10	16	18	63	Must likely are DoS packets, source addresses are either from AmericaOnLine ISP or reserved subnetwork addressess (192.168.0.0 or 127.0.0.0). Further investigations should determine source of these packets.	In house
Ext – Int	Tiny Fragments – Possible Hostile Activity	1	0	0	0	1	From 68.33.95.20 to MY.NET.25.21	Snort 522

Int -> Ext Ext -> Int	TFTP – External TCP connection to internal tftp server	0	0	0	5	5	Established TCP communication between MY.NET.190.92 (69/TCP) and 82.82.67.46 on March 10.	In house
Ext -> Int	TFTP – External UDP connection to internal tftp server	0	0	4	0	4	142.165.212.10 attempts to connect to port 69/UDP of hosts MY.NET.190.102, MY.NET.190.202 and MY.NET..6.15	In house
Ext -> Int	TFTP – Internal UDP connection to external tftp server	0	5	0	4	9	Source hosts are external (69/UDP)	In house
Ext -> Int	TFTP – Internal TCP connection to external tftp server	0	0	1	0	1	From 66.160.63.193 (69/TCP) to MY.NET.60.39 (45009/TCP)	In house
Ext -> Int	Traffic from port 53 to port 123	0	0	1	0	1	Message has to be corrected, only alert is from host 63.68.228.197 (123/TCP) to MY.NET.1.3 (53/TCP)	In house
Int -> Ext Ext -> Int	Portscans here...	Not determined	Not determined	Not determined	Not determined	Not determined	Activity from MY.NET.110.72 looks suspicious. The scans detected for hosts MY.NET.1.3 and MY.NET.1.4 are DNS queries since they are DNS servers. Scans detected for MY.NET.34.14 are actually mail transfer since this host is an imap server and mail exchange.	Snort PortScan Preprocessor (SPP)
Ext -> Int	OOS packets	685	752	444	436	2317	Particular attention should be given to internal hosts MY.NET.199.138 and MY.NET.199.158, they repetitively sent OOS packets to several internal web and mail servers. Several external hosts also aimed at internal web and mail servers.	None
	TOTAL	**11355**	**8339**	**6833**	**11147**	**37674**		

Appendix B

The "[UMBC NIDS] External MiMail Alert" was one of the alerts to be analyzed in depth on this report. Based on the log files, we could only predict it as an inbound traffic along with the source and destination addresses and ports. Still, the rule content was unknown to us. Below is a list of signatures found on the bleedingsnort.com website that might provide possible contents for the rule implemented on the university's IDS:

```
alert TCP $EXTERNAL_NET any -> $HOME_NET 25 (msg:"BLEEDING-EDGE Virus
MyDoom.I worm - inbound";
content:"zSG4AUzNIVRoaXMgcHJvZ3JhbSBjYW5ub3QgYmUgc"; nocase;
reference:url,secunia.com/virus_information/8818/; classtype:misc-activity;
flow:established; sid:2001673; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 25 (msg:"BLEEDING-EDGE WORM
Potential MyDoom.AI Email Inbound"; classtype:trojan-activity;
flow:established,to_server; pcre:"/X-AntiVirus\: (scanned for viruses by
AMaViS 0.2.1|Checked by Dr.Web|Checked for viruses by Gordano's
AntiVirus)/"; pcre:"/(Look at my homepage with my last webcam photos!|FREE
ADULT VIDEO! SIGN UP NOW!)/";
reference:url,us.mcafee.com/virusInfo/default.asp?id=description&virus_k=12
9631; sid:2001437; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 25 (msg:"BLEEDING-EDGE WORM
Potential MyDoom.AH Email Inbound"; classtype:trojan-activity;
flow:established,to_server; content:"My name is Jane, I am from Miami, FL";
nocase; content:"with my weblog and last webcam photos!"; nocase;
reference:url,us.mcafee.com/virusInfo/default.asp?id=description&virus_k=12
9631; sid:2001433; rev:2;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 25 (msg:"BLEEDING-EDGE WORM
Potential MyDoom.AH Email Inbound"; classtype:trojan-activity;
flow:established,to_server; content:"tracking number is A866DEC0"; nocase;
reference:url,us.mcafee.com/virusInfo/default.asp?id=description&virus_k=12
9631; sid:2001431; rev:2;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 25 (msg:"BLEEDING-EDGE WORM
Potential MyDoom.AH Email Inbound"; classtype:trojan-activity;
flow:established,to_server; content:"Hi! I am looking for new friends. I am
from Miami, FL."; nocase;
reference:url,us.mcafee.com/virusInfo/default.asp?id=description&virus_k=12
9631; sid:2001435; rev:1;)
```