Global Information Assurance Certification Paper

GIAC Practical
V 4.1
University Intrusion Report

Prepared for: University
Prepared by: Daniel Accioly Rosa
Date Submitted: 13/12/2004

INDEX

1. Executive Summary

1.1. Background

This work is a sample of an Integrated Security Assessment for Corporate Networks. This kind of assessment service is done for all kinds of Companies worldwide and I hope the results of this Analysis are helpful to the University of Maryland.

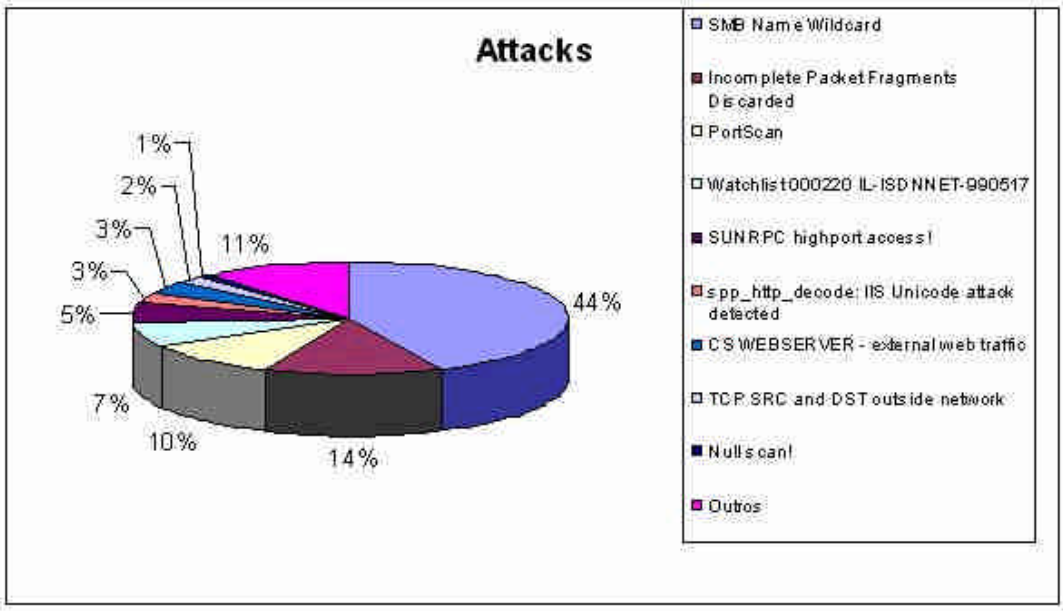
The work was performed by analyzing log files from a given intrusion detection sensor. These log files record unusual traffic on the network, alerts registered for know attacks and probes. Information available on the Internet was also used in the development of this report.

The scope of the analysis was the period between 17-02-2003 through 19-02-2003. During these three days, a total of 109.684 alerts were recorded, or nearly 37.000 alerts per day. Because of the number of security events detected, it is clear that there is not enough staff to properly monitor 37.000 events per day. An abundance of alerts like this makes the analysts indifferent, which has adverse effects on security.

Unfortunately, due to University's restrictions, I did not have access to all the information that was needed, so the work was somewhat compromised by the lack of input required to do full analysis. All points that can be detailed or clarified by further analysis with more information are pointed through the document.

1.2. Attacks

The main attacks that were identified in the network were:



1.3. Problems Found

The analysis of these attacks pointed many problems. The most important are the following:

- Ineffective workstation management

Several applications not related to University business were found on the workstations. Virus and worms also were infecting all the network, showing the need of effective antivirus tools and policy, as well as patch management, application installation and configuration control.

- Poor Internet Traffic control

There was traffic originating from the internet directed to internal computers. Some malicious traffic was also detected coming out of the network to the internet. There is the need to improve perimeter security to better control this traffic. If this is done there may be leak of important information or compromise of internal resources.

- Little documentation of the network

The University was unable to provide all the documentation required for the analysis on the time the alerts were recorded. This is critical. Documentation of network topology, resources and services should be done as soon as possible.

- Poor logging

Several events recorded by the sensor were difficult to analyze because they lacked additional information. The amount of false positives and undocumented rules also impacted the analysis and improved the difficulty to focus on the proper Events of Interest

1.4. Important Recommendations

Most security measures impact usability. This is a fact and cannot be changed. The recommendations below will have a direct impact on the way the University works. No security recommendation should prevent the University to produce its scientifical material as well as teaching students. Security should exist to support these goals and this should be kept in mind when selecting the way these recommendations should be implemented.

The following should be done to improve the security of the network:

- The University should consider migrating it network to a reserved address space. The use of a 10.x.x.x/8 subnet for example prevents external users to direct access the University internal resources, and can be managed at the firewall with NAT (Network Address Translation).

- Firewall rules need to be tightened down. It should be configured to "deny-what-is-not specifically-allowed". If there is no valid business reason to allow connections to port 515 from the internet, then this port and others like it should be blocked by the firewall.

- There are several alerts that are related to different worms and viruses. By adding some anti-virus scanners, both on the workstations and the gateways, this would help cut-down on the virus problem within the network.

- Better sensor configuration should be done. It is important to understand what attacks and probes are being done to my network, so proper tuning of the rules, as well as logging only the necessary events, would give the possibility to get more details on what requires attention and not false positives and bogus traffic. Also, this would make correlation with logs from other devices easier, and could decrease the incident response time and improve its efficiency.

- Central management of these alerts, as well as better rule configuration is essential for adequate protection. The University might want to look for SIM (Security Information Management) Systems in order to consolidate the various logs and alerts. Better logging is essential!

- The use of proxies serves as a good point of control of the network. No traffic should go inbound or outbound without being compared to an application level rulebase that allows only legitimate protocols in and out of the network.

- The use of patch management will help close a lot of the vulnerabilities that many of these worms and viruses target.

- To avoid problems with students and teachers with specific research needs, create laboratories separated from the main University Network that allows installation of tools and testing software. It is important that these labs are separated from the administrative network and segments that hold sensitive traffic. This would give a good and free learning environment for students and not compromise University sensitive and important material.

The content below is a more detailed description on how these conclusions were reached. This is indicated for engineers who need to deeper understand these problems and assess them in a proper way.

2. Detailed Analysis

2.1. University Analysis

The only data provided for the practical were the Alert, OOS and Scan Files. This data provides limited analysis capability over the network, so I had to research a little further to be able to do better analysis.

The scope of the analysis was the period between 17-02-2003 through 19-02-2003. The ideal analysis would have been a whole week, since we cannot assume that the network behavior would be the same of these days, specially the weekends. Due to a personal time restriction, I could not analyze more than three days.

The data is related to a period of Monday through Wednesday.

The following files were analyzed:

File Name	Type
scans.030217	Scan File
scans.030218	Scan File
scans.030219	Scan File
OOS_Report_2003_02_17_6137.oos	Out Of Spec File
OOS_Report_2003_02_18_27913.oos	Out Of Spec File
OOS_Report_2003_02_19_479.oos	Out Of Spec File
Alert.030217	Alert File
Alert.030218	Alert File
Alert.030219	Alert File

2.1.1. Relational Analysis

The following section is an analysis of three days worth of logs from the University. The log files were obtained from http://www.incidents.org/logs. A Snort Intrusion Detection Sensor (IDS) generated the logs. Snort is an open source IDS that was created by Marty Roesch (Snort). More information on Snort can be found at www.snort.org.

The files represented 109684 alerts, 4017 OOS (Out-of-Spec) packets and 64319 scans.

2.1.1.1. Finding Additional Information

The first step I had to do was to find additional information, since only going through the files provided by the University would give me limited vision on the resources available at the network and potential targets to crackers.

By analyzing the scan files, I came to conclusion that the 130.85.0.0 network was the block owned by the University. This was possible since the scan files did not have the "MY.NET" mask. Crossing the alert, oos and scan files information, I could see that there were many hosts having identical final octets. This was a big coincidence, and after reverse-looking the addresses I was able to discover the umbc.edu domain, which is owned by University of Maryland, Baltimore County:

OrgName: University of Maryland Baltimore County
OrgID: UMBC
Address: 1000 Hilltop Circle
City: Baltimore
StateProv: MD
PostalCode: 21250
Country: US

NetRange: 130.85.0.0 - 130.85.255.255
CIDR: 130.85.0.0/16
NetName: UMBCNET
NetHandle: NET-130-85-0-0-1
Parent: NET-130-0-0-0-0
NetType: Direct Assignment
NameServer: UMBC5.UMBC.EDU
NameServer: UMBC4.UMBC.EDU
NameServer: UMBC3.UMBC.EDU
Comment:
RegDate: 1988-07-05
Updated: 2000-03-17

By accessing the http://www.umsc.edu webpage, I could find some information regarding the current University structure. This was used on the construction of this report.

The main sources of information were:

- The University of Maryland Server Hardware List, located at http://www.gl.umbc.edu/hardware.shtml.
- Intermapper System at http://noc2.noc.umbc.edu/~admin/map_screen.html.

Also, the Walter Claire's (http://www.giac.org/practical/GCIA/Wouter_Clarie_GCIA.pdf) and Loic Juillard's (http://www.giac.org/practical/GCIA/Loic_Juillard_GCIA.pdf) GCIA Practicals were very important on organizing this part of the work.

It is important to say that no probes but reverse lookups and traceroutes were made in order to discover information from the University. Also, that all information stated below is sanitized as a practical requirement. All 130.85 was changed for MY.NET.

2.1.1.2. The Network

Overall, the network has some major problems that demand immediate attention. I noticed many types of events that can compromise the security in very serious ways. There appeared to be machines participating in Trojans, zombies, botnets, worms and all types of network scans. Most of this activity originated from outside the University's network, but we also had some traffic coming from inside. There are a lot of problems to resolve, but I think a couple of simple actions would help a lot to improve security.

2.1.1.2.1. Big Picture

According to its webpage, the University of Maryland, Baltimore County (UMBC) network provides Internet access for faculty staff, administrative staff and students, file storage and database servers, e-mail service (IMAP and Pop3), web services for Intranet and Internet users, backup, newsgroups (Usenet) and remote access through dial-up modems and VPN (Resnet).

These services are offered by a number of hosts spread through the campus. The IP distribution appears to be made by physical location, so there are subnets which are composed mainly by workstations on student dorms and other that holds important servers and core network equipment. The following network subnets and core equipment could be identified:

- Support Services Subnet:
Network Address: MY.NET.1.0/24
Services Identified: I could find only DNS and System logging servers.

- Generic Server Farm Subnet:
Network Address: MY.NET.24.0/24
Services Identified: This looks like a big Server Farm with miscellaneous services. The most important found were the Directory Servers, Mail Servers (outgoing mail exchangers and Pop3/IMAP servers), Web servers and several File Severs. Other non-specific or unidentifiable services are provided as well.

- Peoplesoft Subnet:
Network Address: MY.NET.32.0/24
Services Identified: All servers on this subnet are related to UMBC Peoplesoft

- NOC (Network Operation Center) Subnet:
Network Address:MY.NET.9.0/24
Services Identified: The Network Operation Center is the UMBC department responsible for maintaining the network health. It provides off campus dial in access from the Baltimore and Washington DC area. It has real-time monitoring for the main campus servers and is also responsible for troubleshooting network problems throughout both the Main Campus and the Technology Center. UMBC's NOC also manages a Residential Network (see Resnet below) which provides almost 3000 students with ethernet access to

the network and internet. All off campus data communications are managed by the NOC. These include: the commercial internet link, or high speed internet 2 link, and a high speed ring between UMBC, College Park and UMB.

I couldn't conclude what the servers that are in this subnet and are listed on the Intermapper System did by their names, and unfortunately there was no server from this network on the Hardware List (Appendix). But the following servers exist on this subnet: bourbon.noc.ubmc.edu, leo.noc.ubmc.edu, pluto.noc.ubmc.edu, jupiter.noc.ubmc.edu, grain.noc.ubmc.edu, vodka.noc.ubmc.edu. As the NOC subnet holds network management services, they would be important targets for attacks.

- IRC (Imaging Research Center) Subnet:
Network Address: MY.NET.7.0/24
Services Identified: At first I though UMBC was offering Internet Relay Chat Services in its network, As this does not look as a normal practice, I went investigating a little further. IRC is a department from UMBC that deals with visual art and they offer courses for artists with background on technology. No big target but it was one discovery that deserved to be registered. MY.NET.7.1 is the irc-gw.umbc.edu.

- Netware Subnet:
Network Address: MY.NET.30.0/24
Services Identified: As there was an alert logging traffic from and to some hosts on this network, as well as information on the Intermapper System that said there were Novell Servers on this subnet, I thought it was important do document this on the report, since most of the UMBC servers are Solaris and Unix, and most of the workstations are Windows. Further analysis on the reason these servers exist is done in section 2.2.1.2.2. I have reason to believe that these might be honeypot servers.

- Resnet:
Network Address: Various on the MY.NET.0.0/16 address space
Services Identified: "ResNet" generally refers to a university's on-campus (or residential) computer network infrastructure. Users access this network by hooking their computers into a dorm's RJ45 data port or connect via wireless. From there the user can access the rest of UMBC's local campus network or continue out into the Internet. One interesting fact of ResNet is that users may run servers, but they are required by security policy not to offer pirated MP3s, warez, images, etc. Port 24842 is currently open for users wishing to make their servers available outside of ResNet. No other ports are available for external access to servers within ResNet. Examples of acceptable servers (by security policy) include freeware webservers (offering only non-copyright-violating material), licensed game servers, homebrew quote-of-the-day servers, etc. Access servers identified for this network include MY.NET.9.12 (acs.noc.umbc.edu) and 138.85.30.66 (anubis.umbc.edu). Resnet gateway is probably MY.NET.8.33 (resnet-gw.umbc.edu).

- Ernie:
Network Address: MY.NET.1.1 and others!
Services Identified: Ernie is one big core router that connects several subnets on the network. Doing Reverse lookups I was able to discover some of its network addresses

(MY.NET.1.1, MY.NET.2.1, MY.NET.5.1, MY.NET.6.1, MY.NET.7.1, MY.NET.8.1, MY.NET.9.1, MY.NET.10.1, MY.NET.11.1, MY.NET.12.1, MY.NET.13.1, MY.NET.17.1, MY.NET.18.1, MY.NET.20.1). There are a lot more. Ernie is widely used and probably is connected to multiple switches and VLANs. It also appears that it is the entrance of the network. A traceroute command to a host in UMBC network showed him as the first MY.NET.0.0/8 address to be accessed.

2.1.1.2.2. Running Services

The following table lists the internal hosts as well as the services used based on the traffic seen in the log files. I determined the services available based first on the outbound traffic source ports (<1024) seen in the alert file for the five days analyzed. Since those systems sent packets from those services source port, there is a high probability that the system is actually listening on that port too. There is also the name of the service that commonly is related to the port.

This information was also compared to the Hardware List taken from UMBC's webpage and replicated with extra information in the Appendix.

Host	Ports	Common Services for Ports
MY.NET.1.3	53	DNS Server
MY.NET.6.40	25	SMTP Server
MY.NET.6.47	25	SMTP Server
MY.NET.24.8	119	NNTP Server
MY.NET.24.15	515	Print Server
MY.NET.25.10	25	SMTP Server
MY.NET.30.4	53	DNS Server
MY.NET.100.69	515	Print Server
MY.NET.100.165	21,80	FTP Server, Web Server
MY.NET.137.7	53	DNS Server
MY.NET.243.122	80	Web Server

Besides those machines, some other traffic caught my attention:

- MY.NET.30.3 and 30.4

Both machines appears to be running multiple services, like NDAP (Novell Directory Access Protocol)/NCP (Novell Core Protocol), RPC Services/Windows Messaging, HTTP, NetBIOS, and others. It was strange to find out Windows and Novell services on the same host. These could be either Novell 4 or later software or honeypots, since there is a rule in Snort to create an alert each time traffic is detected from outer networks toward these machines.

More information would be needed to proper asses the situation of these computers.

- MY.NET.X.X Print Server traffic

It seems that either there are a lot of print servers running on the network or someone might be probing the hosts for them. Dozens of connections from internet machines are detected to internal computers on port 515. These are detected by a rule that points traffic aimed to port 515 from outside networks. No scans were detected on the scan files to this port, so I'm assuming this requires further investigation. As most traffic goes to MY.NET 10.69 and MY.NET.24.15 I presume that they are the main Print Servers and the situation of the others is unknown, in need of further analysis.

I decided to do a Linkgraph on this and found two distinct situations. The first one shows two computers trying to connect to several different internal hosts. The first computer (81.48.108.90) is from an adsl service provider in UK and the other one (81.53.10.95) is from an adsl service provider in France.

The second situation shows two computers on the Internet trying to do multiple connections on port 515 to the same hosts. I could confirm one of these hosts to be a real print server by looking at the Hardware List at the Appendix. This was the printhost.umbc.edu using the address MY.NET.24.15 and is a Sun Ultra 5 print server running Solaris and serving SSH and LPD services. The other could not be identified, but, the computers on the Internet had IPs from the Maryland area, showing that this might be legitimate.

Further analysis should be done on the first case by analyzing the content of the packets directed to the computers on port 515. This could be something from a configuration error, virus, Trojans, or even convert channel and botnets.

On the second case I'd recommend to keep monitoring those IPs. If the situation persist look for configuration errors as these might be laptops that were configured to use the print servers locally at the University campus and now are connected directly to the internet outside UMBC network.

The linkgraph is available on section 2.2.1.2.3.

- P2P traffic

Several machines are running Kazaa and E-Donkey. I could notice that by the traffic to ports 1214 and 4662 as shown below:

```
02/19-00:30:11.105778  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.100.82:1113 -> MY.NET.250.154:1214

02/19-00:35:00.620114  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.91.121:4662 -> MY.NET.222.134:2507
```

Also there are Out-of-Spec packets that were captured that are sign of these P2P protocols. Below follows an example:

```
02/17-00:49:04.265209 68.47.200.243:3665 -> MY.NET.201.62:6346
TCP TTL:114 TOS:0x0 ID:29386 IpLen:20 DgmLen:1187 DF
******** Seq: 0x37A0462  Ack: 0x43BEB085  Win: 0x5018  TcpLen: 0
```

```
00 00 6F 39 11 91 A3 3B A1 C8 FF 57 1D C5 FD DE  ..o9...;...W....
49 00 81 0A 00 64 04 00 00 08 CA 18 44 2F C8 F3  I....d......D/..
83 00 00 00 05 00 00 00 BE 64 4E 00 44 61 76 65  .........dN.Dave
20 4D 61 74 74 68 65 77 73 20 42 61 6E 64 20 2D   Matthews Band -
20 41 6E 67 65 6C 28 6C 69 76 65 29 2E 4D 70 33   Angel(live).Mp3
00 31 32 38 20 4B 62 70 73 20 34 34 20 6B 48 7A  .128 Kbps 44 kHz
20 35 3A 32 31 1C 75 72 6E 3A 73 68 61 31 3A 4C   5:21.urn:sha1:L
52 35 4C 45 59 4E 34 4F 57 4D 56 58 4E 32 4E 45  R5LEYN4OWMVXN2NE
```

Althought I suspect this capture was done as OOS because it has a TCP length of 0, the most important thing here is that it proves to be a P2P packet because of its payload (mp3 filename).

- SMB Traffic

There are many Windows machines in the network and Windows machines like to talk to each other and find new "friends" using ports 137 and 139. This is regular traffic. There is no reason for the sensor to alert the administrator on port 137 communication within internal hosts. The sensor should be better tuned and this traffic ignored.

On the other hand, when an attacker does a scan or probe on a Windows machine, it tries to answer back by establishing communication also using this port. The attacker also gets to know all the information that the victim host has on other Windows machines on the LAN. This is not desirable on the internet channel and usually shows that malicious traffic is going through. SMB traffic should be blocked on the firewall from the internet and monitoring should be done for these packets to guarantee that no attacker is probing our internal network without our knowledge (multiple layer defense).

2.1.1.2.3. Link Graph and Network Topology

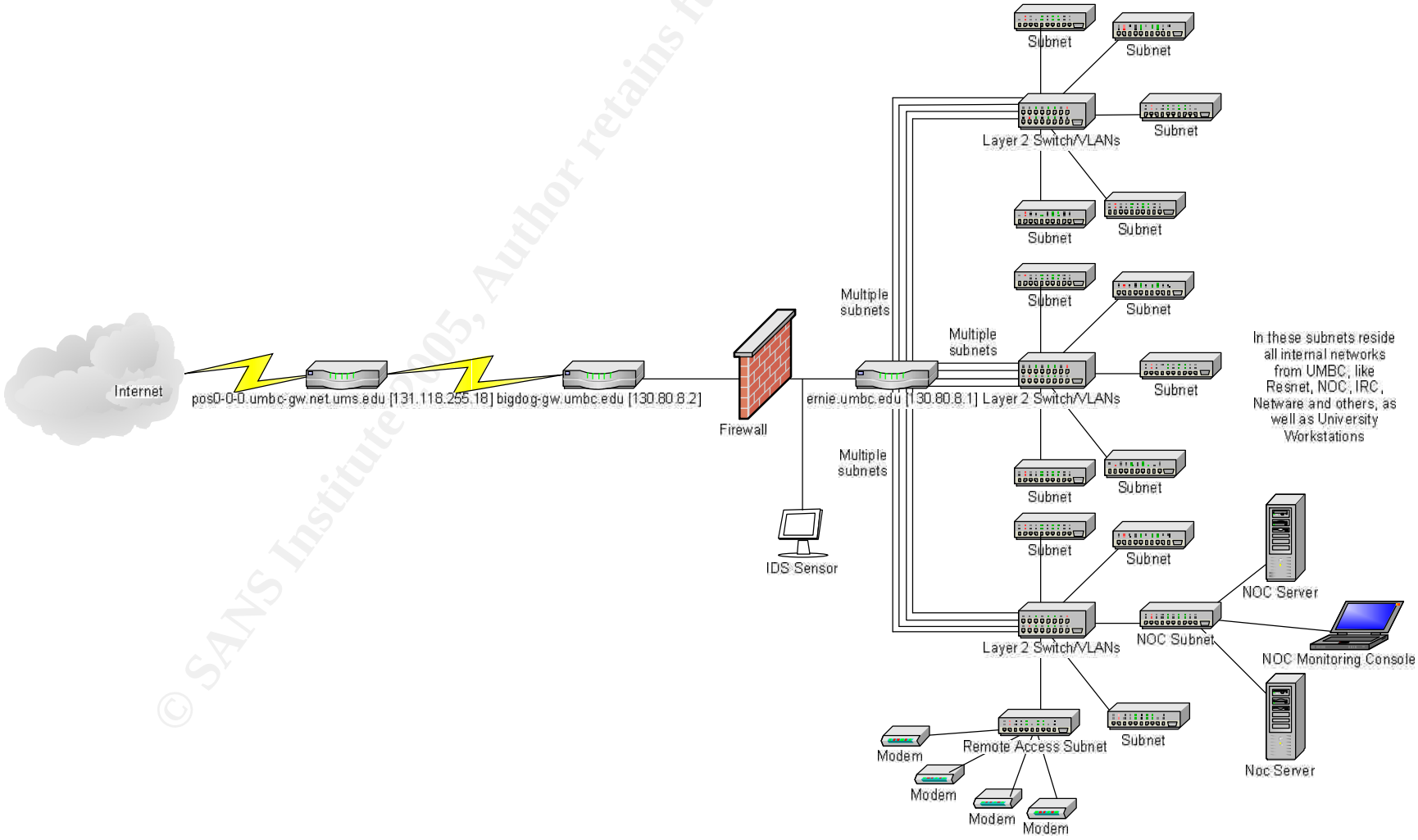
Below the estimated network topology is displayed. We can see UMS (University System of Maryland) providing access to UMBC through a remote router. This was discovered doing a traceroute back to ernie and bigdog-gw. I could not find out for sure who was the Internet router, but as both bigdog-gw and ernie are translated as MY.NET.8.2 and MY.NET.8.1 respectively and ernie is the big core router, thus connected to the internal network, I'm assuming bigdog-gw as the border router and connected to ernie to provide an exit to the Internet. Other thing that helped me to assume bigdog-gw as the border router is that it is also known as ecsborder-gw.umbc.edu.

I decided to assume a Firewall on the topology, since I could discover by information available on UMBC webpage that the University has a firewall and that would be the most important position for it to be. The sensor was put in that position because it was able to capture traffic from all network to internet and from the internet to the internal network. No traffic between internal hosts was captured in the alert, oos and scan files.

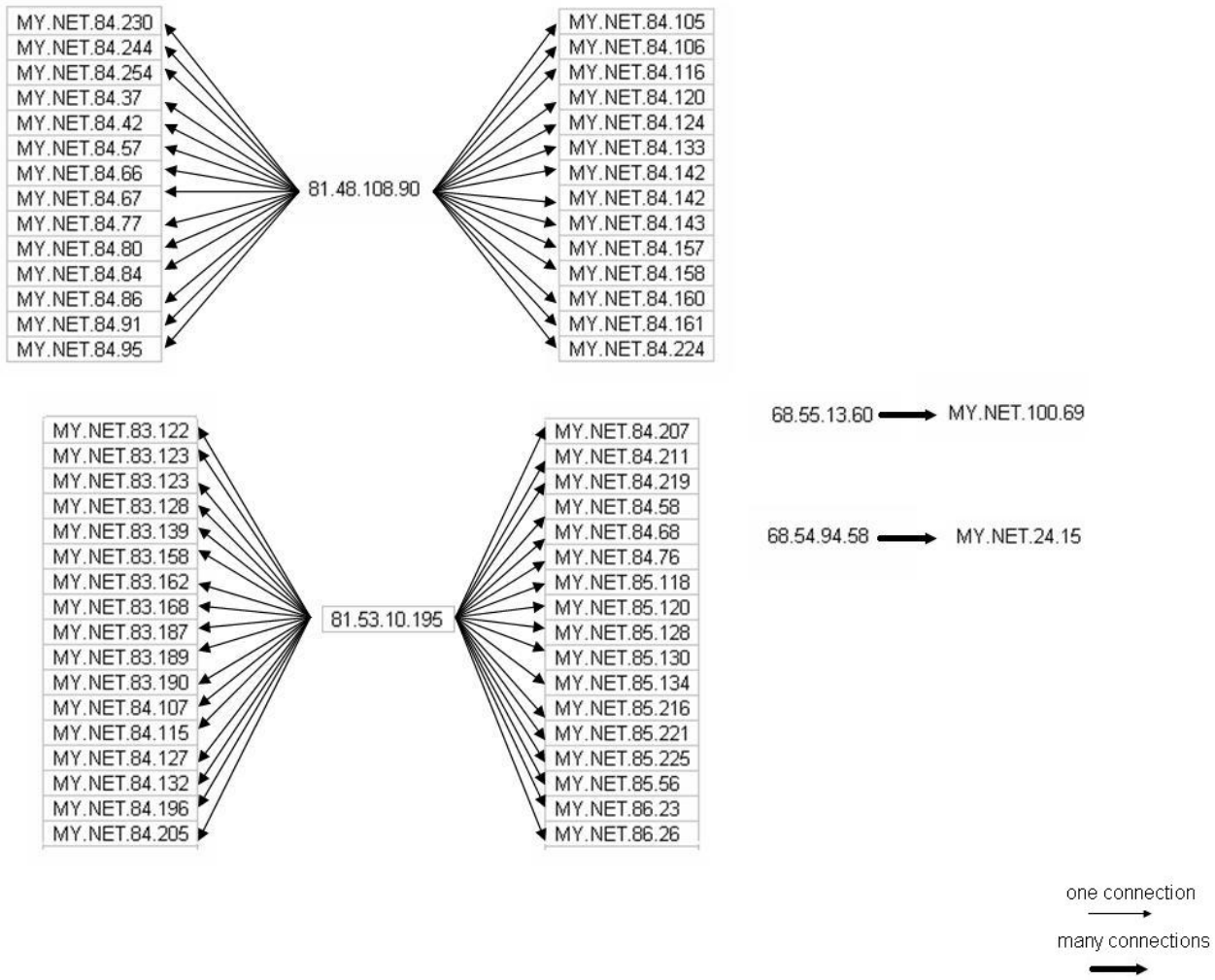
We can also see in the topology ernie working as a router between the various subnets. I'm assuming there are several layer-2-switches, since a "layer-3-switch" could route packets between those VLANs/subnets. We can also see, in the endpoints, some switches/hubs

which are the connection points for the hosts and are entrance points for the wireless network. I decided to put the Resnet modem pools and the NOC subnet as examples of such connections.

In the next page you are also able to see the a Linkgraph. It shows the difference types of access detected by the snort rule "Connect 515 from outside". One looks like a scan, where we have an address from an adsl provider trying to connect to multiple internal hosts on port 515. The other are internet IPs from the Maryland area trying to connect to legitimate print servers on port 515. More information can be found on section 2.2.1.2.2, "MY.NET.X.X Print Server traffic".



University of Maryland, Baltimore County Estimated Network Topology.



Linkgraph showing the difference between the two types of connections found in the "515 connect from outside" alert.

2.1.2. Alert Analysis

Several alerts were recorded by the sensor on the network. Unfortunately, the original rules that triggered those alerts were not available to better understand the Administrator motives to call this traffic "Events of Interest".

I performed the analysis below on using the default snort rule base and the default configuration available on the sensor.

The following alerts were recorded:

Total Number of Events	
Event Description	**Number of Events**
SMB Name Wildcard	47454
Incomplete Packet Fragments Discarded	14871
PortScan	11520
Watchlist 000220 IL-ISDNNET-990517	7919
SUNRPC highport access!	5696
spp_http_decode: IIS Unicode attack detected	3869
CS WEBSERVER - external web traffic	3416
TCP SRC and DST outside network	2719
spp_http_decode: CGI Null Byte attack detected	1967
High port 65535 tcp – possible Red Worm - traffic	1616
TFTP - Internal TCP connection to external tftp server	1360
Watchlist 000222 NET-NCFC	987
TFTP - External UDP connection to internal tftp server	905
Null scan!	867
MY.NET.30.4 activity	645
High port 65535 udp - possible Red Worm - traffic	598
Tiny Fragments - Possible Hostile Activity	406
connect to 515 from outside	385
Possible trojan server activity	352
Queso fingerprint	343
IDS552/web-iis_IIS ISAPI Overflow ida nosize	290
TCP SMTP Source Port traffic	283
EXPLOIT x86 NOOP	241
External RPC call	209
Port 55850 tcp - Possible myserver activity - ref. 010313-1	129
MY.NET.30.3 activity	116
CS WEBSERVER - external ftp traffic	102
IRC evil - running XDCC	82
TFTP - External TCP connection to internal tftp server	71
EXPLOIT x86 setuid 0	57

Total Number of Events	
Event Description	**Number of Events**
NMAP TCP ping!	56
EXPLOIT x86 stealth noop	34
Notify Brian B. 3.54 tcp	21
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	20
SNMP public access	20
EXPLOIT x86 setgid 0	17
TFTP - Internal UDP connection to external tftp server	12
Notify Brian B. 3.56 tcp	11
Probable NMAP fingerprint attempt	6
FTP passwd attempt	5
PHF attempt	2
SMB C access	2
Fragmentation Overflow Attack	1
Port 55850 udp - Possible myserver activity - ref. 010313-1	1
RFB - Possible WinVNC - 010708-1	1

Some of the alerts are explained briefly below. These were taken from Mike Bell´s GCIA Practical. It is available at http://www.sans.org/y2k/practical/Mike_Bell_GCIA.doc

- Watchlist 000220 IL-ISDNNET-990517

A detect setup to alert on activity from an ISDN network in Israel.

- Watchlist 000222 NET-NCFC

A detect setup to alert on activity from net NCFC (The Computer Network Center Chinese Academy of Sciences).

- SNMP public access

SNMP is a protocol used for the monitoring and management of network attached devices. Many devices now have SNMP agents available. SNMP can be used in the recognizance phase of an attack, or with the proper password can be used to change the configuration of a device.

- Null scan!

A scan which attempts to evade filtering devices by setting all TCP flags to a null (zero) value. A second use of the Null scan is OS fingerprinting.

- SMB Name Wildcard

A wildcard scan will elicit a listing of all NetBIOS names which are known by the requested host. Windows machines usually do this scans in order to map other similar

computers on the network. When malicious, there appear to be two main reasons for these scans: to enumerate targets known by that host, and propagation of the Internet worm network.vbs.

- Queso fingerprint

A scan which sends a series of OOS packets to a host in order to try to determine its operational system. This information can then be used to target known OS vulnerabilities.

- NMAP TCP ping!

A scan performed by the tool NMAP to determine which hosts are up on a network. This scan is used to evade blocking by sites which do not allow ICMP echo requests into their network. NMAP sends an ACK to target hosts in an attempt to elicit a RST from live hosts.

- Connect to 515 from outside

This alert is produced by a connection attempt to TCP port 515 originating from within the $EXTERNAL_NET network. There are exploits associated with LPRng, which runs at port 515, allowing execution of arbitrary code or a possible DOS of the printing services.

- Probable NMAP fingerprint attempt

Nmap provides sophisticated OS fingerprinting scans capable of identifying a large number of operational systems. An attacker can then use this information to target known vulnerabilities of those operating systems and associated software.

- External RPC call

This alert was triggered by scans to port 111 or 32771, the portmapper service. Information can be retrieved from portmapper which can be used to target known vulnerabilities in RPC related services.

- Tiny Fragments - Possible Hostile Activity

Tiny fragmentation is used to avoid detection on filtering devices and IDS systems which do not do packet re-assembly. Fragments can thus at times avoid detection, where the entire packet may not.

2.1.2.1. Top Talkers

The following destination addresses were the most attacked computers on the network, or at least had the bigger number of alerts generated.

2.1.2.1.1. Top Destinations

Below is a table that shows us the top 10 IPs that were attacked:

Alerts	
Destination Address	**Alerts**
198.247.231.42	6959
216.111.123.20	6653
MY.NET.252.126	5510
MY.NET.100.165	4323
216.209.164.171	1790
MY.NET.206.242	1282
MY.NET.222.174	1229
209.10.239.135	1113
192.168.0.253	905
MY.NET.24.34	810

What this table tells us is that a large number of alerts are being generated from inside and outside of our network. This could be caused by bad firewall policy, as well as bad network architecture (lack of a proxy to control outbound traffic).

In a good network architecture, the internal network should not be visible from outside. Usually is a good practice to hide the internal address space by assigning the host reserved IPs like the 10.0.0.0/8 subnet for example and NAT the outbound traffic to the internet. We can see that this is not the case.

A screened subnet might exist on the firewall to allow outer access to public services.

I reviewed the most serious attacks that targeted the internal addresses.

- MY.NET.222.174

MY.NET.222.174 is a very interesting case. It has several alerts recorded where it acts like a source of Red Worm infection, it originates CGI Null Byte and Unicode attacks to different web servers on the internet, and do some portscans on servers. It appears as some "smart user" that is trying to use our network resources to perform attacks. It would not be a big problem, since we could locate this machine and shut it down, reprehending the user, but I found other interesting traffic.

Some packets originating from the same source (212.179.105.210) on different ports target port 4662, which is common port for e-Donkey P2P application. The traffic happens every day at the same time.

I also found some out-of-spec packets with strange payload in it. An example is detailed below:

```
02/18-09:54:17.103069 148.63.237.176:1025 -> MY.NET.222.174:4662
TCP TTL:113 TOS:0x0 ID:57405 IpLen:20 DgmLen:113 DF
****P*** Seq: 0x2717A0A  Ack: 0x0  Win: 0x2000  TcpLen: 20
E3 44 00 00 00 01 10 28 8D 51 58 BE CA DC 37 E0  .D.....(.QX...7.
68 31 E6 09 88 43 9A 94 3F ED B0 36 12 02 00 00  h1...C..?..6....
00 02 01 00 01 10 00 66 6F 75 72 70 61 73 74 6D  .......fourpastm
69 64 6E 69 67 68 74 03 01 00 11 2D 00 00 00 D9  idnight....-....
E3 13 28 BA 1B D8 41 C7 A6                       ..(...A..
```

We have the risk of a remote attacker accessing our computer and controlling it for remote attacks. I would definitively go for more analysis with more information on this attack on section 2.3.2.

- MY.NET.24.34

MY.NET.24.34 looks like a Windows machine that all the network is trying to talk. Maybe is a Domain Controller, WINS or alike. We can see an enormous amount of SMB packets to it, so it is normal to be in the most alerted hosts.

I also noticed some alerts pointing attacks from it. For me they looked as packets that were replies of normal connections, and thus false-positives. There are also some malformed packets with the 12****S* flags from multiple sources aimed to this host. As 12 is 001100 in binary, and they this is in a reserved part of the TCP header, configuring something out-of-spec, I couldn't figure out what that was, and I would need more information to investigate.

Below follow an example of this kind of OOS packet:

```
02/16-09:42:45.247376 217.97.149.38:58581 -> MY.NET.24.34:80
TCP TTL:43 TOS:0x0 ID:38081 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xBE865D37  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 468896246 0 NOP WS: 1
```

2.1.2.1.2. Top Sources (Top Three Most Suspicious external Addresses)

Besides the most attacked destinations, I also ranked the Top Ten Attackers. This was important to understand what kind of threat we were under. Below follows the information:

Alerts	
Source Address	**Alerts**
MY.NET.211.6	**13181**
169.232.84.146	**4724**

Alerts	
Source Address	**Alerts**
212.179.88.96	1280
212.179.105.210	1154
MY.NET.132.42	1018
212.179.91.129	964
12.35.158.199	820
159.226.5.220	761
MY.NET.204.74	735
66.72.199.111	682

The top three outside attackers that were investigated are listed below with the available whois information:

- 169.232.84.146

This IP is for University of California:

```
University of California, Office of the President UCNET-BLK (NET-169-228-
0-0-1)
                                  169.228.0.0 - 169.237.255.255
University of California, Los Angeles UCLANET4 (NET-169-232-0-0-1)
                                  169.232.0.0 - 169.232.255.255
```

- 212.179.88.96 and 212.179.105.210

```
These addresses are from a telecommunications company/internet provider
in Israel (bezeqint.net/BEZEQ-INTERNATIONAL-LTD):

inetnum:        212.179.80.0 - 212.179.89.255
netname:        CABLES-CONNECTION
descr:          CABLES-CUSTOMERS-CONNECTION
country:        IL
[…]
role:           BEZEQINT HOSTMASTERS TEAM
address:        Bezeq International
address:        40 hashacham st.
address:        Petach Tikva 49170 Israel
phone:          +972 1 800800110
fax-no:         +972 3 9203033
e-mail:         hostmaster@bezeqint.net
```

and

```
netnum:        212.179.96.0 - 212.179.106.255
netname:        CABLES-CONNECTION
descr:          CABLES-CUSTOMERS-CONNECTION
country:        IL
[…]
role:           BEZEQINT HOSTMASTERS TEAM
```

```
address:      Bezeq International
address:      40 hashacham st.
address:      Petach Tikva 49170 Israel
phone:        +972 1 800800110
fax-no:       +972 3 9203033
e-mail:       hostmaster@bezeqint.net
```

I have also made some analysis on the inside IPs:

- MY.NET.204.74

This host was defined as one of the most suspicious hosts on our internal network. Please see section 2.2.1.2.4 for a description of its activities.

From that info I conclude that internal networks might offer a major risk to the University. Despite the large number if malformed packets from the OOS files that indicates the wide use of Peer-to-Peer networks, Instant Messaging programs and virus, I could identify attacks that could impact on the network availability, theft of information and damage to reputation (USAF attack by the MY.NET.204.74 host). Action should be taken to remediate the situation as soon as possible.

We also have to worry though for the possibility of these machines being used as zombies for attackers on DOS attempts, or direct compromise, since we can find packets from the Internet directed toward them. A proxy should be used for all communication with the external world.

2.1.2.1.3. Top Ports

At last, but not al least, we have the most common attacked ports:

Alerts	
Destination Port	**Alerts**
137	47432
80	10616
32771	5693
1214	2463
135	1790
65535	1488
4662	1488
0	967
2708	832
1321	669

Then ports related to the most important attacks are outlined below:

- 80

Port 80 is the common port for HTTP. It is, of course, target for Unicode and CGI attacks. It looks like we have many IIS servers in our network vulnerable to these attacks. We should remove these servers were they are not necessary (they are installed by default in some Windows machines) and patch the ones that should exist.

- 1214 and 4662

Port 1214 and 4662 are for P2P Networks. There are a lot of Kazaa and e-Donkey hosts in the University. They should be removed as soon as possible. They can propagate virus, consume bandwidth and ruin user experience on the desktop by adding spyware and bugs. Also there is the legal issue on copyright material being shared without license.

- 135

Port 135 is a very interesting case. I found traffic from a host outside our network to another host outside our network. This traffic should not have been captured by out ids!!! It looks like a spoofed communication that requires further analysis. I will detail that in the section below.

- 0

Ports 0 is invalid and its used for multiple attacks. In our case, the port 0 stood for scans and fingerprint attempts.

- 65535

Port 65535 is known for being used by some Trojans and Linux worms. It seems that our network has some of these, as well as attackers accessing compromised machines on the Internet.

2.1.2.2. Most Suspicious Internal Hosts

One thing that is important to point in the report is the list of the most suspicious internal hosts. The machines outlined below definitively need to be further investigated, since the events recorded in the logs were serious enough to catch my attention:

- MY.NET.211.6

Several alerts could be found on the 19th regarding an infection with the Red Worm. This worm, also called Adore and should not to be confused with "Code Red", is a Linux-specific infection, typified by a backdoor listening on port 65535 (which otherwise should not be used). Traffic seen from this port strongly indicates a compromised host.

Adore spreads in Linux systems using four different, known vulnerabilities already used by Ramen and Lion worms. These vulnerabilities concern BIND named, wu-ftpd, rpc.statd and lpd services.

This internal machine was probably controlling (through a backdoor) or infecting another one on the Internet. The address that was registered was 140.32.16.100, which is owned by the USAF Academy at the Dod Network Information Center.

We can assume from the information above that this host might be trying to attack a United States Air Force computer. This does not look like a good idea and surely it is not desirable in the UMBC network. Action should be taken immediately to identify the user and further investigate the event and, if needed, inform the authorities.

-MY.NET.132.42

There was an alert registered at 10:37 on the 17th that showed the internet host 35.10.81.123, which is a computer at Michigan State University, accessing MY.NET.132.42 on port 6660, which is an IRC (Internet Relay Chat) port. This alert was registered as an EXPLOIT x86 NOOP, which is a series of NOPs sent to a remote program in order to crash it or take control over it. Below is the extract from the alert files that shows the attack:

```
02/17-10:36:28.202377  [**] EXPLOIT x86 NOOP [**] 35.10.81.123:2506 ->
MY.NET.132.42:6660
02/17-10:36:28.275441  [**] EXPLOIT x86 NOOP [**] 35.10.81.123:2506 ->
MY.NET.132.42:6660
02/17-10:36:28.388332  [**] EXPLOIT x86 NOOP [**] 35.10.81.123:2506 ->
MY.NET.132.42:6660
02/17-10:36:29.241309  [**] EXPLOIT x86 NOOP [**] 35.10.81.123:2506 ->
MY.NET.132.42:6660
02/17-10:36:30.034576  [**] EXPLOIT x86 NOOP [**] 35.10.81.123:2506 ->
MY.NET.132.42:6660
```

After that, at 11:00, the internal host MY.NET.132.42 started sending incomplete fragments to some different IPs on the Dulles (VA) area hosted by America Online. This traffic can be malicious, since fragments usually are used for many kinds of attacks, like covert channels and denial of service. The packets were sent until 12:45. The IPs that received the packets were 172.179.250.18, 172.181.251.235, 172.180.246.250 and 172.181.116.159.

The fact that our internal host received an alert for an IRC port shows that it might be infected with some kind of Trojan that is controlled through this protocol. It is common on the Internet for masters to control their zombies through Internet Relay Chat. This host requires further investigation.

Also, even the fragmentation event being one of the noisiest of all the alerts and is triggered because packet fragments were detected but not all the packets arrived, no stimulus was found for this activity, even after an exhaustive search through the log files.

One interesting thing is that I did notice that each connection that triggered this alert had both a source and destination port of 0. This activity could be due to several things, possibly a misconfiguration or a router corrupting packets. But it could also be crafted packets designed for a DOS since obviously the OS stacks were not designed to accept connections on this port or to create a connection with 0 as the source port. Obviously there is a problem with connections that utilize port 0, either as a source or a destination, which is not specified in the TCP RFC.

Like most of the things on this report, further investigation is needed.

- MY.NET.246.54

In order to elude Intrusion Detection Systems, attackers sometimes try to send commands and code broken into several packets through fragments small enough to pass the whole network, but in a high number so the IDS won't reassemble them, allowing bad traffic (exploits and other nasty stuff) to pass through.

Snort generates alerts when it finds a fragmented packet with datagram length smaller than 25 bytes, like the rule below:

alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Tiny Fragments"; dsize:< 25; fragbits:M; classtype:bad-unknown; sid:522; rev:2;)

Snort also uses a preprocessor for alerting to packets like this.

Even with the RFC allowing these small fragments, I could not find an application that uses small fragmentation for any functionality, so I'm assuming that this rule cannot be a false positive.

This host, which I suspect to be a Windows machine because of the SMB Name Wildcard traffic generated, has been victim of several port scans on the 17th and the 19th from 11:00 to 19:00. The source IP for this scan was 216.18.16.107, which appears to be a regular Internet user from the Vancouver area in Canada.

This user from Canada has been scanning all University network during the three days investigated. He could have found an important vulnerability in this host and explored it, turning our workstation into a zombie that can be used to do further attacks on the University Network and other places.

2.1.3. Scan Analysis

2.1.3.1. Big Picture

There are many scanning activities on the network, especially coming from inside. We can see below (section 2.2.3.2.2) that the top 4 sources of scanning were insiders. It should not

be this way. There is no reason for our internal hosts to be scanning other internal hosts or external hosts after ports or vulnerabilities. Quoting Mike Poor: "If you can't play nice on my network, you can't play on my network at all!".

Analysis of the scanning activities provided to be a very good support to all alerts, since most of the attacks were proceeded by a scan.

2.1.3.2. Top Talkers

Below are the top 5 scanned and scanners identified. This was done using the Snort Preprocessor, which is not a Snort Rule, but is capable of counting the number of connections from the same port, IP or to an IP in a specific time. When a certain number of events occur, it triggers a scan alert, and logs by our parameters that we configure in the snort.conf file.

2.1.3.2.1. Top Destinations

Below are the top 5 scanned IPs identified.

Scans	
Destination IP	**Scans**
MY.NET.84.250	2521
MY.NET.12.2	231
MY.NET.25.11	216
24.42.148.105	155
68.71.102.23	155

I would have to have more information on these hosts to do further analysis. With the info I had, this only proved useful to support alert analysis.

2.1.3.2.2. Top Sources

Below are the top 5 scanners identified.

Scans	
Source IP	**Scans**
MY.NET.98.31	6185
MY.NET.97.136	5377
MY.NET.242.174	5349
MY.NET.98.150	3058

Scans	
Source IP	Scans
80.14.80.158	2521

I would have to have more information on these hosts to do further analysis. With the info I had, this only proved useful to support alert analysis.

2.1.4. Out of Spec Packets Analysis

2.1.4.1. Big Picture

The total number of OOS packets in the network was considerably small compared to the number of alerts (3852 oos packets compared to a total of 109.684 alerts). If we assume that only a part of the network traffic is related to malicious activity and thus result an alert on the IDS, we can estimate a bigger number of total traffic.

Doing further analysis we could verify that 57,62% of the packets were originated from only ten hosts.

Most of the scan entries for these highly active source addresses showed the SYN/FIN combination flags set in the TCP header. This looks very much like a technique employed to increase the odds of gaining access to a site and eliciting a reply from the scanned host. In almost all cases the scan was conducted once per source-destination pair. These packets should be blocked at the firewall, as they represent no valid TCP communication attempt.

The analysis of these packets had proven to be very helpful supporting the alert analysis, but the analysis of the oos packets in particular was not a big source of identifying many attacks.

The same happened in Mike Bell´s practical, that can be found on the address below:
http://www.giac.org/practical/Mike_Bell_GCIA.doc

2.1.4.2. Top Talkers

Below are the top 5 destination and source hosts of Out-of-Specification packets

2.1.4.2.1. Top Sources

Below are the top 5 originators of OOS packets.

OOS

Source IP	Packets
148.64.169.5	319
68.164.35.154	233
61.114.222.241	123
213.98.16.183	120
210.253.215.113	110

2.1.4.2.2. Top Destinations

Below are the top 5 destination of OOS packets.

OOS	
Destination IP	Packets
MY.NET.220.106	374
MY.NET.70.225	324
MY.NET.207.2	269
MY.NET.211.106	219
MY.NET.6.47	204

2.2. Three Most Important Attacks

The following alerts were identified as Events of Interest, and thus detailed below:

2.2.1. Tiny Fragments - Possible Hostile Activity

2.2.1.1. Description of Detect

Some traffic from host 24.112.169.243 on the Internet to MY.NET.201.62 generated the alert "Tiny Fragments - Possible Hostile Activity" on the 17th. It was probably a result by a Preprocessor Plugin that states that any fragmented packet with a size too small (payload < 25 bytes) is one packet that should be examined. I could find several sources on the Internet, including snort.org, supporting the fact that no modern equipment needs to fragment a datagram in a size smaller than 512 bytes.

Also, other attacks from internal and external sources generated simmilar alerts on the sensor.

As tiny fragmentation may be an attempt to elude our network defenses to do numerous other forms of attacks (decoy attack), like Buffer Overflows, Denial of Service and other, I decided to look further into this attack.

These detects came on all days at very specific time periods. They didn't last more than 5 minutes. They also were from different sources to different destinations. No attacker repeated the attack nor the same target was attacked twice from different sources.

There were no source or destination ports registered on the first fragment. We could also notice some scan being done right before the attack. Usually was a Null scan after a Nmap Stealth Scan. We can se better in the section below:

```
02/17-22:46:08.634090  [**] spp_portscan: portscan status from
24.112.169.243: 1 connections across 1 hosts: TCP(1), UDP(0) STEALTH [**]
02/17-22:30:02.052624  [**] Null scan! [**] 24.112.169.243:0 ->
MY.NET.201.62:0
02/17-22:30:02.052644  [**] Tiny Fragments - Possible Hostile Activity
[**] 24.112.169.243 -> MY.NET.201.62
02/17-22:30:03.316935  [**] Null scan! [**] 24.112.169.243:0 ->
MY.NET.201.62:0
02/17-22:46:17.623065  [**] spp_portscan: portscan status from
24.112.169.243: 2 connections across 1 hosts: TCP(2), UDP(0) STEALTH [**]
02/17-22:46:17.687461  [**] spp_portscan: portscan status from
MY.NET.98.167: 6 connections across 6 hosts: TCP(0), UDP(6) [**]
02/17-22:46:23.887296  [**] spp_portscan: portscan status from
MY.NET.98.167: 3 connections across 3 hosts: TCP(0), UDP(3) [**]
02/17-22:30:05.961311  [**] Tiny Fragments - Possible Hostile Activity
[**] 24.112.169.243 -> MY.NET.201.62
02/17-22:30:05.961430  [**] Tiny Fragments - Possible Hostile Activity
[**] 24.112.169.243 -> MY.NET.201.62
```

2.2.1.2. Reason why Detect was Selected

If someone is trying to elude our IDS is that he or she is not trying to do good to our network. This is could be also used to make noise a cover the main attack, since modern IDSs can detect these patterns and this particular alert came in very noisy.

I was intrigued by this traffic, and because of the lethality of this attack (can cover lots of other malicious activities) I decided to look into it a little further.

2.2.1.3. Detected was generated by

Thanks to Andrew B.'s homepage I was able to figure out the information below. Andrew B page is at http://www.dpo.uab.edu/~andrewb/snort/snortdoc/preplugin.html.

This detected was generated with the minfrag preprocessor. It checks for fragmented packets and if the packet is a fragment and its size is less than or equal to the threshold value then it generates the alert: "Tiny Fragments - Possible Hostile Activity" and also logs the packet. It also toggles the detection bit for this packet so that it is not passed to the detection engine.

It was not possible to dtermine the threshold value, thus its default value is 25 bytes.

2.2.1.4. Probability the Source Address was Spoofed

There are two possibilities here. Since this was a very noisy attack, the first one covers the possibility it was a decoy used to draw attention while a more sneaky attack was conducted. In this case it is natural that the address would be spoofed, so it would make the attack-response more difficult and draw the attention of the main target.

Also, we should consider the possibility of covert channel. In this case the probability of spoofing of the source address is low, since the attacker is trying to cover his communication with the target host, and he is probably trying to communicate with it, thus needing to receive answers back.

2.2.1.5. Attack Mechanism

Tiny fragmentation is used to avoid detection on filtering devices and IDS systems which do not do packet re-assembly. Fragments can thus at times avoid detection, where the entire packet may not.

Many IDSs are known to have issues regarding the reassembly of IP fragments, and could miss an attack carried over such means. Firewalls suffer from the same issues, and can be tricked into allowing packets through that should normally be rejected. Furthermore, there is a small history of OS issues related to unorthodox fragmentation.

Historically, handling of fragmentation has been a problem in both IP stacks and the IDS systems. While the limited number of attacks based on fragmentation are easily picked up by anomaly or signature based systems, IDSs which fail to properly reassemble fragments can miss any fragmented attack. Firewalls have often proved susceptible to fragmented TCP or UDP headers, allowing traffic which should have been filtered to pass through.

An attacker may pass a fragment containing a TCP/UDP header which is allowed to pass through a firewall, then follow this up with a fragment which overwrites the previous headers, but is allowed due to poor connection tracking.

An attacker may fragment an exploit, so that it is not detected by IPS nor filtered by IPS products.

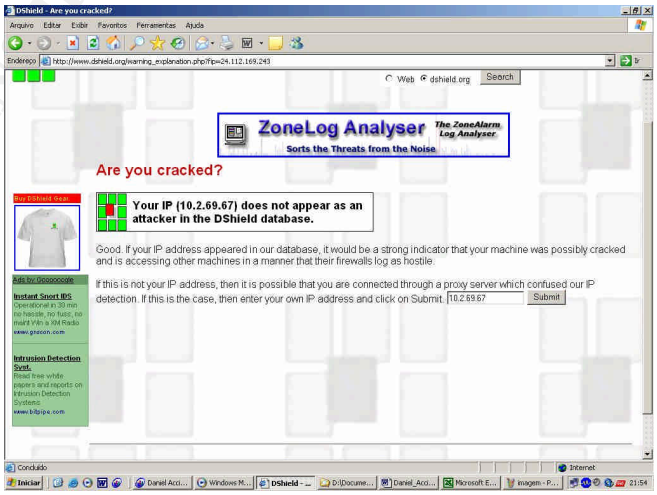
Specially in this attack the attacker uses small fragments to force some of the TCP header information into the next fragment. This may produce a case whereby the TCP flags field is forced into the second fragment and filters that attempt to drop connection requests will be unable to test these flags in the first octet thereby ignoring them in subsequent fragments.

This attack can be used to circumvent user-defined filtering rules. The attacker hopes that a filtering router will examine only the first fragment and allow all other fragments to pass.

This attack can be prevented at the router by enforcing rules, which govern the minimum size of the first fragment. This first fragment should be made large enough to ensure it contains all the necessary header information.

2.2.1.6. Correlations

No previous alerts associated with the source hosts were recorded on the IDS logs, but portscans and SMB Wildcard alerts. A search through the Dshield database did not find any previous attacks in their logs associated with the source host.



Peter van Oosterom (http://www.giac.org/practical/GCIA/Peter_Van_Oosterom.pdf) submitted a detect involving 20 byte fragments with an ID of 0. This was part of a network scan. His traffic had both the more fragments and don't fragment bit set which is not the case with the above trace. It is possible that the above trace is part of the scan and the don't fragment bit did not get set for some reason. However, there are no other packets with a fragment ID of 0 in the log file for the day in question.

I could find no correlations for this exact traffic.

2.2.1.7. Evidence of Active Targeting

Several alerts were generated in this attack, along with some Null Scans, directed to the same internal host for a specific period of time. Also, I could find several different portscans from this attacker to several hosts and some SMB Wildcard alerts.

Although I think the attacker actively targeted the internal host MY.NET.201.62, he spent some time looking for his victim before striking.

2.2.1.8. Severity

The severity of this attack is somewhat high due to the high lethality rating and little effective countermeasures.

The lethality of the attack is high. It can cover other very serious compromises and thus is important. I'm assuming we don't have proper countermeasures because the attack seemed to be successful. We cannot define the criticality of the hosts attacked since we don't have enough information on them. I would arbitrary estimate it as medium though.

This way I'm assuming the following calculation in a 1-5 grade:

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

Severity = (3 + 4) – (1 + 2) = 4

2.2.2. MY.NET.222.174

2.2.2.1. Description of Detect

MY.NET.222.174 was a host that called my attention. It has several alerts recorded where it acts like a client of source of Red Worm infection trojan, it originates CGI Null Byte and Unicode attacks to different web servers on the internet, and do some portscans on other servers. It appears as some "smart user" that is trying to use our network resources to

perform attacks. It would not be a big problem, since we could locate this machine and shut it down, reprehending the user, but I found other interesting traffic.

Some packets originating from the source 212.179.105.210 on different ports target port 4662, which is common port for e-Donkey P2P application. The traffic happens every day at the same time and this IP is from a cable user from Israel which is in the Administrator Watchlist:

```
inetnum:     212.179.96.0 - 212.179.106.255
netname:     CABLES-CONNECTION
descr:       CABLES-CUSTOMERS-CONNECTION
country:     IL
[…]
role:        BEZEQINT HOSTMASTERS TEAM
address:     Bezeq International
address:     40 hashacham st.
address:     Petach Tikva 49170 Israel
```

This watchlist is a customized alert configured to register a log entry each time a computer with specific addresses accesses UMBC's network.

I also found some out-of-spec packets with strange payload in it. An example is detailed below:

```
02/18-09:54:17.103069 148.63.237.176:1025 -> MY.NET.222.174:4662
TCP TTL:113 TOS:0x0 ID:57405 IpLen:20 DgmLen:113 DF
****P*** Seq: 0x2717A0A  Ack: 0x0  Win: 0x2000  TcpLen: 20
E3 44 00 00 00 01 10 28 8D 51 58 BE CA DC 37 E0  .D.....(.QX...7.
68 31 E6 09 88 43 9A 94 3F ED B0 36 12 02 00 00  h1...C..?..6....
00 02 01 00 01 10 00 66 6F 75 72 70 61 73 74 6D  .......fourpastm
69 64 6E 69 67 68 74 03 01 00 11 2D 00 00 00 D9  idnight....-....
E3 13 28 BA 1B D8 41 C7 A6                       ..(...A..
```

We can see a clear string "fourpastmidnight" that could mean a remote command through a P2P known port. I understand that the application that is using this P2P port might not be a P2P application, but a Trojan of some kind. I could not confirm that, but it is a possibility.

My analysis is that this packet is out-of-spec because the ACK flag is not set with the PUSH flag. In a TCP communication, the sender sends data with both ACK and PUSH flags set, and the receiver answers back with the ACK flag only for all packets. We can see "Ack:" field with the "0x0" value, meaning it is not set, thus being an OOS packet.

With all this evil traffic generated from the MY.NET host, and this other strange traffic coming toward it from a distant internet computer in Israel, a suspicious country because of the lack of regulation and actions on the Internet issues, I though that this could be a case of misuse of an internal host.

The University is liable for actions originating from its network, so should do whatever possible to stop attacks. This is even more important if UMBC infrastructure is being used as part of coordinated attacks to other institutions/websites on the Internet.

2.2.2.2. Reason why Detect was Selected

As already said, the University is liable for actions originating from its network, so should do whatever possible to stop attacks. This is even more important if UMBC infrastructure is being used as part of coordinated attacks to other institutions/websites on the Internet.

Identify what is this attack is important for preventing legal problems, as well as impacting availability of the network (coordinated DoS attacks usually consumes a lot of bandwidth, even if you are the source). Also, a host infected with a trojan is a door for information leakage and trampoline for other attacks.

2.2.2.3. Detected was generated by

Snort IDS. There were several different snort alerts on this attack. The most important were:

```
- Red Worm:

02/17-14:03:45.551644  [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.222.174:4634 -> 24.50.140.77:65535
```

Detected by:

I coudn´t find the original rule, so I estimate the attack to be detected by something simillar to this:

```
alert tcp any any -> any 65535 (msg:" High port 65535 tcp - possible Red
Worm - traffic";)
```

and

- Watchlist:

```
02/17-03:00:53.488594  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.105.210:4876 -> MY.NET.222.174:4662
```

Detected by:
It is a custom rule. The exact rule could not be found, but I expect it to be like this:

```
alert tcp 212.179.0.0/16 any -> $HOME_NET any (msg:" Watchlist 000220 IL-
ISDNNET-990517";)
```

2.2.2.4. Probability the Source Address was Spoofed

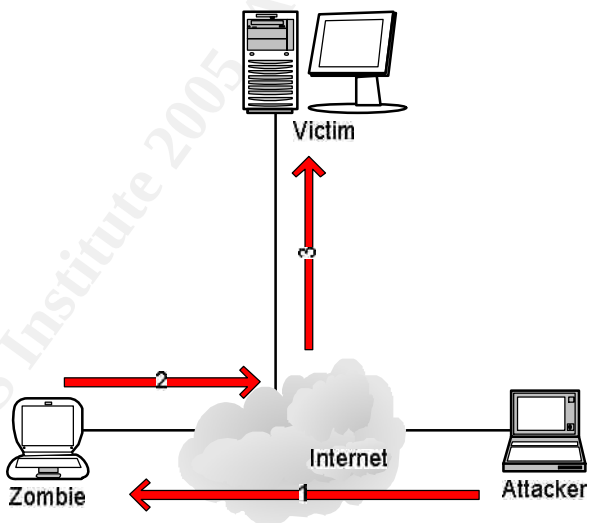
There are two addresses to analyze in this attack. The first one is the Israeli Internet address. It is probably not spoofed because its attacks are aimed toward controlling our internal machine, so it has to receive communication back.

Also, the attacks originating from our internal networks were all "controlling attacks", I mean, they were aimed to control another hosts, like the Unicode and Red Worm traffic. So I believe it is not spoofed as well.

2.2.2.5. Attack Mechanism

The idea of the attack is to compromise a remote machine in order to make it difficult for the victim to trace the original attacker back. It does this infecting the remote machine using a virus, Trojan or installing a remote agent through a known vulnerability. After that, he gains control over it and can lunch attacks from the controlled host. This host can also be called zombie.

Below follows a diagram of such attack:



0) Even before the attack begins, the attacker probes the Internet for prospected victims, by fingerprinting operational systems, portscaning hosts and such.

1) After it finds one, the first thing that happens is that the attacker accesses the zombie (which is still clear, but with known vulnerabilities) and it does a deeper scan scans for known services, vulnerabilities and ways to explore them. On this phase it compromises the system using one of these vulnerabilities, does privilege escalation and install a Trojan or backdoor for later access and use.

2) The second phase of the attack is done remotely by the attacker. It commands the zombie to scan the internet for other victims. It could also look for a specific target and actively scans for known vulnerabilities without the possibility of being identified.

3) The third phase is the attack itself, where the attacker exploit its target and gets what it wants.

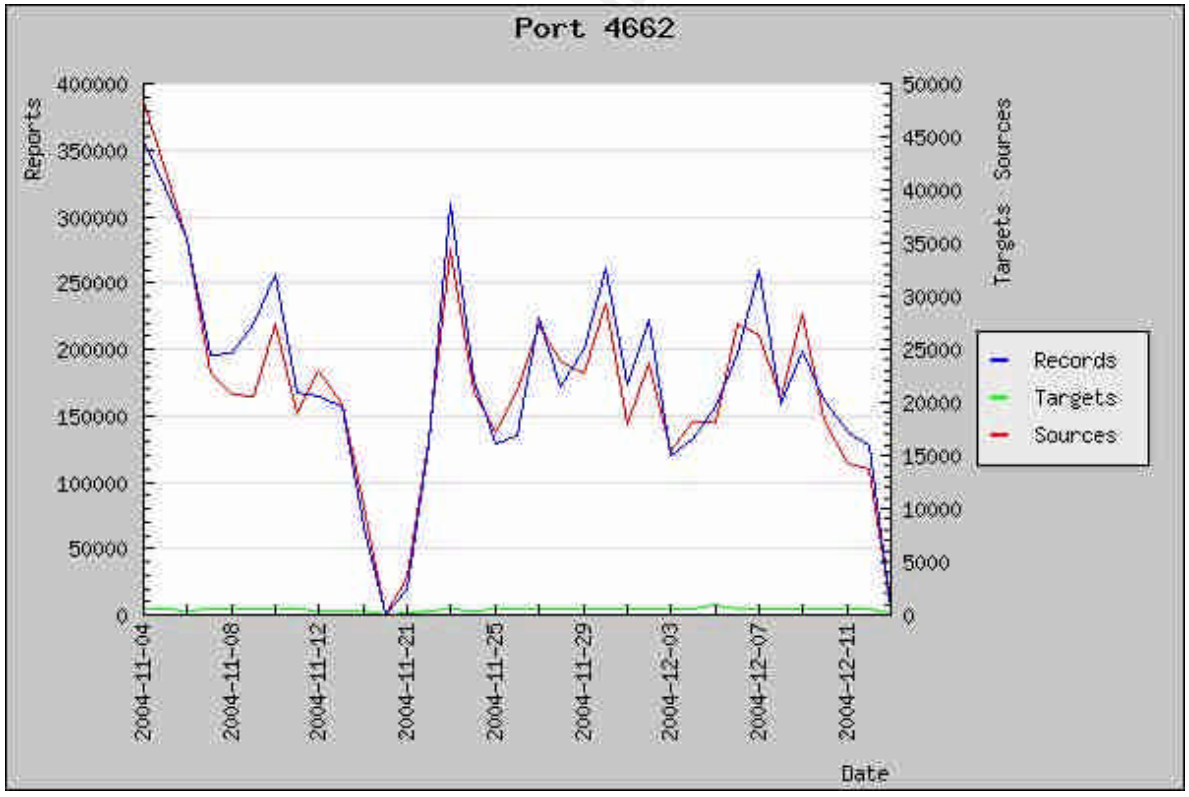
The only hope for the victim is to collaborate with the administrator of the zombie in order to access its logs and try to traceback the attacker. Many times this is not possible because the attackers usually chooses undocumented and poorly managed networks for zombie infection, like Universities.

This method is also used for creating botnets and performing Denial of Service Attacks.

2.2.2.6. Correlations

Since there are a lot of attacks to correlate, I decided to focus on the most important: the 4662 e-donkey 2000 server port.

Accessing dshield website I could find this information:



It shows that port 4662 is scanned very frequently on the Internet. This happens because it is used as a famous P2P program. When a network has DHCP, the P2P clients don't know

that the address lease changed, and keep trying to connect to old addresses while the computers that holds the P2P software already changed IPs.

I could not find any Trojans associated with this port. Besides, I could find a table at http://www.edonkey2000.com that showed that more than 1100 bugs were reported in their last version:

Project	Open	Fixed	Not a bug	Won't Fix	Deferred	Works for me	Duplicate	Total
Download center	56	7	6	0	1	1	1	72
eDonkey2000	841	122	104	17	12	17	6	1119
eTree	0	1	0	0	0	0	0	1
Installer	6	0	0	0	0	1	0	7
Kdrive	42	37	3	0	1	2	1	86
MetaMachine PluginSDK	5	2	1	0	0	0	0	8
Overnet	1676	223	174	22	14	58	95	2262

1100 bugs can mean some vulnerabilities that can be exploited, especially with a program that reads, writes and executes files on the computer. I didn't have access to their patching and support website, since I had to be registered to do so, but I checked on bugtrack mailing list and unfortunately I couldn't find any vulnerabilities reported.

Nothing was found either on other students practicals.

2.2.2.7. Evidence of Active Targeting

Several alerts were generated in this attack, along with some Null Scans, directed to the same internal host for a specific period of time. This attack was not recorded for any other host in the network. Besides that I could find several different portscans from this attacker and some SMB Wildcard alerts.

Although I think the attacker actively targeted the internal host MY.NET.201.62, he spent some time looking for his victim before striking.

2.2.2.8. Severity

The severity of this attack is surely high due to the high lethality rating and little effective countermeasures.

The lethality of the attack is high. It gives the attacker a very strong position to compromise UMBC security, I'm assuming we have somewhat proper countermeasures to avoid this attack, but they are not properly configured because the attacker can access a host directly

in the University network, We cannot define the criticality of the hosts attacked since we don't have enough information on them. I would arbitrary estimate it as small, since it seems to be a regular workstation.

This way I'm assuming the following calculation in a 1-5 grade:

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

Severity = (2 + 5) – (1 + 3) = 3

2.2.3. Port 135

2.2.3.1. Description of Detect

Traffic was detected with the source and destination addresses outside the network. This could indicate one of the two things: routing problems or address spoofing. Since either are big problems, snort detects it as traffic that could compromise the security of the network. The packets were directed to a single internet IP (216.209.164.171), which is a computer from Bell Comm in Canada, on port 135 and they started about 21:45. It lasted for about 7 minutes until 21:52.

This port is the Microsoft DCE Locator service also known as end-point mapper and it is used on Microsoft systems for DHCP, DNS and WINS services. It works just like Sun RPC portmapper, except that end-points can also be named pipes. Microsoft also relies upon DCE RPC to remotely manage services.

The traffic was very fast (about 5 packets/sec) and we could notice different source addresses. These addresses were between the 171.165.35.x subnet to the 171.165.182.x.

We also could notice that as the time passed, these source IPs were incrementing in 2 or 3, and all the source ports were in the 1000-2000 range.

Below is a transcript of part of the traffic with these characteristics:

```
02/19-21:45:01.668488  [**] TCP SRC and DST outside network [**]
171.165.35.69:1115 -> 216.209.164.171:135
02/19-21:45:01.668497  [**] TCP SRC and DST outside network [**]
171.165.35.70:1025 -> 216.209.164.171:135
02/19-21:45:01.717426  [**] TCP SRC and DST outside network [**]
171.165.35.76:1389 -> 216.209.164.171:135
02/19-21:45:01.717445  [**] TCP SRC and DST outside network [**]
171.165.35.77:1441 -> 216.209.164.171:135
02/19-21:45:01.717577  [**] TCP SRC and DST outside network [**]
171.165.35.78:1103 -> 216.209.164.171:135
02/19-21:45:02.741606  [**] TCP SRC and DST outside network [**]
171.165.35.125:1877 -> 216.209.164.171:135
```

```
02/19-21:45:02.789522  [**] TCP SRC and DST outside network [**]
171.165.35.133:1254 -> 216.209.164.171:135
02/19-21:45:02.838182  [**] TCP SRC and DST outside network [**]
171.165.35.140:1204 -> 216.209.164.171:135
02/19-21:45:02.869472  [**] TCP SRC and DST outside network [**]
171.165.35.147:1367 -> 216.209.164.171:135
02/19-21:45:03.077403  [**] TCP SRC and DST outside network [**]
171.165.35.183:1884 -> 216.209.164.171:135
02/19-21:45:03.110277  [**] TCP SRC and DST outside network [**]
171.165.35.189:1196 -> 216.209.164.171:135
```

I searched the 45 minutes prior to the attack for detects of internal systems compromised by external attackers, but I couldn't find any evidence of remote attackers gaining access to internal hosts in the network that could have done that.

2.2.3.2. Reason why Detect was Selected

This attack looks like a Denial of Service attempt against 216.209.164.171. I believe in this because of the large number of packets in a small timeframe directed to the same host. This could have happened in two ways: A remote attacker took control over one host of our network or the attack was started by one of the University users.

In either case this is a big problem. The first would require an immediate action in order to prevent leak of information and compromise of the network. The other requires immediate disciplinary action to educate other users not to do the same, thus justifying further investigation.

2.2.3.3. Detected was generated by

The following Snort Rule:

alert tcp $EXTERNAL_NET any -> $EXTERNAL_NET any (msg:" TCP SRC and DST outside network ";)

This rule means that any packet that passes over the sensor that does not have an address from the INTERNAL_NET network either on the source or destination fields should be registered as an alert.

2.2.3.4. Probability the Source Address was Spoofed

The probability of spoofing of the source address is very high! As the packets appear to be crafted by an automated process (incrementing IP addresses, source ports in a specified range, etc…) there is a high probability that this is a DoS attack. The architecture of this attack requires that the packets do not return to the source, since they would also DoS the attacker if they did return.

My point of view is that the attacker IP was not real.

2.2.3.5. Attack Mechanism

The attack mechanism is a machine flooding another host with RCP TCP packets with a spoofed source address. Port 135 is a very well known port for performing attacks on the net. Several worms, especially the Blaster worm, used this service as vector for contamination.

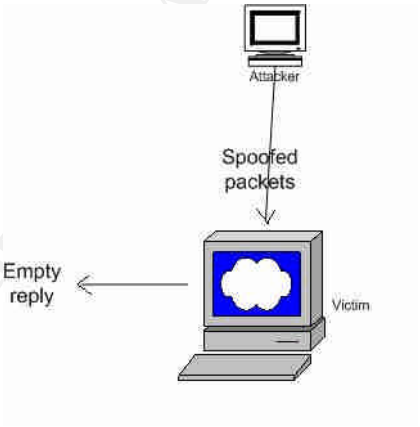
Also, there are vulnerabilities concerning DoS attacks that are able to consume a server total processing time without using too much band. They were discovered and made available since 1998 affecting systems from Windows NT 4 to 2000:

http://www.winnetmag.com/WindowsSecurity/Article/ArticleID/9252/WindowsSecurity_9252.html

http://support.microsoft.com/default.aspx?scid=kb;en-us;193233&sd=tech

Even being a DoS attack that takes advantage over common RPC vulnerabilities, or a worm that is trying to insistently infect a host, these attacks clearly uses spoofed packets not to be identified.

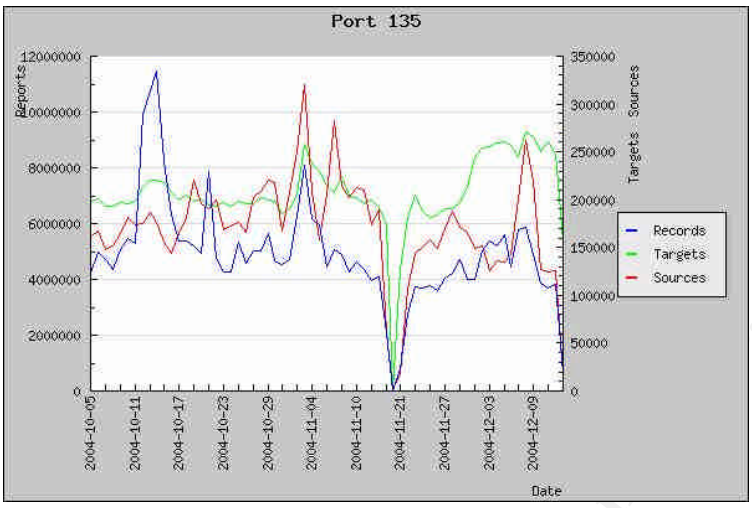
Below we can see an attacker initiating a DoS by sending a spoofed RPC packet to its victim. The victim responds to an empty or invalid address, and hangs.



2.2.3.6. Correlations

A lot is seen on the internet on RPC vulnerabilities. Most of the things are related to viral infections, and not this traffic.

Port 135 is one of the most searched ports on d-shield database:



I was able to find vulnerability reports stated on the section above and a report from an University in the Chicago area that suffered from this attack:

http://nsit.uchicago.edu/alert/port-135.html

I've also managed to do a small search for the 216.209.164.171 IP and this is what I found:

```
Bell Canada BELLCANADA-4 (NET-216-208-0-0-1)
                                  216.208.0.0 - 216.209.255.255
HSE (Bell Nexxia) HSE002-CA (NET-216-209-152-0-1)
                                  216.209.152.0 - 216.209.167.255
```

Bell Canada is a telecommunication company that provides Internet Services for its users. The target may be a company server or just a simple home user that got in conflict with a hacker on a chat room.

2.2.3.7. Evidence of Active Targeting

Several alerts were generated in this attack, but they were all the same time, in sequential hours and directed to the same host.

The target was actively target. The attacker knew what it was after.

2.2.3.8. Severity

The severity of this attack is not as high as the others due to the medium lethality rating and somewhat effective countermeasures.

The lethality of the attack is medium. Although it can DoS a remote system, it probably can't compromise UMBC network. The main problem here would be legal issues, where the university could be charged for the attacker actions. We cannot define the criticality of

the hosts attacked since we don't have enough information on them. I would arbitrary estimate it as small, since it seems to be a regular workstation.

This way I'm assuming the following calculation in a 1-5 grade:

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

Severity = (2 + 3) – (1 + 4) = 0

2.3. Recommendations

Appropriated network architecture, IDS tuning and sensor placement, Firewall rules, virus scanners, use of application-layer proxies and patch management would be the most important countermeasures for the University.

The University should consider migrating it network to a reserved address space. The use of a 10.x.x.x/8 subnet for example prevents external users to direct access our internal resources, and can be managed at the firewall with NAT (Network Address Translation).

Firewall rules need to be tightened down. It should be configured to "deny-what-is-not specifically-allowed". If there is no valid business reason to allow connections to port 515 from the internet, then this port and others like it should be blocked by the firewall.

There are several alerts that are related to different worms and viruses. By adding some anti-virus scanners, both on the workstations and the gateways, this would help cut-down on the virus problem within the network.

Better sensor placement should be done. It is important to understand what attacks and probes are being done to my network, So a sensor outside the firewall is important. We also want to see what went through, so another sensor inside the firewall might be important. Central management of these alerts, as well as better rule configuration is essential for adequate protection. The University might want to look for SIM (Security Information Management) Systems in order to consolidate the various logs and alerts.

The use of proxies serves as a good point of control of the network. No traffic should go inbound or outbound without being compared to an application level rulebase that allows only legitimate protocols in and out of the network.

Policy should be in place and awareness should be conducted with the users. There is no justification for so many inside activity as identified above. Users should be reprehended when doing bad actions and even loose the access to the network. Management support helps the Admins to achieve that.

And lastly patch management will help close a lot of the vulnerabilities that many of these worms and viruses target. My recommendation is to treat computers not managed by the University as untrusted. Force these computers to use some type of end point security where they must logon to some type of server where their machine is checked for current patches and A/V definitions. If the computer is not up to date, force the user to patch the machine before access is granted to University resources, in most cases, the University's Internet connection.

3. Analysis Process

The initial part of the Analysis process was to evaluate the number and type of events that I had to study.

The requirement of the Practical was to do the process with only three files of each type. Even thinking that only a 7-day analysis (full week) would really show a pattern of the network, I decided not to push the envelope and stick with the requirement. I put in the report this was a University requirement and moved on.

I choose the files based on their size. I did not want to get files too big because as larger they were, more difficult they would be to manage (more records!). I also though that if they were too small I would end up with lack of information and I would have to make lots of guessing in the report. That would not be good.

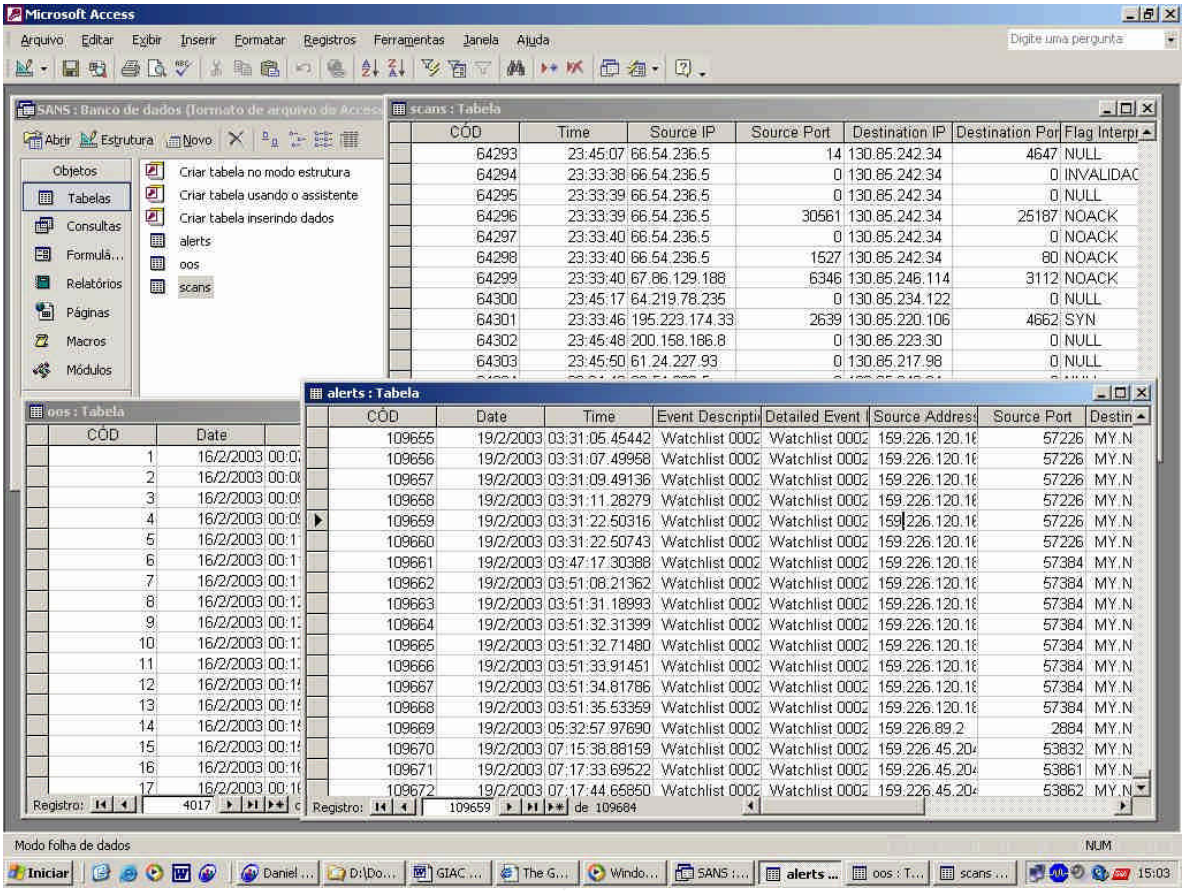
In the end I choose about 4Mb alert files, 500Kb oos files and 2Mb scan files. These represented 109684 alerts, 4017 oos packets and 64319 scans.

My first try was to follow advice from other students practicals to organize the files. I started looking for how to user pearl scripts, grep commands, shell scripts and other tools in Windows, since I'm not very used to Unix.

I found very good ideas on Tim Kroeger´s GCIA Practical (http://www.sans.org/practical/Tim_Kroeger_GCIA.pdf). He used shell scripts with grep, cat and other UNIX commands to do the "dirty work". Although these were not very "elegant", they gave me the impression to be very effective. Unfortunately I could not go very far with those because I got stuck with installing Unix and organizing the files properly for edition.

After that I decided I had to find a good solution for Windows. I gathered all my Ms Office experience and started working with Excel 2003. The first problem was that most of the files were too big to fit in a worksheet. The Excel's limit is 65536 records, and I had tables with more that 100K records. I noticed by this time I had to work with a database.

I imported all the sheets into three Ms Access tables and started working with simple sql commands. I could do all queries I wanted with technology I could understand. That sounded good with just one problem. I had to manipulate the files a little bit to convert them in tables.



MS Access 2003 with the Alerts, OOS and Scans tables

The first and more obvious problem to be resolved was the scan and alert files. Each line was an entry and, before throwing the data on Access I divided the three files in ten using notepad. I used and excel macro to divide the one column in many using the "Data-to-Columns" feature and fetched the ten files in Excel. From the almost 18K entries I lost only 5, which I considered to be IDS output format problems, thus not events of interest.

The second and bigger problem was the oos files. They were not one entry per line, and the packet payload registered in the file was not standardized.

First I decided that the file payload information was not important to be put in the Access table. I would search the text file using WordPad's "find" if I thought it needed to be inspected for specific content. After that I looked for a tool that would allow me to print specific records that matched a search criteria and creates a file that could be worked using the "Data-to-Columns" feature in Excel.

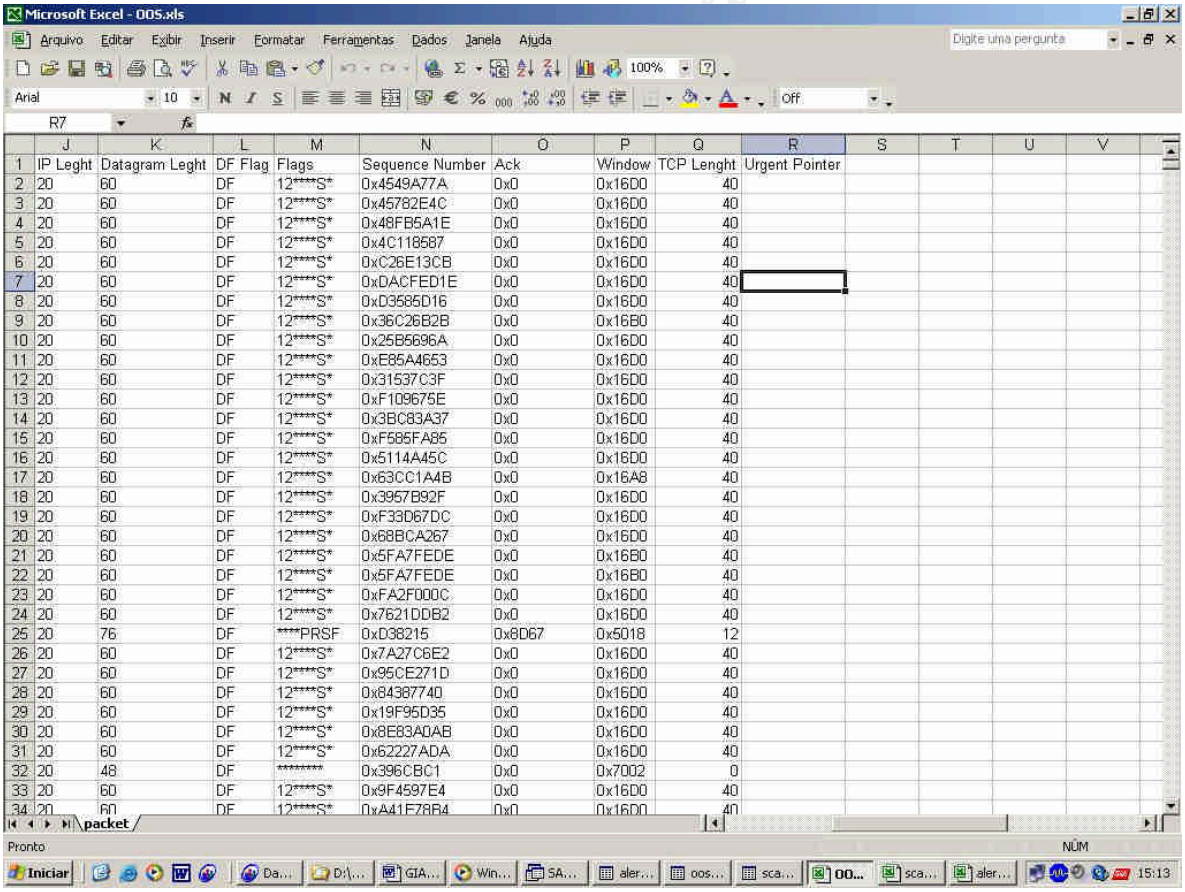
I found in Ian Eaton's GCIA Practical a set of awk commands that proven to be very useful! Searching the net I found that awk had a MS Windows version that not only worked on XP, but was very easy to install. It's called gawk (GNU AWK) and I downloaded it from http://gnuwin32.sourceforge.net/packages/gawk.htm. A very easy-to-go manual is available online at http://www.gnu.org/software/gawk/manual/gawk.html and it was used by me to do all the work needed on this project.

Using gawk I created a small program that extracted the lines that were common for all oos packets and outputted them for three separate xls files. Below is the small and very simple awk program that was used:

```
BEGIN { print "Daniel Accioly Rosa AWK script for preparing OOS files to
work in excel - GCIA Practical!" }
# Prints the first three lines of the OOS packet, thus eliminating the
packet payload info and
#keeping only the relevant information in three separate xls files

/->/ { print $0 > "packet.xls" }
/TCP TTL/ && /TOS/ { print $0 > "IP.xls" }
/Seq/ && /Ack/ || /Frag/ { print $0 > "tcp.xls" }
```

Later I transformed those files in one Excel worksheet and imported the records in Access. Bingo! I got the information in only one place where I could query using sql. Now I could go on with the Analysis.



Microsoft Excel - OOS.xls

	J	K	L	M	N	O	P	Q	R
1	IP Leght	Datagram Leght	DF Flag	Flags	Sequence Number	Ack	Window	TCP Lenght	Urgent Pointer
2	20	60	DF	12****S*	0x4549A77A	0x0	0x16D0	40	
3	20	60	DF	12****S*	0x45782E4C	0x0	0x16D0	40	
4	20	60	DF	12****S*	0x48FB5A1E	0x0	0x16D0	40	
5	20	60	DF	12****S*	0x4C118587	0x0	0x16D0	40	
6	20	60	DF	12****S*	0xC26E13CB	0x0	0x16D0	40	
7	20	60	DF	12****S*	0xDACFED1E	0x0	0x16D0	40	
8	20	60	DF	12****S*	0xD3585D16	0x0	0x16D0	40	
9	20	60	DF	12****S*	0x36C26B2B	0x0	0x16B0	40	
10	20	60	DF	12****S*	0x25B5696A	0x0	0x16D0	40	
11	20	60	DF	12****S*	0xE85A4653	0x0	0x16D0	40	
12	20	60	DF	12****S*	0x31537C3F	0x0	0x16D0	40	
13	20	60	DF	12****S*	0xF109675E	0x0	0x16D0	40	
14	20	60	DF	12****S*	0x3BC83A37	0x0	0x16D0	40	
15	20	60	DF	12****S*	0xF585FA85	0x0	0x16D0	40	
16	20	60	DF	12****S*	0x5114A45C	0x0	0x16D0	40	
17	20	60	DF	12****S*	0x63CC1A4B	0x0	0x16A8	40	
18	20	60	DF	12****S*	0x3957B92F	0x0	0x16D0	40	
19	20	60	DF	12****S*	0xF33D67DC	0x0	0x16D0	40	
20	20	60	DF	12****S*	0x68BCA267	0x0	0x16D0	40	
21	20	60	DF	12****S*	0x5FA7FEDE	0x0	0x16B0	40	
22	20	60	DF	12****S*	0x5FA7FEDE	0x0	0x16B0	40	
23	20	60	DF	12****S*	0xFA2F000C	0x0	0x16D0	40	
24	20	60	DF	12****S*	0x7621DDB2	0x0	0x16D0	40	
25	20	76	DF	****PRSF	0xD38215	0x8D67	0x5018	12	
26	20	60	DF	12****S*	0x7A27C6E2	0x0	0x16D0	40	
27	20	60	DF	12****S*	0x95CE271D	0x0	0x16D0	40	
28	20	60	DF	12****S*	0x84387740	0x0	0x16D0	40	
29	20	60	DF	12****S*	0x19F95D35	0x0	0x16D0	40	
30	20	60	DF	12****S*	0x8E83A0AB	0x0	0x16D0	40	
31	20	60	DF	12****S*	0x62227ADA	0x0	0x16D0	40	
32	20	48	DF	********	0x396CBC1	0x0	0x7002	0	
33	20	60	DF	12****S*	0x9F4597E4	0x0	0x16D0	40	
34	20	60	DF	12****S*	0xA41E78B4	0x0	0x16D0	40	

Oos entries after selected with gawk, imported and separated in Excel

Peter Van Oosterom did on his GCIA Practical an excellent group of tables to give a good overview of what the files had to say about the attacks on the network. I decided to follow his idea and organized the "Big Picture" session with the same kind of info he did.

After that I only parsed through the information using Excel, Access, gawk, Wingrep and the original files to get the results.

The hardware used were three computers:

1) Athlon Thinderbird 850Mhz with 392 MB RAM and 80 GB HD – Windows XP
2) Pentium III 650 Mhz with 256 MB RAM and 40GB HD – Windows XP
3) Notebook Compaq Armada E500, Pentium II 350 128 MB RAM and 10 GB HD – Windows 2000

4. References

Most of the references are through the text, they were also placed here.

- Mike Bell GCIA Practical
http://www.sans.org/y2k/practical/Mike_Bell_GCIA.doc

- Glen Larratt GCIA Practical
http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html

- Al Evans GCIA Practical
http://www.sans.org/y2k/practical/Al_Evans_GCIA.doc

- Andrew G. Siske, Jr. GCIA Practical
http://www.sans.org/y2k/practical/Andy_Siske_GCIA.htm

- Shong Chong GCIA Practical
http://www.sans.org/y2k/practical/Shong_Chong_GCIA.doc

- Tim Kroeger GCIA Practical
http://www.sans.org/practical/Tim_Kroeger_GCIA.pdf

- Walter Claire
http://www.giac.org/practical/GCIA/Wouter_Clarie_GCIA.pdf

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

- The Internet Engineering Task Force, All the RFC's, October 1, 2002
http://www.ietf.org

- Stephen Northcutt, Judy Novak, Network Intrusion Detection An Analyst's Handbook, 2nd Edition September 2000, October 31, 2002

- White Hats Intrusion detection database, Jan 2003
http://www.whitehats.com

- SANS, past student practical's, Jan 2003
http://www.giac.org

- Microsoft Website
http://www.microsoft.com

- The Snort Webpage
http://www.snort.org

- Dshield
http://www.dshield.org

- Arin Whois Database Search
http://www.arin.net/whois/

- Shon Harris, CISSP All-in-One Exam Guide, Second Edition (All-in-One) Second Edition

- Marty Roesch, Snort an Open Source Network Intrusion Detection System. December 2, 2002

- The Honeynet Project, September 2002
http://project.honeynet.org/

- Internet Security System Xforce Website
http://www.iss.net/security_center/advice/Intrusions/2003016/default.htm
http://xforce.iss.net/

- Network Magazine
http://www.networkmagazine.com/article/NMG20020701S0003

- Silicon.com
http://software.silicon.com/malware/0,3800003100,39124939,00.htm

- GAWK
http://gnuwin32.sourceforge.net/packages/gawk.htm

- Gnu.org
http://www.gnu.org/software/gawk/manual/gawk.html

- Kurt Seifried Information Security Ports
http://www.seifried.org/security/ports/

- Andrew B's homepage (Snort Documentation)
http://www.dpo.uab.edu/~andrewb/snort/snortdoc/preplugin.html

- An Analysis on Fragmentation Attacks
http://www.inet-sec.org/docs/DoS/fragma.html

5. Appendices

5.1. Appendix: Server Listing

The following list was taken from the University of Maryland Server Hardware List, located at http://www.gl.umbc.edu/hardware.shtml. The IP addresses were found by resolving the hostnames provided on the list and served to support the work through this document.

Grouping	Hostname	IP	Hardware	OS	Useage	OS Version	Services
Console Service							
	console	MY.NET.6.30	Intel	Linux	Big scary console server		SSH
	console-h1	MY.NET.27.28	PowerPC	Linux-embedded	Little less scary console server		SSH
Directory Service							
	fett.umbc.edu	MY.NET.24.65	Sun E220R, 2Proc	Solaris	Master Directory Server (ds-master.umbc.edu)	1.0	SSH, LDAP
	dengar.umbc.edu	MY.NET.25.34	Sun NetraT1	Solaris	Slave Directory Server	1.0	SSH, LDAP
	ig88.umbc.edu	MY.NET.25.35	Sun NetraT1	Solaris	Slave Directory Server	1.0	SSH, LDAP
Authentication Service							
	kerberos2.umbc.edu	MY.NET.24.57	SGI Indy	IRIX	Secondary KDC		
	kerberos.umbc.edu	MY.NET.24.59	SGI Indy	IRIX	Primary KDC		
Mail Delivery							
	mx1del.umbc.edu	IP Not Resolved by NSLookup	Sun Netra t1	Solaris	Mail Delivery/Lookup	1.0	SSH, SMTP
	mx2del.umbc.edu	IP Not Resolved by NSLookup	Sun Ultra5	Solaris	Mail Delivery/Lookup	1.0	SSH, SMTP
	mx3del.umbc.edu	IP Not Resolved by NSLookup	Sun Ultra5	Solaris	Mail Delivery/Lookup	1.0	SSH, SMTP

mx4del.umbc.edu	IP Not Resolved by NSLookup	Sun Netra t1	Solaris	Mail Delivery/Lookup	1.0	SSH, SMTP
mx1in.umbc.edu	IP Not Resolved by NSLookup	Netra X1	Solaris	Mail Delivery/Lookup	1.0	SSH, SMTP
mx2in.umbc.edu	IP Not Resolved by NSLookup	Netra X1	Solaris	Mail Delivery/Lookup	1.0	SSH, SMTP
mx3in.umbc.edu	IP Not Resolved by NSLookup	Netra X1	Solaris	Mail Delivery/Lookup	1.0	SSH, SMTP
Outgoing Mail Relays						
mx1out.umbc.edu	IP Not Resolved by NSLookup	Netra X1	Solaris	Outgoing Mail Relay	1.0	SSH, SMTP
mx2out.umbc.edu	IP Not Resolved by NSLookup	Netra X1	Solaris	Outgoing Mail Relay	1.0	SSH, SMTP
mx3out.umbc.edu	IP Not Resolved by NSLookup	Netra X1	Solaris	Outgoing Mail Relay	1.0	SSH, SMTP
IMAP/POP Mail Reading						
mr1.umbc.edu	MY.NET.25.18	Sun Enterprise 250	Solaris	UMBC Remote Mail Access	1.0	SSH, IMAP, POP
mr2.umbc.edu	MY.NET.25.19	Sun Enterprise 250	Solaris	UMBC Remote Mail Access	1.0	SSH, IMAP, POP
mr3.umbc.edu	MY.NET.25.20	Sun Enterprise 250	Solaris	UMBC Remote Mail Access	1.0	SSH, IMAP, POP
mr4.umbc.edu	MY.NET.25.21	Sun Enterprise 220R	Solaris	UMBC Remote Mail Access	1.0	SSH, IMAP, POP
Web Services						

auxwww1.umbc.edu	MY.NET.24.29	SGI O2 Server	IRIX	WebCT CourseWare	1.0	SSH, HTTP(80) on webct.umbc.edu
auxwww2.umbc.edu	MY.NET.24.30	SGI Origin 200	IRIX	MyUMBC	1.0	SSH, HTTP(80) on my.umbc.edu, HTTPS on my.umbc.edu
auxwww3.umbc.edu	MY.NET.24.14	SGI Octane (2x R10k)	IRIX	MyUMBC (your.umbc.edu)	1.0	SSH, HTTP(80) on your.umbc.edu, HTTPS on your.umbc.edu
www4.umbc.edu	MY.NET.24.36	Sun NetraT1	Solaris	virthost.umbc.edu	1.0	SSH, HTTP(80) on virthost.umbc.edu
cgi.umbc.edu	MY.NET.6.14	SGI Challenge S	IRIX	cgi.umbc.edu web area		SSH, HTTP(80)
www5.umbc.edu	MY.NET.24.37	Sun Netra T1	Solaris	web development	1.0	SSH
www6.umbc.edu	MY.NET.24.40	Sun Netra T1	Solaris	webauth.umbc.edu	1.0	SSH, HTTP(80) on webauth.umbc.edu, HTTPS on webauth.umbc.edu
www7.umbc.edu	MY.NET.24.41	Sun Netra T1	Solaris	webadmin.umbc.edu	1.0	SSH, HTTP(80) on webadmin.umbc.edu, HTTPS on webadmin.umbc.edu
www8.umbc.edu	MY.NET.24.43	Sun E250	Solaris	www.umbc.edu	1.0	SSH, HTTP(80) on www.umbc.edu
www9.umbc.edu	MY.NET.24.45	Sun E220R, 1proc	Solaris	userpages.umbc.edu	1.0	SSH, HTTP(80) on userpages.umbc.edu
AFS File/Database Servers						
bfs1.afs.umbc.edu	MY.NET.24.112	Sun E250 1proc	Solaris	AFS File Server (data storage)	1.0	SSH
bfs2.afs.umbc.edu	IP Not Resolved by NSLookup	SGI ORIGIN 200	IRIX	AFS File Server (data storage)	1.0	SSH
bfs3.afs.umbc.edu	MY.NET.24.26	SGI ORIGIN 200	IRIX	AFS File Server (data storage)	1.0	SSH

	sauvignon.umbc.edu	130.86.6.38	SGI Challenge S (1x R4400)	IRIX	AFS File Server (admin / sw installs)	1.0	SSH
	smirnoff.umbc.edu	MY.NET.60.6	SGI Origin 200	IRIX	Central and Rem/Ora backups	1.0	SSH
	wedge.umbc.edu	MY.NET.6.45	Sun Netra T1 Ac200	Solaris	AFS File Service (software)	1.0	SSH
	biggs.umbc.edu	MY.NET.60.43	Sun Netra T1 Ac200	Solaris	AFS File Service (software)	1.0	SSH
	hfs1.afs.umbc.edu	IP Not Resolved by NSLookup	Intel P850	Linux	AFS File Server (User Homes)	1.0	SSH
	hfs2.afs.umbc.edu	MY.NET.24.107	Intel P850	Linux	AFS File Server (User Homes)	1.0	SSH
	hfs3.afs.umbc.edu	MY.NET.24.67	Intel P850	Linux	AFS File Server (User Homes)	1.0	SSH
	hfs4.afs.umbc.edu	MY.NET.6.51	Intel P850	Linux	AFS File Server (User Homes)	1.0	SSH
	hfs5.afs.umbc.edu	MY.NET.24.110	Intel P850	Linux	AFS File Server (User Homes)	1.0	SSH
	hfs6.afs.umbc.edu	MY.NET.24.111	Netra T1 AC200	Solaris	AFS File Server (User Homes)	1.0	SSH
	hfs7.afs.umbc.edu	MY.NET.24.114	Netra T1 AC200	Solaris	AFS File Server (User Homes)	1.0	SSH
	db1.afs.umbc.edu	MY.NET.24.101	Sun NetraX1	Solaris	AFS Database Server		SSH
	db2.afs.umbc.edu	MY.NET.24.23	Sun NetraX1	Solaris	AFS Database Server		SSH
	db3.afs.umbc.edu	MY.NET.24.87	Sun NetraX1	Solaris	AFS Database Server		SSH
Other							
	ds2.gl.umbc.edu	MY.NET.60.9	SGI Challenge S (R4400)	IRIX	NFS File Server	1.0	SSH, NFS, HTTP(80)
	news.umbc.edu	MY.NET.24.8	Intel	Linux	Usenet News Service	1.0	SSH, SMTP, NNTP
	listproc.umbc.edu	MY.NET.24.20	Netra T1	Solaris	Mailing Lists	1.0	SSH, SMTP, ILP, HTTP
	ragnarok.umbc.edu	MY.NET.24.27	SGI Challenge S	IRIX	Anon FTP, license service, Proxy Server		SSH, FTP

		(R5000)				
jarjar.umbc.edu	MY.NET.6.15	Sun Enterprise 250	Solaris	Remedy		SSH
threepio.umbc.edu	MY.NET.6.17	Sun Enterprise 250	Solaris	Instructional/Academic Oracle Databases	1.0	SSH
hubris.ucs.umbc.edu	MY.NET.24.62	Intel 2xP2	Linux	Development	1.0	SSH
curly.umbc.edu	MY.NET.1.11	Sun NetraX1	Solaris	System Logging	1.0	SSH
alumni.umbc.edu	MY.NET.60.17	SGI O2	IRIX	Alumni Email Accounts		SSH, TELNET, RLOGIN, SHELL, POP, IMAP, SMTP
umbc7.umbc.edu	MY.NET.6.7	SGI Origin 200	IRIX	Faculty/Staff UNIX Shell Access, Mail Delivery, Web Service, Remote Mail Access	1.0	SSH, SMTP, TELNET, RLOGIN, SHELL, HTTP(80)
irix2.gl.umbc.edu	MY.NET.60.11	SGI Origin 200 (2x R10000)	IRIX	Unrestricted UNIX Shell Access	1.0	SSH, TELNET, RLOGIN, SHELL
linux1.gl.umbc.edu	MY.NET.24.91	2x P850	Linux	Unrestricted UNIX Shell Access	1.0	SSH, TELNET, RLOGIN, SHELL
linux2.gl.umbc.edu	MY.NET.24.89	2x P850	Linux	Unrestricted UNIX Shell Access	1.0	SSH, TELNET, RLOGIN, SHELL
linux3.gl.umbc.edu	MY.NET.24.92	2x P850	Linux	Unrestricted UNIX Shell Access	1.0	SSH, TELNET, RLOGIN, SHELL
titan.umbc.edu	MY.NET.6.20	SGI Challenge XL (20x R10000)	IRIX	Research Computing	1.0	SSH, TELNET, RLOGIN, SHELL
watto.gl.umbc.edu	MY.NET.6.46	Sun Ultra5	Solaris	GL ftp server, AFS/NFS translator	1.0	SSH, FTP, NFS
printhost.umbc.edu	MY.NET.24.15	Sun Ultra5	Solaris	LPRNG Printing Svc.	1.0	SSH, lpd
cal1.umbc.edu	MY.NET.24.3	Sun NetraX1	Solaris	CorporateTime Calendar	1.0	SSH
cal2.umbc.edu	MY.NET.24.4	Sun NetraX1	Solaris	CorporateTime Calendar	1.0	SSH
milter1.umbc.edu	MY.NET.25.2	SunFire 280R	Solaris	Milters: Spam & AntiVirus	1.0	SSH

milter2.umbc.edu	MY.NET.25.3	SunFire 280R	Solaris	Milters: Spam & AntiVirus	1.0