Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 4.1

Submission date:
October 6, 2004

Wouter Clarie
SANS Germany 2004
Delegate Conference
Munich – April 2004

Table of Contents

List of Figures

Part I – Executive Summary

This is an analysis of the network traffic of the University of Maryland, Baltimore Country during three days (April 8 - 10, 2004), based on data provided by the university's intrusion detection system, which tries to detect malicious or suspicious activity. During these three days, a total of 59,972 alerts were generated, or nearly 20,000 alerts per day. It is clear that not enough staff is available to actually monitor 20,000 events per day. An abundance of alerts makes the analysts indifferent, which has adverse effects on security.

Traffic evaluated by an insufficiently finetuned rule set results in many irrelevant alerts Additionally, because the rule set is not really up to date, many attacks or suspicious events might go unnoticed. This gives a false sense of security. If the amount of alerts cannot be drastically reduced by finetuning and updating the ruleset, more resources might be needed to keep the intrusion detection system operational. Possible solutions include: hire more people to monitor the alerts, or make use of automatic correlation infrastructures. These options should be thoroughly researched.

It is important to update not only the intrusion detection system itself, but also the hosts it monitors. Some of the software on the internal network (including the network that provides public services) has been found to be outdated. This often makes these important assets vulnerable to attack. Care must be taken in maintaining a database of software versions to be able to cross-check vulnerability information efficiently.

A number of hosts on the network have been compromised by worm or virus infections. The consequences of this fact are not only unpleasant or risky for the rest of the network, there are also legal issues. Hosts that have been compromised are often used as a stepping stone for new attacks to either internal or external systems. If a host in the internal network actively participates in e.g. a distributed denial of service attack on external targets, the University of Maryland, Baltimore County might be held liable. Measures such as a revised firewalling policy should be taken to prevent these scenarios from occurring. With universities often being hotbeds of worm activity, relations with other universities in the region should be strengthened, to be able to exchange information on worm activity. We need to become more proactive.

Several of the events that were recorded by the intrusion detection system are difficult to research in depth, because of insufficient logging. Storage capacity that can be saved by finetuning the rule sets should be used for more extensive logging. This would make correlation with other devices and infrastructures easier and could result in a faster incident response process.

In the next sections, a more in-depth analysis of the network events will be given.

Part II – Detailed Analysis

1 Files Analyzed

The following table contains the files (from http://isc.sans.org/logs/) that were analyzed.

Analyzed Data		
Alerts	**Scans**	**Out of Specification**
alert.040408	scans.040408	oos_report_040404
alert.040409	scans.040409	oos_report_040405
alert.040410	scans.040410	oos_report_040406

As you may have noticed, the OOS files have different timestamps. The data inside, however, is from the correct dates. For example, oos_report_040404 contains the OOS data from April 8, etc. Some of the scans and alert files were slightly damaged. They have been extracted as much as possible with the normal Unix tools.

2 Relationship Analysis

2.1 Network Topology

First of all, as part of the logs has been anonymized and some have not, I will just use the real addresses of the hosts in this report. This makes life a lot easier, since it allows us to do reverse DNS lookups. It is not news that we are talking about the umbc.edu domain here. After some time trying to reconstruct the network environment, I found out that UMBC has web pages describing their infrastructure. I had already found out quite a lot of information myself, but being able to cross-check it was convenient. Here are the resources I used:

- Intermapper System at http://noc2.noc.umbc.edu/~admin/map_screen.html (advertised on the public website)
- System Hardware List at http://www.gl.umbc.edu/hardware.shtml
- Loic Juillard's GCIA Practical Assignment

From the scan and alerts files, certain things were pretty obvious. There are a few subnets that are used to host the big services. The 130.85.1/24 subnet hosts DNS, the 130.85.12/24 one hosts some mail services, 130.85.24/24 has the web services, the directory server, FTP server, the news server etc, 130.85.25/24 has the big mail installations, with the milters, the outgoing mail exchangers, and the IMAP/POP servers. And then, the 'mysterious' subnet 130.85.30/24, which is hosting Novell Netware servers, probably with storage facilities, confirmed by Tim Kroeger's analysis and a practical by Andrew J. Wagoner.

There is one big connection point in this network, called ernie.umbc.edu. Ernie has at least 120 IP addresses on the network. The following figure is possible setup of the network, based on a diagram by Loic Julliard.

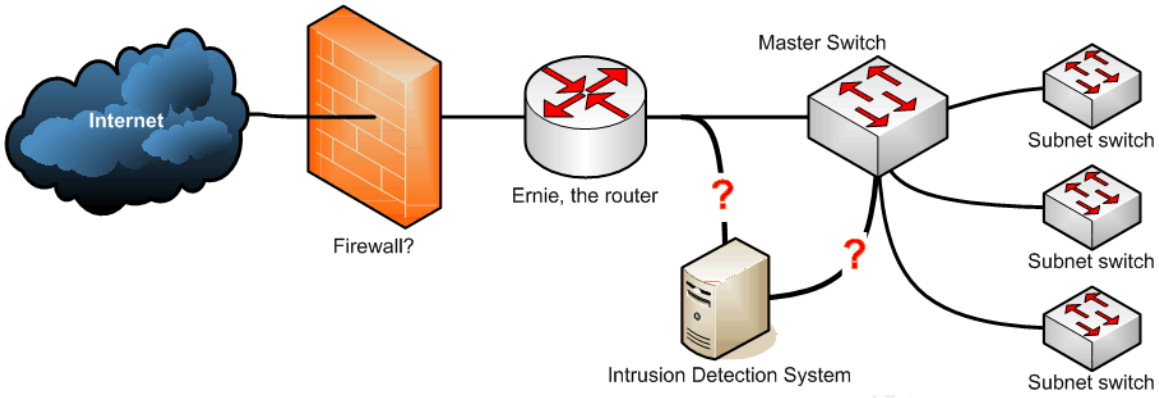

Figure 1: University Network Diagram

2.2 Link Graph

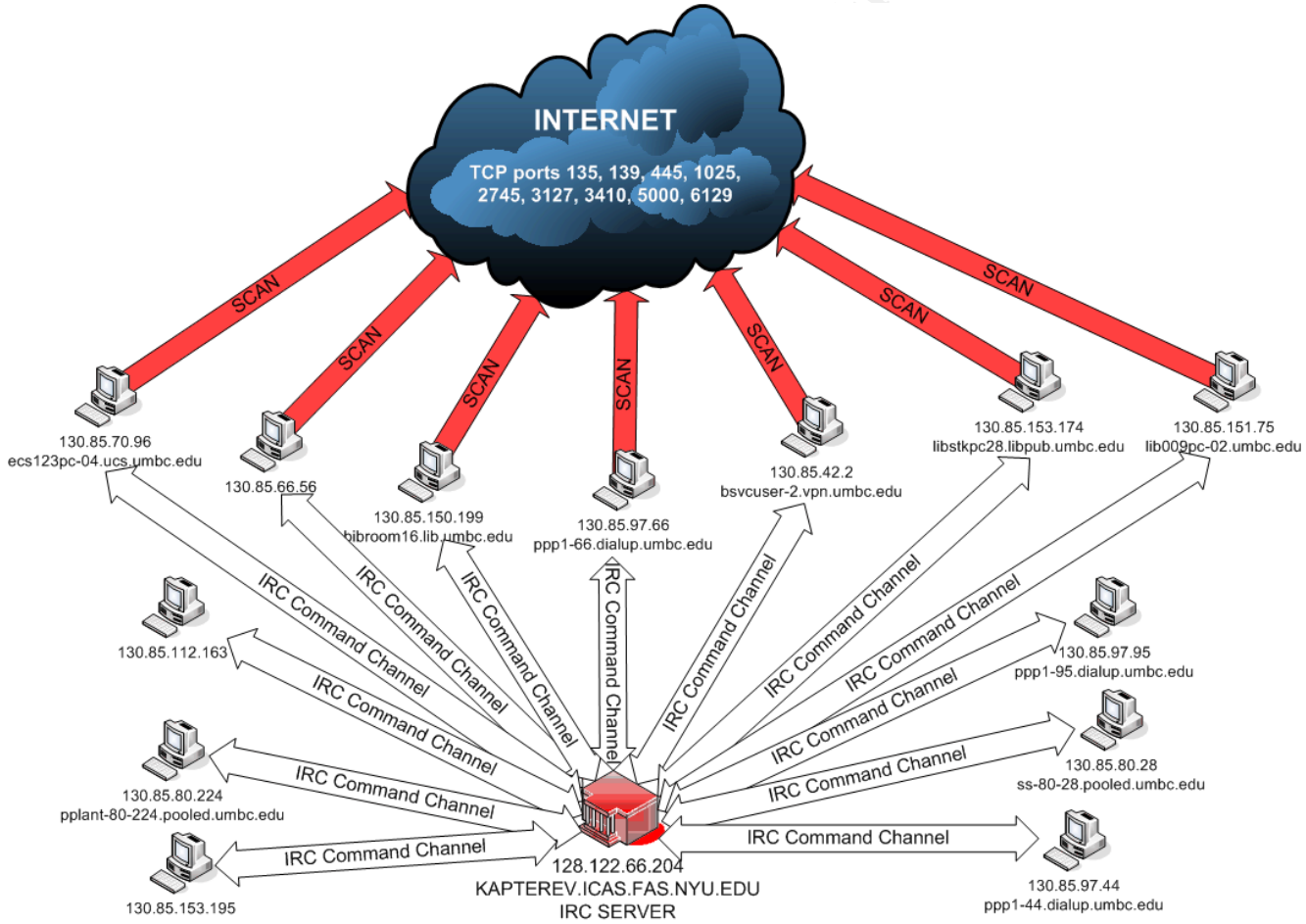

Figure 2: Link Graph of Bot Network

This link graph illustrates the bot network at the university: the compromised hosts, the IRC server at another university, and the activity of the hosts. Refer to section 3.2 for details.

3 Detects

3.1 List of Detects

Alert	Total
EXPLOIT x86 NOOP	17730
MY.NET.30.3 activity	9420
High port 65535 tcp - possible Red Worm - traffic	8962
SMB Name Wildcard	7920
MY.NET.30.4 activity	7248
DDOS mstream handler to client	3265
Possible trojan server activity	942
External RPC call	930
Null scan!	851
NMAP TCP ping!	805
SUNRPC highport access!	582
TCP SRC and DST outside network	253
Incomplete Packet Fragments Discarded	177
UMBC NIDS Internal MiMail alert	129
High port 65535 udp - possible Red Worm - traffic	114
DDOS shaft client to handler	84
UMBC NIDS IRC Alert IRC user /kill detected, possible trojan	73
FTP passwd attempt	71
IRC evil - running XDCC	66
UMBC NIDS IRC Alert Possible sdbot floodnet detected attempting to IRC	42
SMB C access	38
EXPLOIT x86 setuid 0	38
UMBC NIDS External MiMail alert	36
EXPLOIT x86 stealth noop	26
EXPLOIT x86 setgid 0	25
TCP SMTP Source Port traffic	21
UMBC NIDS IRC Alert Possible Incoming XDCC Send Request Detected	14
SYN-FIN scan!	13
NIMDA - Attempt to execute cmd from campus host	13
UMBC NIDS IRC Alert Possible drone command detected.	12
FTP DoS ftpd globbing	11
EXPLOIT NTPDX buffer overflow	10
TFTP - Internal UDP connection to external tftp server	9
RFB - Possible WinVNC - 010708-1	8
EXPLOIT x86 NOPS	8
Attempted Sun RPC high port access	6
Probable NMAP fingerprint attempt	5
DDOS mstream client to handler	5
NETBIOS NT NULL session	3
IRC Alert User joining XDCC channel detected. Possible XDCC bot	2
TFTP - External TCP connection to internal tftp server	2
External FTP to HelpDesk MY.NET.70.50	1
External FTP to HelpDesk MY.NET.53.29	1
External FTP to HelpDesk MY.NET.70.49	1

3.2 Detect 1: IRC Related Activity – Possible botnets

Alert	Number
UMBC NIDS IRC Alert IRC user /kill detected, possible trojan	73
UMBC NIDS IRC Alert Possible sdbot floodnet detected attempting to IRC	42
UMBC NIDS IRC Alert Possible drone command detected.	12
Total	**127**

3.2.1 Description of Detect

This detect is actually a group of detects, all related to certain IRC (Internet Relay Chat) activity on the network. They are caused by three different Snort rules. A total of 127 alerts were generated. There are 49 internal and 33 external hosts involved. The top ten internal and external hosts can be found below, along with the number of alerts they generated.

Internal Hosts	
Host	**Alerts**
130.85.5.44	17
130.85.151.75	10
130.85.153.174	8
130.85.112.163	7
130.85.80.224	7
130.85.70.96	7
130.85.153.195	6
130.85.60.40	6
130.85.150.199	5
130.85.80.28	4

External Hosts	
Host	**Alerts**
128.122.66.204	69
66.40.25.214	9
206.252.192.194	5
24.72.40.108	4
202.91.34.9	4
216.201.150.42	3
216.109.195.222	3
195.169.138.124	2
69.50.174.218	2
64.62.196.26	2

These alerts are usually generated by traffic from computers that have been compromised by a worm. Most worms try to take over a machine using one or more exploits, and continue by hiding themselves and connecting to an IRC server. The attacker then connects to the IRC server as well, joins a channel, and by way of entering commands in that channel, gives specific orders to the compromised hosts. Some of the worms have limited functionality, but others can exploit multiple vulnerabilities and have a whole set of functionalities, such as scanning whole IP ranges for specific ports in order to exploit even more machines, open an unrestricted shell on a specific port, perform a Denial of Service attack, send large amounts of SPAM, or even patch the system and remove itself completely. They can also be used to scan the internal networks from an internal host - conveniently bypassing the border firewall and/or one or more of the perimeter devices - and return the results to the attacker.

It is important to realize that seeing this kind of traffic usually means the hosts have already been compromised and are now 'obeying' the commands of the attacker(s).

3.2.2 Reason this Detect Was Selected

As network resources such as bandwidth are a big cost for most educational institutions such as universities, (Distributed) Denial of Service attacks can be very expensive if they are not prevented from leaving the internal network.

Additionally, as the University of Maryland, Baltimore County actually created custom rules to detect this kind of activity (the names start with 'UMBC NIDS') , it must be that they find it very important to track this activity.

And finally, these worms also perform many activities that are usually illegal (such as port scanning, sending SPAM, etc.), so there is a liability problem too, as noted before by various people.

3.2.3 Detect was Generated by

Snort Intrusion Detection System using custom rules. We will try to reconstruct the rules.

UMBC NIDS IRC Alert IRC user /kill detected, possible trojan.

```
alert tcp $EXTERNAL_NET 6666:7000 -> $HOME_NET any (msg:"UMBC NIDS IRC
Alert IRC user /kill detected, possible trojan."; content:"KILL";
flow:to_client,established;)
```

This is the 'reaction' of the IRC server to the client. If an operator or server uses the /kill command, the server sends a message to the client and then drops the connection by sending a FIN. In 'normal' IRC traffic, the client usually disconnects from the server itself.

UMBC NIDS IRC Alert Possible sdbot floodnet detected attempting to IRC.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 6666:7000 (msg:" UMBC NIDS IRC
Alert Possible sdbot floodnet detected attempting to IRC."; flags:S;)
```

This alert is triggered for outgoing connection setup to IRC servers, but in the logs, there is not much more information than that. If the IRC server is running on a TCP port outside the 'normal' 6666-7000 range, this rule is obviously not sufficient. Additionally, it also fires on 'normal' IRC traffic. This is reflected in the alert message by the word 'Possible'.

UMBC NIDS IRC Alert Possible drone command detected.

It is very difficult to reconstruct this particular rule. The 'drone commands' this rule refers to, can be just about anything. A 'drone command' is basically a command sent to the IRC client (the 'drone'; in this case: the trojan on the infected machine) with instructions. This could be: reboot, hide, but also: portscan, infect other hosts, etc. It should be noted, however, that these commands are usually very dangerous. As far as I know, they are hardly ever false positives.

3.2.4 Probability the Source Address was Spoofed

The IRC protocol is based on TCP connections, which require a three-way handshake. This handshake is virtually impossible to complete with spoofed source addresses. Therefore, the probability that the source address was spoofed, is close to zero.

Spoofing IP addresses in this case is also pointless. We need two-way communications with the hosts, to get results from scans etc. Also note that the attacker's real IP address is not known: only the IP address of the host that was compromised and is running as an IRC server, is known to the particular bot.

3.2.5 Attack Mechanism

First, the host is compromised using one of the unpatched holes in the software, or using a backdoor left on the computer by another worm or attack. You can find resources on the vulnerabilities that are 'usually' exploited by these worms in the References section of this document. Then a trojan is installed, which immediately connects to an IRC server configured in the trojan program. When that has happened, the attacker has complete control over the compromised host.

Often, the attacker then commands the host to start scanning for the ports that are associated with the previously mentioned vulnerabilities. In this case, the ports are 135, 139, 445, 1025, 2745, 3127, 3410, 5000, and 6129. On this IDS, this generated a lot of log files. Some hosts managed to scan more than 400,000 ports in three days. Other hosts remained quiet.

If we correlate the alert data (which contains the alerts for IRC connections being set up and killed) with the data about port scanning, we see some interesting patterns. Have a look at this example:

```
04/09-08:36:07.472302  [**] UMBC NIDS IRC Alert Possible sdbot floodnet
detected attempting to IRC [**] 203.85.151.75:1237 -> 128.122.66.204:7000
```

Immediately followed by:

```
Apr  9 08:36:08 130.85.151.75:1240 -> 130.8.128.221:2745 SYN ******S*
Apr  9 08:36:08 130.85.151.75:1242 -> 130.8.128.221:1025 SYN ******S*
Apr  9 08:36:08 130.85.151.75:1243 -> 130.8.128.221:445 SYN ******S*
Apr  9 08:36:08 130.85.151.75:1244 -> 130.8.128.221:3127 SYN ******S*
Apr  9 08:36:08 130.85.151.75:1245 -> 130.8.128.221:6129 SYN ******S*
Apr  9 08:36:08 130.85.151.75:1246 -> 130.8.128.221:139 SYN ******S*
Apr  9 08:36:08 130.85.151.75:1247 -> 130.8.128.221:3410 SYN ******S*
Apr  9 08:36:08 130.85.151.75:1248 -> 130.8.128.221:5000 SYN ******S*
Apr  9 08:36:08 130.85.151.75:1249 -> 130.200.109.15:2745 SYN ******S*
Apr  9 08:36:08 130.85.151.75:1251 -> 130.200.109.15:1025 SYN ******S*
Apr  9 08:36:08 130.85.151.75:1252 -> 130.200.109.15:445 SYN ******S*
...
Apr  9 09:52:08 130.85.151.75:1656 -> 130.236.237.49:445 SYN ******S*
Apr  9 09:52:08 130.85.151.75:1662 -> 130.236.237.49:3127 SYN ******S*
Apr  9 09:52:08 130.85.151.75:1669 -> 130.236.237.49:6129 SYN ******S*
```

```
04/09-10:18:36.578620  [**] UMBC NIDS IRC Alert IRC user /kill detected,
possible trojan. [**] 128.122.66.204:7000 -> 203.85.151.75:1237
```

What happens is that a trojan installed on 130.85.151.75 (one of the compromised hosts) connects to an IRC server on 128.122.66.204 on port 7000. One second later, the host is already scanning the network for the ports mentioned above.

In other words, the prime goal of compromising these hosts seems to be to compromise even more hosts. Note also that the IRC servers that these trojans connect to, usually reside on compromised hosts as well. In the example, the host with the trojan, connected to 128.122.66.204, which resolves to KAPTEREV.ICAS.FAS.NYU.EDU. I do not think New York University is hosting an IRC service on that kind of address. Patrik Sternudd agrees. This host is also discussed in section 4.3 as one of the most suspicious external hosts.

The net result of this analysis is that at least 13 hosts on the network have been found compromised. The write-up of the full analysis would take too long, but at least the following hosts were compromised by the same family of worms. The first seven of them were actively scanning the network afterwards. The last six were still quiet, or had no other alerts associated to them, apart from the IRC communications with the same server. But since there is no benign reason to communicate with an IRC server on a compromised host that is usually not providing IRC services, it is quite certain that in fact they have been compromised.

Compromised Hosts	
IP address	**FQDN**
130.85.70.96	ecs123pc-04.ucs.umbc.edu
130.85.66.56	(does not resolve)
130.85.42.2	bsvcuser-2.vpn.umbc.edu
130.85.151.75	lib009pc-02.umbc.edu
130.85.153.174	libstkpc28.libpub.umbc.edu
130.85.97.66	ppp1-66.dialup.umbc.edu
130.85.150.199	bibroom16.lib.umbc.edu
130.85.112.163	(does not resolve)
130.85.80.224	pplant-80-224.pooled.umbc.edu
130.85.153.195	(does not resolve)
130.85.97.44	ppp1-44.dialup.umbc.edu
130.85.80.28	ss-80-28.pooled.umbc.edu
130.85.97.95	ppp1-95.dialup.umbc.edu

At least 8 of the hosts in the original top 10 internal hosts list in 3.2.1 were compromised. This has been confirmed in a previous GCIA Practical by Patrik Sternudd. Refer to the link graph in section 2.2 for a graphical view of all compromised hosts that were controlled by 128.122.66.204.

Please note that some of these systems appear to be university-owned computers (such as library computers) and that one of them is using a VPN connection. They are not only students' systems. In other words: not only should corrective action be taken to prevent further spreading of the worm itself (by appropriate packet filtering or firewalling and security policy changes), a number of hosts also need to be taken off the network, analyzed and rebuilt.

3.2.6 Correlations

These scans were first announced in the Handler's Diary at the Internet Storm Center on April 1, 2004. They had not been identified at that time. Continued scanning of the same ports was discussed 5 days later in the Handler's Diary again.

On March 18, there had already been an interesting write-up of the Agobot/Polybot family of worms and their relationship. This document also refers the reader to an excellent analysis of the Phatbot worm by LURHQ. A few other worms, such as 'W32.HLLW.Polybot' and 'W32.HLLW.Gaobot.gen' have been described in detail by Symantec and other vendors.

These bots or worms all exploit one or more of the vulnerabilities described in these security bulletins from Microsoft: MS01-059, MS02-061, MS03-001, MS03-007, MS03-026, MS03-043, MS03-049, and MS-04-011.

The vulnerabilities have been assigned CAN entries by the CVE: CAN-2001-0876, CAN-2001-0877, CAN-2002-1145, CAN-2003-0109, CAN-2003-0352, CAN-2003-0533, CAN-2003-0717, CAN-2003-0812 and others.

These issues have been and are actively discussed on the UNISOG University Security Operations Group) mailing list hosted at DShield / SANS Institute.

Links to all cited resources can be found in the References section at the end of this document.

3.2.7 Evidence of Active Targeting

The IRC traffic itself, and especially the commands being issued that cause the compromised hosts to start scanning networks, are certainly active targeting. Unfortunately, because we do not have the full packet logs, there is no way to be absolutely sure of the commands issued to each host. However, the hosts have most likely been compromised as part of a wide-scale scanning effort. Material that strongly suggests this, can be found at the end of part 3.2.5.

3.2.8 Severity

Criticality. As the whole network is possibly targeted, the criticality of the target is very high. The information about this network seems to indicate that all the important assets are located on UNIX servers, which are not affected by this list

of vulnerabilities. There are also Windows servers on the network, but there is no information on the importance of these systems. Value: **3**.

Lethality. Infection with one of these trojans means total compromise of the system, so the lethality of this is very high: **5**.

System countermeasures. Students' computers are usually not or not very well protected by a host-based firewall and are often not patched properly. Because the number of possible targets is very high, it is difficult to say if all of them are well-protected. Value: **3**.

Network countermeasures. There is no evidence of any effective network countermeasures in this scenario. Machines from the outside can easily connect to inside hosts without apparent restrictions, and vice versa. Note that it is very difficult to actually implement network countermeasures for hosts that have been compromised on another network. As the connection to the botnet is going to ports between 6666 and 7000, which are the 'normal' ports for IRC traffic, it is not really possible to block that outgoing traffic, unless IRC is banned on the whole network as part of the university security policy. Value: **1**.

Severity = (criticality + lethality) – (system + network countermeasures)

Severity = (3 + 5) – (2 + 1) = 5

3.3 Detect 2: FTP DoS ftpd globbing

3.3.1 Description of Detect

```
04/09-09:51:55.991064  [**] FTP DoS ftpd globbing [**]
63.196.157.142:40424 -> MY.NET.24.27:21

04/10-10:44:29.313011  [**] FTP DoS ftpd globbing [**]
65.243.215.17:1913 -> MY.NET.24.27:21

04/10-11:48:19.564709  [**] FTP DoS ftpd globbing [**]
24.106.112.246:4109 -> MY.NET.24.27:21

04/10-11:53:58.127486  [**] FTP DoS ftpd globbing [**]
140.239.150.248:3387 -> MY.NET.24.27:21
04/10-11:53:59.239259  [**] FTP DoS ftpd globbing [**]
140.239.150.248:3387 -> MY.NET.24.27:21
04/10-11:54:00.367257  [**] FTP DoS ftpd globbing [**]
140.239.150.248:3387 -> MY.NET.24.27:21
04/10-11:54:01.611847  [**] FTP DoS ftpd globbing [**]
140.239.150.248:3387 -> MY.NET.24.27:21
04/10-12:01:51.020061  [**] FTP DoS ftpd globbing [**]
140.239.150.248:3387 -> MY.NET.24.27:21
04/10-12:01:56.192552  [**] FTP DoS ftpd globbing [**]
140.239.150.248:3387 -> MY.NET.24.27:21
04/10-12:01:59.997002  [**] FTP DoS ftpd globbing [**]
140.239.150.248:3387 -> MY.NET.24.27:21
04/10-12:02:01.330520  [**] FTP DoS ftpd globbing [**]
140.239.150.248:3387 -> MY.NET.24.27:21
```

This attack targets the University of Washington FTP daemon (wu-ftpd) versions 2.5.0, 2.6.0 and 2.6.1. These versions have a bug in the way they handle filename globbing, causing a possibility for remote root compromise. The attacker needs a valid login to the service. This can be a normal user account or an anonymous login.

I highly recommend to take this server off the network immediately, and perform a forensic analysis on it. It is not clear at this moment if the server has actually been compromised. The repeated attempts by the fourth attacker, who apparently managed to fire 8 alerts on the IDS (in a single session, based on the source port number), may indicate that the exploit did not work against it. Furthermore, after the attacks, no alerts were generated by the FTP server itself. This may be because the attackers are only using it for activity for which there are not rules in the Snort system. But the most important remark here is: none of this can be confirmed without analysis on the machine itself.

Because of the very bad track record of the wu-ftpd and the fact that it appears not to be maintained anymore, I would recommend switching to another FTP server package, such as ProFTPD or vsftpd.

3.3.2 Reason this Detect Was Selected

Years ago this exploit was very 'popular', because many sites were running the University of Washington FTP daemon. It basically meant instant root access back then. Currently, many other FTP daemons are in use, because of wu-ftpd's bad reputation with security bugs. However, universities are a classic target of hackers, because most of them still have a lot of old and vulnerable software running. It is not certain that this targeted FTP server is vulnerable, but it is very likely. Let us start the analysis. This is the banner, grabbed from the particular server:

```
$ ftp ftp.umbc.edu
Connected to 130.85.24.27.
220 ragnarok.umbc.edu FTP server (Version wu-2.6.1(3) Thu Jun 28 19:17:44 EDT
2001) ready.
```

It is running version 2.6.1, which is known to be vulnerable. We are not completely sure yet, because a patch has been issued for this problem. The patch does not change the version number, however. Additionally, the patch was only released on November 29, 2001, and the build number on this version says June 28, 2001, more than five months before the patch was released! It is now quite certain that this particular server is vulnerable.

Note: The other FTP server on the network, ftp1.umbc.edu, is running ProFTPD 1.2.9, which is a completely different product, and is not vulnerable to this kind of attack..

3.3.3 Detect was Generated by

Snort intrusion detection system with an outdated rule set. There used to be a signature called "FTP DoS ftpd globbing", but it is not there anymore. It has now been replaced by two signatures, Snort ID's 1377 and 1378 and has been renamed to "FTP wu-ftp bad file completion attempt X", with X being either '~[' or '~{'.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP wu-ftp bad file
completion attempt ["; flow:to_server,established; content:"~"; content:"[";
distance:1; reference:bugtraq,3581; reference:bugtraq,3707; reference:cve,2001-
0550; reference:cve,2001-0886; classtype:misc-attack; sid:1377; rev:14;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP wu-ftp bad file
completion attempt {"; flow:to_server,established; content:"~"; content:"{";
distance:1; reference:bugtraq,3581; reference:bugtraq,3707; reference:cve,2001-
0550; reference:cve,2001-0886; classtype:misc-attack; sid:1378; rev:14;)
```

3.3.4 Probability the Source Address was Spoofed

As this attack requires an interactive TCP session with two-way communications, it would be very hard to spoof the source address of this attack. Spoofing in this scenario is highly unlikely.

3.3.5 Attack Mechanism

There is a bug in the way wu-ftpd handles filename 'globbing'. From the Security Advisory 'Globbing Vulnerabilities in Multiple FTP Daemons' by COVERT Labs:

```
[...] when an FTP daemon receives a request involving a file that has a tilde
as its first character, it typically runs the entire filename string through
globbing code in order to resolve the specified home directory into a full
path. This has the side effect of expanding other metacharacters in the
pathname string, which can lead to very large input strings being passed into
the main command processing routines. This can lead to exploitable buffer
overflow conditions, depending upon how these routines manipulate their input.
```

In other words: by constructing a special string that includes characters to be expanded by the globbing code (such as the tilde sign '~' being expanded to the home directory) and sending that to the FTP server in a command, we can make the server overwrite a buffer, which later on causes a shell to be executed as the root user, effectively providing a full compromise of the server.

Team Teso released an exploit for this vulnerability, called 7350wurm.c. It is available from Packet Storm. Possibly, this is the exploit code that was used, and which generated the alerts.

3.3.6 Correlations

To SecurityFocus, this vulnerability is known as the "Wu-Ftpd File Globbing Heap Corruption Vulnerability" and has been assigned BugTraq ID 3581. It is also referred to as CVE-2001-0550. CERT has released two advisories on these problems:

- CA-2001-33: Multiple Vulnerabilities in WU-FTPD
- CA-2001-07: File Globbing Vulnerabilities in Various FTP Servers

The vulnerabilty is discussed in detail in an excellent GCIH Practical by Warwick Webb. It was also analyzed as part of a GCIA Practical by Maarten Van Horenbeeck and a GCIH Practical by David McGuine.

The 2002 version of the SANS / FBI Top 20 also refers to wu-ftpd as being insecure.

3.3.7 Evidence of Active Targeting

Based on the available information, there is no indication of the FTP server being attacked in any other way than by the wu-ftpd exploit. There was scanning activity to it from several hosts (a total of 24 packets), but none of these packets were destined for port 21.

None of the four hosts involved in these attacks generated any other logging on the intrusion detection system during the observed time frame: no scanning activity, no alerts.

This may indicate that the attackers had already done some reconnaissance activity before, in a stealthy way, and that they were determined to attack this server in particular.

3.3.8 Severity

Criticality. An FTP server is to be considered critical. It is part of the service network, so it might have trust relationships with other servers in the network. This particular server allows anonymous logins, so the data on the server itself may not be that valuable. However, other data on the machine, which is not accessible from the FTP directories, might be more valuable. Value: **4**.

Lethality. Successful exploitation of this vulnerability usually results in remote root compromise. Value: **5**.

System countermeasures. This host has not been patched for over three years. It is quite safe to assume that the system countermeasures are close to zero. Moreover, it is a public service providing anonymous access. Value: **0**.

Network countermeasures. It does not appear as if the access to this server has been blocked anywhere in the path, as the cited attackers all managed to get an interactive session. It is also very hard to implement network countermeasures for this vulnerability, as it resides in a plaintext service on a freely accessible FTP server, unless access from the outside is not necessary. Value: **1**.

Severity = (criticality + lethality) – (system + network countermeasures)

Severity = (4 + 5) – (0 + 1) = 8

3.4 Detect 3: MY.NET.30.3 and MY.NET.30.4 Activity

Alert	Number
MY.NET.30.3 activity	9420
MY.NET.30.4 activity	7248
Total	**16668**

3.4.1 Description of Detect

The alerts indicate that there was traffic to the hosts 130.85.30.3 and 130.85.30.4 on the internal network. The alerts were generated by traffic to the following TCP ports:

130.85.30.3			
Port	**Alerts**	**Port**	**Alerts**
524	8898	21	3
80	340	8000	3
2745	53	715	3
6129	34	427	3
3019	13	12849	2
4899	12	1433	2
1080	9	446	1
3128	8	55838	1
5000	8		
1025	8		
3410	7		
20168	7		
389	5		

130.85.30.4			
Port	**Alerts**	**Port**	**Alerts**
51443	5267	8000	3
80	1458	715	3
524	344	21	2
2745	54	1433	2
6129	49	446	1
4899	12	57778	1
3128	9	26112	1
1080	7	10080	1
20168	7	20480	1
1025	6	12849	1
3410	6	3862	1
5000	6	55838	1
389	5		

Port 524 is the port used by Netware Core Protocol (NCP). This is where the Novell Directory Services are located. In other words: this is the protocol used to authenticate users and to negiotate authorization. Port 51443 is for the Secure iFolder product of Novell (a.k.a. NetStorage), which provides secure, SSL encrypted access to the storage attached to the server. Port 80 is used on both machines, because it provides a nice login screen for the users. See the next section, 3.4.2, for details.

The other ports are basically the classic ports scanned by numerous hosts. The fact that the ports and the number of alerts are almost the same on both systems clearly indicates that this is caused by a horizontal scan. This is confirmed by a scan log analysis.

Now that we have seen the usage of these servers, let us see who connected to them. The criteria used to build the query were: destination port 80, 524 or 51443, and an exchange of more than 10 packets. The last criterium is introduced because the servers apparently attract many curious users. They just go to the HTTP server on port 80, have a look at the login page, and stop there.

There are quite a few that need more than 10 packets just for that, so those were manually filtered.

The result of that is a list of 27 different hosts. Another 9 were filtered out because they had not sent more than 80 packets and thus were not really a danger for stealing information or brute forcing the authentication system or anything related. These are the 19 hosts that were left:

Host	P80	P524	P51443	FQDN
66.149.110.200	0	380	0	user-119arm8.biz.mindspring.com
151.196.115.104	0	789	0	pool-151-196-115-104.balt.east.verizon.net
128.183.35.77	16	0	363	w223gest.gsfc.nasa.gov
131.92.177.18	0	2169	0	aeclt-cf00a4.apgea.army.mil
134.192.65.152	0	287	0	hshsl152.umaryland.edu
66.151.181.4	242	0	0	default-gw.bos3.fastsearch.net
68.55.113.194	0	0	671	pcp311543pcs.woodln01.md.comcast.net
68.55.116.84	0	146	0	pcp312201pcs.woodln01.md.comcast.net
68.55.129.60	0	0	160	pcp295040pcs.owngsm01.md.comcast.net
68.55.178.168	0	1114	0	pcp233959pcs.elictc01.md.comcast.net
68.55.250.229	0	165	0	pcp261188pcs.howard01.md.comcast.net
68.55.27.157	0	514	0	pcp02560368pcs.owngsm01.md.comcast.net
68.55.62.244	22	0	836	pcp02894056pcs.catonv01.md.comcast.net
68.57.90.146	0	1199	0	pcp912734pcs.brndml01.va.comcast.net
68.81.0.87	18	0	2976	pcp01333933pcs.columb01.pa.comcast.net
69.137.43.10	14	0	429	pcp08648674pcs.towson01.md.comcast.net
69.138.242.40	18	0	212	pcp07724660pcs.nrockv01.md.comcast.net
69.138.77.62	0	1535	0	pcp08479849pcs.desoto01.md.comcast.net
69.3.85.94	23	0	269	h-69-3-85-94.mclnva23.dynamic.covad.net

I tried to locate all of the addresses. At least 15 of the hosts are in the Baltimore area. Two of them are 'around', more precisely in Richmond, Virginia and Arlington, D.C. Then there is a host in Boston (default-gw.bos3.fastsearch.net), which appears to be a gateway or proxy server. And then there is one address (user-119arm8.biz.mindspring.com) that NeoTrace Pro locates in the New York City area, but that does not look right. The trace from my system goes through New York, Philadelphia, Washington DC and back to New York, so probably the last hops just have incorrect location information. I will consider this host as being in the Baltimore area too.

In other words: all of the locations seem reasonably close to the university and could be the residence of university staff, except the address in Boston. However, the only traffic that was sent by the Boston connection went to port 80. This was probably someone looking for NetStorage login pages all over the internet. As there is no port 524 or 51443 traffic from this host, no sensitive information could have been exchanged.

The area information does not completely rule out the possibility of intrusion, though. A smart attacker who managed to get his hands on password information could possibly be located in the area too. On the other hand, downloading content this way would not be very smart at all, since it is all logged by the IDS.

3.4.2 Reason this Detect Was Selected

First of all: the people who constructed this rule set, found this traffic important enough to create custom rules for it. Additionally, many alerts were generated for these rules.

To further understand the importance of this traffic or these hosts, some research is needed. The reverse DNS of these systems shows us that they are called lan1.umbc.edu (130.85.30.3) and lan2.umbc.edu (130.85.30.4). lan2 also has an alias called novell.umbc.edu. As quite a few people have pointed out in the past, these are indeed Novell servers. There is also a 130.85.30.2 (lan3.umbc.edu), but there are no alerts for traffic to that machine anywhere. Maybe it is a backup server.

On the Software page of the Office of Information Technology on the university website, there is a reference to the use of these servers:

```
Novell Netware is used for our campus faculty and staff print and file servers.
```

Ok, so they are file and print services. Very interesting targets for an attacker. In the Winter 2003 OIT newsletter, which is "`provided as a service to UMBC students, faculty and staff`", we can read:

```
Access Your Novell Files Via the Web
If you want to access your personal or department's novell files from home, you
can now do so via the Web through Novell's Web Access feature. Just visit
http://novell.umbc.edu and login with your usual UMBC userid & password. You
can upload, download or delete files, and even modify directories.
```

It is not clear whether this only refers to staff. The document "Novell for Windows NT/2000/XP Installation" seems to indicate it is a staff-only thing:

```
If you are a faculty or staff member and do not yet have a Novell account
please speak to your department head. To use the Novell system you must be on
the UMBC campus and be connected to the Local Area Network.
```

The second sentence in this document is probably outdated now, since it contradicts the previous quote from the newsletter.

In other words: these rules seem to be aimed at monitoring the staff members who access the Novell system from home to use the files that are stored on there. And of course: to monitor people who are trying to access these resources without being authorized to do so. They provide a convenient audit trail, even

though the traffic is only recorded in one direction and there are no packet captures.

3.4.3 Detect was Generated by

Snort Intrusion Detection System with custom rules. Trying to reconstruct the rules:

```
alert ip $EXTERNAL_NET any -> 130.85.30.3 any (msg:"MY.NET.30.3 activity";)
alert ip $EXTERNAL_NET any -> 130.85.30.4 any (msg:"MY.NET.30.4 activity";)
```

These rules log all IP traffic coming from external hosts (outside the 130.85/16 network) and going to 130.85.30.3 or 130.85.30.4. No return traffic is logged.

3.4.4 Probability the Source Address was Spoofed

All of this traffic consists of TCP packets, as part of TCP connections. Due to the nature of the setup of a TCP connection, it is highly unlikely that the source addresses were spoofed. As the possible goal of an attacker in this scenario would be to steal information from the fileserver, spoofing addresses would not be useful.

3.4.5 Attack Mechanism

There is no sign of an attack in these traces, but they looked interesting enough to do a more thorough analysis of them.

Possible ways to attack these services would be to login to the system using credentials that have been compromised by other means. But as noted before, this would leave serious traces.

Another possibility is a brute force attack on credentials. I do not know whether or not this is feasible with Netware products. Nowadays, most authentication systems have failure timers installed: every time you enter a wrong password, you have to wait longer for the system to allow you to try again. Many password systems also lock out the user completely after a number of failed logins.

3.4.6 Correlations

In his GCIA Practical, Erik Montcalm noted that port 524 (Novell Directory Services) should not be available to anyone but the internal users. I do not agree. The data that is transmitted over this channel is encrypted. Moreover, users need valid login credentials to be able to use the service.

The idea to assess the risk of the connecting hosts by trying to pinpoint them on a map, was inspired by Tim Kroeger's analysis.

In the Fall 2004 newsletter of the Office of Information Technology, it was noted that the Netware systems are to be phased out in exchange for a new Active

Directory based Windows domain. Whether or not this is a good decision for security, I do not know.

3.4.7 Evidence of Active Targeting

No evidence of active targeting has been found.

3.4.8 Severity

Criticality. The targets are very critical systems. They provide access to the file systems of university staff members. If these systems are compromised, a huge breach of confidentiality of information can be the result. Value: **5**.

Lethality. As I have no insight into the security architecture of Novell Netware servers and there is no real sign of a specific attack here, I will assign a safe value to the lethality factor. In this case, I think assigning a rather high value is better: better safe than sorry. The fact that this is a modern product produced by a mature vendor is not a good indication of lethality. Novell Netware has had serious vulnerabilities in the past. Value: **3**.

System countermeasures. The Netware servers are running version 6.0. This is not the latest version of the software. There is no information available about the state of patches on these systems, or the level of hardening of the underlying operating systems. There appears to be no restriction of the hosts that can connect to these systems. Again, I am going to assign a safe value, in this case that is a **2**.

Network countermeasures. As this service is announced as being accessible from the internet, I see no real network countermeasures. It is not clear whether or not other traffic to this host is blocked by a perimeter device. This is again a safe value, because of a lack of information. Value: **2**.

Severity = (criticality + lethality) – (system + network countermeasures)

Severity = (5 + 3) – (2 + 2) = 4

4 Network Statistics

4.1 Top Five Talkers

4.1.1 The Alert Logs

Based on the raw alerts data, these are the top five internal and external talkers.

Internal based on Alerts		
IP address	**FQDN**	**Alerts**
130.85.11.7	dc2.ad.UMBC.EDU	4794
130.85.84.235	(does not resolve)	3382
130.85.60.16	linux2.gl.umbc.edu	2169
130.85.97.51	ppp1-51.dialup.umbc.edu	619
130.85.75.13	chpdm.umbc.edu	482

As far as I can tell from the naming of number one, this must be a domain controller in an Active Directory system, so it must be a Windows machine. All alerts that were generated by this machine were SMB Name Wildcards, all sent to the 169.254/16 'autoconfiguration' range. Windows automatically puts you in that range when it cannot find a DHCP server. This looks like a configuration problem. Number two is scanning for eDonkey (port 4662). That traffic is incorrectly regarded as mstream because it originates from port 12754. Also a number of 'Possible trojan server activity' alerts are generated by this host, because it uses local port 27374. This looks suspicious, especially because it also generates 'High port 65535 tcp - possible Red Worm – traffic' alerts caused by traffic to port 65535. Three is one of the Linux shell servers, with an SSH session connected to it from port 65535, which generates 'Red Worm traffic'. As far as I can tell, the traffic looks like normal SSH. From this IP, there is also one UDP packet going to the TFTP (69) port of 128.186.103.201, which resolves to neptune.gsfc.nasa.gov and appears to be close to Baltimore. The fourth one is a dialup connection from the university, generating many TCP SYN packets to port 65535 at the same host (619 packets) and also scanning other hosts on different ports (not in the alert data). Number five is a Windows system used by the Center for Health Program Development and Management. It is generating SMB Name Wildcards (UDP port 137) to all sorts of servers outside the network. These look very much like response packets to probes from scanners.

External based on Alerts		
IP address	**FQDN**	**Alerts**
68.81.0.87	pcp01333933pcs.columb01.pa.comcast.net	2994
141.157.102.155	pool-141-157-102-155.balt.east.verizon.net	2694
131.92.177.18	aeclt-cf00a4.apgea.army.mil	2169
69.138.77.62	pcp08479849pcs.desoto01.md.comcast.net	1535
68.57.90.146	pcp912734pcs.brndml01.va.comcast.net	1199

Number one generated a lot of RPC traffic to the Novell server on 130.85.30.4. The second one is the same case as number three in the first list: the return

packets in the SSH session (also see Appendix). The third entry is a session to the 130.85.30.3 Novell server on port 524, generating the usual 'MY.NET.30.3 traffic' alert for each packet. Number four and five are the same thing. The first four addresses are located in the Baltimore area. The fifth one is in Richmond. The four last entries are most probably caused by 'normal' traffic from students or teachers of the university, combined with a lack of IDS rule set tuning.

4.1.2 The Scans Logs

There are some very active scanners in the network. Here are the tables.

Internal TCP based on Scans			
IP address	**FQDN**	**Scans**	**Activity**
130.85.111.51	trc208pc-02.engr.umbc.edu	1047994	Port 135
130.85.81.39	someone @ umbc.edu	745449	Port 135
130.85.70.96	ecs123pc-04.ucs.umbc.edu	471920	Agobot/Phatbot scanning
130.85.66.56	someone @ umbc.edu	334879	Agobot/Phatbot scanning
130.85.42.2	bsvcuser-2.vpn.umbc.edu	253156	Agobot/Phatbot scanning

First there is one PC in the engineering department scanning for port 135, followed by another 'anonymous' machine inside the university network, also scanning for port 135. The next three are internal systems infected with Phatbot or Agobot, scanning for their specific series of ports: 135, 139, 445, 1025, 2745, 3127, 3410, 5000, and 6129. One thing which is very important: one of the VPN users apparently is infected. As the VPN is a layer of defense, it has already been breached this time.

Internal UDP based on Scans			
IP address	**FQDN**	**Scans**	**Activity**
130.85.153.35	refweb06.libpub.umbc.edu	1079394	Seemingly random ports and hosts
130.85.110.72	eds-lin1.engr.umbc.edu	128279	Seemingly random ports and hosts
130.85.111.34	fsc2.engr.umbc.edu	89137	eMule activity
130.85.53.169	ecs122pc06.ucslab.umbc.edu	72703	Seemingly random ports and hosts
130.85.97.30	ppp1-30.dialup.umbc.edu	70570	Korean File sharing App

Three of these are apparently scanning random hosts and ports. I could not find any pattern whatsoever, and no references describing this were found on the internet. Number three is just using the eMule file sharing application, and number five is using the Korean File sharing application. Credits go to Michael Wisener for finding that out.

External TCP based on Scans			
IP address	**FQDN**	**Scans**	**Activity**
61.146.52.26	someone @ chinanet.cn.net	28219	Port 80
138.100.42.180	(does not resolve)	27798	Port 80
136.142.36.112	(does not resolve)	26338	Port 6129
64.218.200.19	SBC Internet Services	25641	Port 6129
211.239.150.130	someone @ hostway.co.kr	23863	Port 80

Apart from the fact that four of these five addresses belong to the Asia-Pacific region and that they are probably looking for vulnerable HTTP servers and open Dameware Remote remote control software, there is not much interesting going on here.

External UDP based on Scans			
IP address	**FQDN**	**Scans**	**Activity**
210.192.127.28	(does not resolve)	257	Ports 1025-1029
130.49.65.144	dorms-pppoe-1-65-144.pittsburgh.resnet.pitt.edu	241	Port 137
130.228.88.235	130.228.88.235.ip.tele2adsl.dk	185	Port 137
212.144.11.198	dialin-212-144-011-198.arcor-ip.net	159	Port 137
219.133.113.140	someone @ chinanet.cn.net	137	Port 137

First, some checks for Microsoft RPC/LSA ports, which can be exploited easily. Then we have all traffic going to port 137, another Windows port, looking for vulnerable machines.

4.2 Top Five Targeted Services

Setting up lists of top targeted services is not an easy task. Just extracting the information from the alerts database would produce very useless results, because we have already seen that the rules produce vast amounts of false positives. The scans logs are an other option. And then there is the OOS traffic.

4.2.1 The Alert Logs

A 'dumb' analysis on the raw data of the alerts file generates the first list for the services that were targeted inside the network. The fact that port 80, being the most 'popular' port on the internet for 'normal' traffic, is being attacked the most, would not be surprising. But at least 16,000 of the alerts for port 80 were actually 'EXPLOIT x86 NOOP' alerts, which are commonly known as being false positives with servers serving binary data. Also the 'Possible trojan server activity' alerts were filtered out, because they were just normal traffic to the main web server (www.umbc.edu). Additionally, all the alerts for port 22 (SSH) were 'Possible red worm traffic'. These turned out to be all false positives too. All of this port 22 traffic went to 130.85.60.16, also known as linux2.gl.umbc.edu, a shell server for the students. See also the analysis in 4.1. These alerts were filtered. The port 51443 traffic was not filtered, because all the alerts were 'MY.NET.30.4 traffic', which is a rule that probably just records all traffic to that host. The sole fact that all traffic to that host is logged, indicates that it must be providing an important and valuable service. The result of the filtering operation is shown in the second table.

Raw data from database		
Port	**Description**	**Alerts**
80	HTTP	18044
524	Novell	9242
51443	RPC	5267
22	SSH	2694
1025	Microsoft RPC/LSA	936

Corrected data		
Port	**Description**	**Alerts**
524	Novell	9242
51443	RPC	5267
80	HTTP	2028
1025	Microsoft RPC/LSA	936
111	RPC	930

4.2.2 The Scans Logs

So, what do the scans logs say? Outgoing UDP is certainly not interesting. 90% of it are false positives, mainly our own DNS server traffic (2,424,972 packets from 130.85.1.{2,4,5} to other DNS servers), 189,538 packets for the Korean file sharing application mentioned before, another 83,000+ packages for eMule protocols, our own NTP servers synchronizing, some online gaming and Kazaa… Having more than 4 million UDP packets logged without precise filtering apparently is not very useful.

Incoming UDP is somewhat more interesting. The majority of that traffic (1074 packets) goes to port 137, the NetBIOS Name Service, probably looking for open shares on Windows machines. Next is 82 packets to port 53. These probably came in fast, causing them to be alerted as a port scan. The rest is mainly DCOM-related scanning to ports 1025-1029.

If we are talking about incoming TCP, port 135 (Microsoft RPC) is certainly the champion with 1,932,041 packets. This is probably scanning from MSBlast and similar worms trying to exploit vulnerabilities in the RPC implementation (see CAN-2003-0352, CAN-2003-0528, CAN-2003-0533, CAN-2003-0605 and CAN-2003-0715).

4.2.3 Out of Specifications Logs

Out of Specifications		
IP Address	**FQDN**	**Entries**
68.54.84.49	pcp01741335pcs.howard01.md.comcast.net	968
202.144.28.167	kurinji.tenet.res.in	691
141.224.64.4	bessie.augsburg.edu	144
193.170.194.27	(does not resolve)	135
66.225.198.20	unknown.splashhost.net	97

As these are 'Out of Specifications', we can expect some 'weird' trafic here. The first IP is from someone checking his e-mail (POP3, TCP port 110 on 130.85.6.7) every minute. Every packet gets tagged as "SYN 12****S* RESERVEDBITS" by Snort. The same traffic is generated by host number two, but to port 4662 (eDonkey) on host 130.85.70.225 this time. This host is located in India. Number three: again the same tagging by Snort, this time to port 25 (SMTP) on 130.85.12.4, one of the mailservers. The fourth entry is sending these packets to

port 113 (ident) on host 130.85.110.82 and a few to port 25 (SMTP) on 130.85.12.6. Number five is the same story, again to port 25 of 130.85.12.6.

A mailing list post by Victor Barahona seems to indicate that this is caused by Explicit Congestion Notification (ECN) bits in the TCP header being set. ECN itself is described in RFC 3168: "The Addition of Explicit Congestion Notification (ECN) to IP". From the RFC:

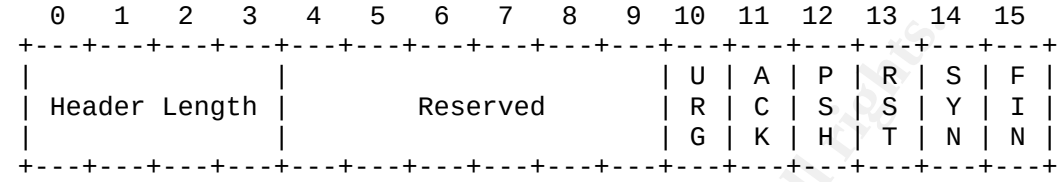

Figure 3: The old definition of bytes 13 and 14 of the TCP header.

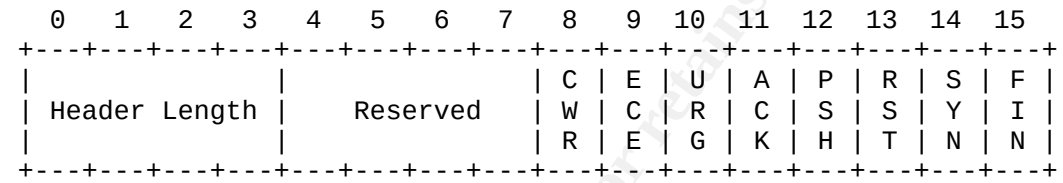

Figure 4: The new definition of bytes 13 and 14 of the TCP Header.

```
We call a SYN packet with the ECE and CWR flags set an "ECN-setup SYN packet"
[...] Before a TCP connection can use ECN, Host A sends an ECN-setup SYN packet
```

I did not find any indication of malicious activity or intent in this traffic.

4.3 Most Suspicious External Source Addresses

4.3.1 KAPTEREV.ICAS.FAS.NYU.EDU

This host, 128.122.66.204 or KAPTEREV.ICAS.FAS.NYU.EDU, is actively involved in controlling worm or trojan compromised hosts. It is running an IRC server on port 7000 for that purpose. In the analyzed timeframe, a total of 13 different internal hosts connected or communicated with this IRC server. The server is most probably also compromised, because a university usually does not provide IRC services, and certainly not on hosts with 'obscure' names such as this one. Refer to the Link Graph in section 2.2 for a graphical overview of what hosts it controlled.

Registration information:

```
OrgName:    New York University
OrgID:      NYU
Address:    Academic Computing Facility
Address:     251 Mercer Street
City:       New York
StateProv:  NY
PostalCode: 10012
Country:    US
```

```
NetRange:   128.122.0.0 - 128.122.255.255
CIDR:       128.122.0.0/16
NetName:    NYU-NET
NetHandle:  NET-128-122-0-0-1
Parent:     NET-128-0-0-0-0
NetType:    Direct Assignment
NameServer: CMCL2.NYU.EDU
NameServer: EGRESS.NYU.EDU
NameServer: NYUNSB.NYU.EDU
Comment:
RegDate:    1986-05-02
Updated:    2001-05-21

TechHandle: ZN68-ARIN
TechName:   New York University
TechPhone:  +1-212-998-3431
TechEmail:  NOC@nyu.edu
```

4.3.2 ip248.netriplex.com

This is the host that triggered 8 FTP globbing alerts on the intrusion detection system. The IP address is 140.239.150.248. The PTR record gives us the FQDN ip248.netriplex.com, but this name does not resolve back. Therefore, along with the netblock registration, I am providing the registration information of the domain name itself too. NeoTrace Pro puts this address in the Boston area.

Netblock:

```
Allegiance Telecom Companies Worldwide ALGX-HVD-BLK1 (NET-140-239-0-0-1)
                                  140.239.0.0 - 140.239.255.255
CONFUSCIUS LTD HTW-04627 (NET-140-239-150-241-1)
                                  140.239.150.241 - 140.239.150.254
```

Domain name:

```
   NETRIPLEX, LLC
   1112 Boylston Street
   Second Floor, PMB17
   Boston, Massachusetts 02215
   United States

   Registered through: GoDaddy.com
   Domain Name: NETRIPLEX.COM
      Created on: 18-Nov-02
      Expires on: 18-Nov-04
      Last Updated on: 21-Jul-04

   Administrative Contact:
      Manager, IT  itmanager@netriplex.com
      NETRIPLEX, LLC
      1112 Boylston Street
      Second Floor, PMB17
      Boston, Massachusetts 02215
      United States
      6172424855      Fax -- 0000000000
   Technical Contact:
      Manager, IT  itmanager@netriplex.com
      NETRIPLEX, LLC
      1112 Boylston Street
```

```
      Second Floor, PMB17
      Boston, Massachusetts 02215
      United States
      6172424855      Fax -- 0000000000

   Domain servers in listed order:
      NS1.NETRIPLEX.COM
      NS4.NETRIPLEX.COM
```

4.3.3 61.146.52.26

This is the top external scanner, which sent us more than 28,000 packets. The IP address does not have a PTR record. This is the registration information:

```
inetnum:      61.140.0.0 - 61.146.255.255
netname:      CHINANET-GD
descr:        CHINANET Guangdong province network
descr:        Data Communication Division
descr:        China Telecom
country:      CN
admin-c:      CH93-AP
tech-c:       IC83-AP
mnt-by:       APNIC-HM
mnt-lower:    MAINT-CHINANET-GD
status:       ALLOCATED PORTABLE
changed:      hm-changed@apnic.net 20040914
source:       APNIC

person:       Chinanet Hostmaster
address:      No.31 ,jingrong street,beijing
address:      100032
country:      CN
phone:        +86-10-66027112
fax-no:       +86-10-58501144
e-mail:       hostmaster@ns.chinanet.cn.net
e-mail:       anti-spam@ns.chinanet.cn.net
nic-hdl:      CH93-AP
mnt-by:       MAINT-CHINANET
changed:      hostmaster@ns.chinanet.cn.net 20021016
remarks:      hostmaster is not for spam complaint,please send spam complaint
to anti-spam@ns.chinanet.cn.net
source:       APNIC

person:       IPMASTER CHINANET-GD
nic-hdl:      IC83-AP
e-mail:       ipadm@gddc.com.cn
address:      NO.1,RO.DONGYUANHENG,YUEXIUNAN,GUANGZHOU
phone:        +86-20-83877223
fax-no:       +86-20-83877223
country:      CN
changed:      ipadm@gddc.com.cn 20040902
mnt-by:       MAINT-CHINANET-GD
remarks:      IPMASTER is not for spam complaint,please send spam complaint to
abuse@gddc.com.cn
source:       APNIC
```

5 Correlations

As these events and this network have been analyzed before, there are many references to other GCIA Practicals. These have been referenced in the

appropriate sections, as well as in the References section. 'New' vulnerability information and background is referenced in the specific sections too. Please refer to the other parts of this document for these correlations.

6 Compromised Internal Hosts

In section 4.1.2, I already mentioned that quite a few hosts had been infected with some form of the Agobot/Phatbot worm. This analysis is based on the scanning pattern showed by these hosts. Worm compromised hosts are listed in section 3.2.6, based on the outgoing IRC connections these hosts made to a 'controlling' IRC server. In addition, ftpd.umbc.edu might have been compromised. Refer to section 3.3 for details on that host.

7 Defensive Recommendations

- Fine tune the IDS rule set for port scanning, because the service networks are in there too. For example, the DNS servers obviously generate a lot of UDP traffic, which is recorded as port scanning.

- Fine tune the rest of the rule set, because it generates so many false positives that analyzing this traffic is nearly impossible if you want to maintain mental sanity of the administrators. Especially the shellcode alerts should be turned off for certain servers or ports, such as the main web server. Updating the rule set would allow to correctly identify the 'new' forms of suspicious traffic instead of alerting on old signatures that are 'too broad'.

- If sufficient hardware is available, log whole sessions for certain alerts. This can be easily accomplished with the normal Snort rules.

- Dump alerts and/or traffic in pcap format, not in ASCII. Maybe this is already the case, and the logs that were provided have been post-processed already. Pcap format allows the analyst to find out a lot more information about what caused the alert, and allows for more in depth analysis of the network topology, in certain scenarios.

- Upgrade the version of Snort. This version does not understand the Explicit Congestion Notification (ECN) bits and marks them as bad traffic. Additionally, Snort has been found vulnerable to a buffer overflow in the stream4 preprocessor, for which exploit code has been posted on the internet. You do not want your Intrusion Detection System to be compromised. Refer to the References section for more information.

- If legally possible, schedule a weekly or daily scan of all network resources, including the service networks and the students' machines, to be able to identify compromised machines fast. Additionally, by using strong authentication mechanisms for the 'untrusted' computers on the network, it is easy to find out who exactly was using a certain IP address at the time of a

compromise. For a nice example of this, refer to OpenBSD's authpf(8) manual page. The idea is simple: all network access is blocked by default, except SSH sessions to a certain host, which is controlling the access. By logging in to this server, the authpf(8) starts, and, based on the configuration, gives the particular host access to certain resources. Authentication can be done with LDAP, Kerberos or any other standard. When the host logs off (closes the SSH session), the access privileges are removed again. This way, not only the access is controlled, but the IP address used is also linked to a user ID.

- If this is not already being done, block all incoming (from the internet) TCP and UDP connections to hosts that are not on public service subnets. It is unclear if this kind of firewalling is set up right now.

Part III – Analysis Process

Hardware and Software

All operations were carried out on OpenBSD 3.5 (i386) in a VMware Workstation on Windows XP SP1 and on the host operating system itself. The machine I used is a 2.6 GHz Pentium IV machine with 1 GB of physical memory and plenty of disk space.

Because of the vast amount of data, I decided to put all records in a MySQL 4.0.x database. Therefore, I wrote several Perl (version 5.8.2) and shell scripts, and used the standard UNIX tools. I based the scripts on work of others. The analysis work was done with SQL statements through a popular web interface to the MySQL database: phpMyAdmin, in this case version 2.6.0 running on an Apache 1.3.29 server with PHP 4.3.

Getting the data into the database was not very hard, but it took a while. For example, the table with UDP scans and TCP scans each contained more than 4 million records. On the cited configuration, importing the data took almost four hours.

I also noticed that generating indexes on the srchost, srcport, dsthost and dstport fields in the tcpscan and udpscan tables speeds things up tremendously, especially when using grouping queries etc. Generating the indexes takes some time, but you only have to do it once, since you are not inserting any data afterwards.

The figures have been produced with Microsoft Office Visio 2003. Tables were managed with Microsoft Excel. To locate certain hosts based on their IP addresses, I used NeoTrace Pro version 3.25.

Handling the Data Files

The data files were provided in a raw ASCII form, which was not ready for analysis. For each of the data files, several operations were needed to be able to handle them with scripts or databases.

Alert Files

Processing the alert files went as follows:

Merge all the files together.

```
$ cat alert.040408 alert.040409 alert.040410 > alert.full
```

Check which lines do not start with a date in the covered timeframe, and dump results, including a line number, to the 'anomalies' file.

```
$ egrep -nv "(^04/08)|(^04/09)|(^04/10)|(^04/11)" alert.full > anomalies
```

Manually fix some of the stuff, using the anomalies file as a guide. It seems these errors are the result of either a concurrency bug in Snort, or a problem with multiple Snort instances running on the same host or logging to the same file. Typical Example:

```
04/08-13:12:37.778948  [**] MY.NET.30.3 activity [**]
131.92.177.18:1033 -> MY.NET.30.3:524
04/08-13:12:37.988681  [**] MY.NET.30.3 activity [**]
131.92.177.1804/08-14:05:40.308048  [**] spp_portscan: portscan status
from MY.NET.80.5: 5 connections across 4 hosts: TCP(5), UDP(0) [**]
:1033 -> MY.NET.30.3:524
```

The third line obviously needs to be added to the first line, and the second line needs a line break after the `131.92.177.18`.

Next, the data from alerts and port scans is fed to the MySQL database using the script outlined in the Appendix.

```
$ ../tools/alert2mysql.pl alert.full
$ ../tools/portscan2mysql.pl alert.full
```

Scans Data

Same thing: concatenate all the files, filter the right dates, and sort the data. The `-H` flag to `sort(1)` is not used on all systems. From the OpenBSD manual page:

```
     -H      Use a merge sort instead of a radix sort.  This option should be
             used for files larger than 60Mb.
```

```
$ cat scans.040408 scans.040409 scans.040410 > scans.full
$ cat scans.full | egrep '(^Apr  8)|(^Apr  9)|(^Apr 10)' | sort -H > scans.tmp
$ mv scans.tmp scans.full
```

Then, import the scans into the database using the Perl scripts.

```
$ ../tools/tcpscans2mysql.pl scans.full
$ ../tools/udpscans2mysql.pl scans.full
```

Out of Specification (OOS) Data

Merge all the OOS files together.

```
$ cat oos_report_040404 oos_report_040405 oos_report_040406 > oos.full
```

Generate a list of the source addresses causing OOS alerts. This is based on a command line by Maarten Van Horenbeeck.

```
$ cat oos.full | egrep '(^04/08)|(^04/09)|(^04/10)' | cut -d " " -f 2 \
  | cut -d ":" -f 1 | sort | uniq -c | sort -r -n > oos.summary
```

References

Relationship Analysis

University of Maryland, Baltimore County (UMBC); Intermapper System;
http://noc2.noc.umbc.edu/~admin/map_screen.html

University of Maryland, Baltimore County (UMBC); System Hardware List;
http://www.gl.umbc.edu/hardware.shtml

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

Tim Kroeger; GCIA Practical Assignment;
http://www.giac.org/practical/GCIA/Tim_Kroeger_GCIA.pdf

Andrew J. Wagoner; GCIA Practical Assignment;
http://www.giac.org/practical/GCIA/Andrew_J_Wagoner_GCIA.pdf

Patrik Sternudd; GCIA Practical Assignment;
http://www.giac.org/practical/GCIA/Patrik_Sternudd_GCIA.pdf

Detect 1: IRC Related Activity – Possible botnets

Michael Meacle; GCIA Practical Assignment;
http://www.giac.org/practical/GCIA/Michael_Meacle_GCIA.pdf

Symantec Security Response; W32.HLLW.Lovgate.J@mm;
http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.lovgate.j@mm.html

Patrik Sternudd; GCIA Practical Assignment;
http://www.giac.org/practical/GCIA/Patrik_Sternudd_GCIA.pdf

SANS Internet Storm Center; Handler's Diary April 1st 2004;
http://isc.sans.org/diary.php?date=2004-04-01

SANS Internet Storm Center; Handler's Diary April 5th 2004;
http://isc.sans.org/diary.php?date=2004-04-05

SANS Internet Storm Center; Handler's Diary March 18th 2004;
http://isc.sans.org/diary.php?date=2004-03-18

LURHQ Threat Intelligence Group; Phatbot Trojan Analysis;
http://www.lurhq.com/phatbot.html

Symantec Security Response; W32.HLLW.Polybot;
http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.polybot.html

Symantec Security Response; W32.HLLW.Gaobot.gen;
http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.gaobot.gen.html

Microsoft Corporation; Security Bulletins;
http://www.microsoft.com/technet/security/bulletin/MS01-059.mspx
http://www.microsoft.com/technet/security/bulletin/MS02-061.mspx
http://www.microsoft.com/technet/security/bulletin/MS03-001.mspx
http://www.microsoft.com/technet/security/bulletin/MS03-007.mspx
http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx
http://www.microsoft.com/technet/security/bulletin/MS03-043.mspx
http://www.microsoft.com/technet/security/bulletin/MS03-049.mspx
http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx

CVE/CAN-entries;
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0876
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0877
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-1145
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0109
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0717
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0812

DShield; UNISOG – University Security Operations Group mailing list;
http://www.dshield.org/mailman/listinfo/unisog

Detect 2: FTP DoS ftpd globbing

CERT Advisory CA-2001-07: File Globbing Vulnerabilities in Various FTP Servers;
http://www.cert.org/advisories/CA-2001-07.html

CERT Advisory CA-2001-33: Multiple Vulnerabilities in WU-FTPD;
http://www.cert.org/advisories/CA-2001-33.html

Network Associates COVERT Labs; Globbing Vulnerabilities in Multiple FTP Daemons;
http://packetstormsecurity.nl/advisories/nai/nai.00-ftp.glob

SecurityFocus; Wu-Ftpd File Globbing Heap Corruption Vulnerability;
http://www.securityfocus.com/bid/3581

http://cve.mitre.org/cgi-bin/cvename.cgi?name=2001-0550

Warwick Webb; GCIH Practical Assignment;
http://www.giac.org/practical/Warwick_Webb_GCIH.doc

David McGuire; GCIH Practical Assignment;
http://www.giac.org/practical/David_McGuire_GCIH.doc

Snort Signatures
http://www.snort.org/snort-db/sid.html?id=1377
http://www.snort.org/snort-db/sid.html?id=1378

Team Teso; 7350wurm.c; http://packetstormsecurity.nl/0205-exploits/7350wurm.c

Patch for wu-ftpd 2.6.1; ftp://ftp.wu-ftpd.org/pub/wu-ftpd/patches/apply_to_2.6.1/ftpglob.patch

The SANS Institute; The Twenty Most Critical Internet Security Vulnerabilities; http://www.sans.org/top20/oct02.php#U5

ProFTPD; http://www.proftpd.org/

vsftpd; http://vsftpd.beasts.org/

Detect 3: MY.NET.30.3 and MY.NET.30.4 Activity

University of Maryland, Baltimore County (UMBC); Novell for Windows NT/2000/XP Installation; http://www.umbc.edu/oit/sans/desktopsupport/installation/novell/windows/2000_xp/

University of Maryland, Baltimore County (UMBC); Software Menu; http://www.umbc.edu/oit/software/

University of Maryland, Baltimore County (UMBC); OIT Newsletter: Winter 2003; http://www.umbc.edu/oit/newsletter/winter2003.html

University of Maryland, Baltimore County (UMBC); OIT Newsletter: Fall 2004; http://www.umbc.edu/oit/newsletter/fall2004.html

Novell; Ask the Experts: BorderManager Cool Solutions Q&A Collection ; Filter exceptions for GroupWise; http://www.novell.com/coolsolutions/bordermag/ask_the_experts.html

Tim Kroeger; GCIA Practical Assignment; http://www.giac.org/practical/GCIA/Tim_Kroeger_GCIA.pdf

Erik Montcalm; GCIA Practical Assignment; http://www.giac.org/practical/GCIA/Erik_Montcalm_GCIA.pdf

Network Statistics

Ryan Troll; Automatically Choosing an IP Address in an Ad-Hoc IPv4 Network; IETF Draft; http://www.ietf.org/proceedings/99mar/I-D/draft-ietf-dhc-ipv4-autoconfig-03.txt

Michael Wisener; GCIA Practical Assignment; http://www.giac.org/practical/GCIA/Michael_Wisener_GCIA.pdf

CVE/CAN-entries;
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0528
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0605
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0715

American Registry for Internet Numbers (ARIN); WHOIS Database Search; http://www.arin.net/whois/

Victor Barahona; Snort-users mailing list; January 12, 2001; http://archives.neohapsis.com/archives/snort/2001-01/0183.html

K. Ramakrishnan; The Addition of Explicit Congestion Notification (ECN) to IP; Request for Comments number 3168; http://www.rfc-editor.org/rfc/rfc3168.txt

Defensive Recommendations

Snort TCP Packet Reassembly Integer Overflow Vulnerability
BugTraq ID: http://www.securityfocus.com/bid/7178
Exploit code: http://www.securityfocus.com/data/vulnerabilities/exploits/p7snort191.sh
CVE/CAN entry: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0209

OpenBSD authpf(8) manual page; http://www.openbsd.org/cgi-bin/man.cgi?query=authpf&sektion=8&manpath=OpenBSD+3.5

OpenBSD; PF: Authpf: User Shell for Authenticating Gateways; http://www.openbsd.org/faq/pf/authpf.html

Analysis Process

OpenBSD sort(1) manual page; http://www.openbsd.org/cgi-bin/man.cgi?query=sort&apropos=0&sektion=1&manpath=OpenBSD+3.5&arch=i386&format=html

Maarten Van Horenbeeck; GCIA Practical Assignment; http://www.giac.org/practical/GCIA/Maarten_Vanhorenbeeck_GCIA.pdf

Appendices

Appendix: SSH session alerted as Red Worm

```
04/08-23:45:06.617679  [**] High port 65535 tcp - possible Red Worm - traffic
[**] 141.157.102.155:65535 -> MY.NET.60.16:22
04/08-23:45:06.631306  [**] High port 65535 tcp - possible Red Worm - traffic
[**] MY.NET.60.16:22 -> 141.157.102.155:65535
04/08-23:45:08.446529  [**] High port 65535 tcp - possible Red Worm - traffic
[**] 141.157.102.155:65535 -> MY.NET.60.16:22
04/08-23:45:08.461483  [**] High port 65535 tcp - possible Red Worm - traffic
[**] MY.NET.60.16:22 -> 141.157.102.155:65535
04/08-23:45:08.653856  [**] High port 65535 tcp - possible Red Worm - traffic
[**] 141.157.102.155:65535 -> MY.NET.60.16:22
04/08-23:45:08.794017  [**] High port 65535 tcp - possible Red Worm - traffic
[**] 141.157.102.155:65535 -> MY.NET.60.16:22
04/08-23:45:08.955410  [**] High port 65535 tcp - possible Red Worm - traffic
[**] 141.157.102.155:65535 -> MY.NET.60.16:22
04/08-23:45:09.535802  [**] High port 65535 tcp - possible Red Worm - traffic
[**] 141.157.102.155:65535 -> MY.NET.60.16:22
04/08-23:45:09.761137  [**] High port 65535 tcp - possible Red Worm - traffic
[**] 141.157.102.155:65535 -> MY.NET.60.16:22
04/08-23:45:10.001812  [**] High port 65535 tcp - possible Red Worm - traffic
[**] MY.NET.60.16:22 -> 141.157.102.155:65535
```

Appendix: Perl scripts to import data into MySQL database

These scripts are basically written by Jason Lam. You can find the originals at http://www.giac.org/practical/Jason_Lam_GCIA.doc. I hacked them up, because some of the file formats had changed slightly.

They do not do sufficient error checking, and there is no real date parsing. The regular expressions used are very bad. But it worked for me. Basically, the same script is used four times, but with a different regular expression and a different table layout. You need the DBI module for it. Use at your own risk.

Import alerts

```
#!/usr/bin/perl -w
# Based on work by Jason Lam

#--[ Settings ---------------
$driver      = "mysql";
$database    = "giac";
$host        = "localhost";
$user        = "giac";
$password    = "mekmitasdigoat";
$tablename   = "alert";
$year        = "2004";
#----------------------------

use DBI;

$file = $ARGV[0];
open ALERT, $file or die "Could not open file";

$dsn = "DBI:$driver:database=$database;host=$host;";
$dbh = DBI->connect($dsn, $user, $password) or die "Database error";
```

```
$tablecreate = "CREATE TABLE IF NOT EXISTS $tablename ("
            . "`id`        INT NOT NULL AUTO_INCREMENT,"
            . "`datestamp`        DATETIME NOT NULL,"
            . "`attack`    TEXT NOT NULL,"
            . "`srchost`   TEXT NOT NULL,"
            . "`srcport`   INT NOT NULL,"
            . "`dsthost`   TEXT NOT NULL,"
            . "`dstport`   INT NOT NULL,"
             . "PRIMARY KEY (`id`) )";

my $query = $dbh->prepare($tablecreate) or die "Could not prepare statement";
$query->execute() or die "Could not create table";

$query = $dbh->prepare("DELETE FROM $tablename") or die "Could not delete
records from table";
$query->execute();

while ($line=<ALERT>) {
        if ($line =~ /^(\d+)\/(\d+)-
(\d+:\d+:\d+).\d+\s+\[\*\*\]\s([\s\W\w]+)\s\[\*\*\]\s([\d\w.]+):(\d+)\s-
>\s([\d\w\W.]+):(\d+)\s$/) {
                # Rules alert
                # $1 = Month, $2=date, $3=time, $4 = attack, $5 = srcip, $6 =
srcport, #7 = dstip, #8 = dstport
             $sql  = "INSERT INTO $tablename (datestamp, attack, srchost,
srcport, dsthost, dstport) VALUES (";
             $sql .= "'$year-$1-$2 $3', '$4', '$5', $6, '$7', $8)";
             $query = $dbh->prepare($sql) or die "Couldn't prepare statement: "
. $dbh->errstr . "\nQuery string was: $sql\n";
             $query->execute();
        }
}
```

Import port scans

```
#!/usr/bin/perl -w
# Based on work by Jason Lam

#--[ Settings ---------------
$driver         = "mysql";
$database       = "giac";
$host           = "localhost";
$user           = "giac";
$password       = "mekmitasdigoat";
$tablename      = "portscan";
$year           = "2004";
#----------------------------

use DBI;

$file = $ARGV[0];
open ALERT, $file or die "Could not open file";

$dsn = "DBI:$driver:database=$database;host=$host;";
$dbh = DBI->connect($dsn, $user, $password) or die "Database error";

$tablecreate = "CREATE TABLE IF NOT EXISTS $tablename ( "
            . "`id`        INT NOT NULL AUTO_INCREMENT, "
            . "`datestamp`        DATETIME NOT NULL, "
            . "`src`       VARCHAR(20) NOT NULL, "
            . "`hosts`             INT NOT NULL, "
```

```
            . "`totaltime`       INT NOT NULL, "
            . "`tcp`             INT NOT NULL, "
            . "`udp`       INT NOT NULL, "
            . "PRIMARY KEY (`id`) )";

my $query = $dbh->prepare($tablecreate) or die "Could not prepare statement";
$query->execute() or die "Could not create table";

$query = $dbh->prepare("DELETE FROM $tablename") or die "Could not delete
records from table";
$query->execute();

while ($line=<ALERT>) {
      if ($line =~ /^(\d+)\/(\d+)-(\d+:\d+:\d+).\d+\s+\[\*\*\]
spp_portscan:\sEnd of portscan from ([\d\S]*.[\d\S]*.[\d\S]+.[\d\S]+): TOTAL
time\((\d+)s\) hosts\((\d*)\) TCP\((\d+)\) UDP\((\d+)\)/ ) {
                  # Portscan
            # $1 = Month, $2=date, $3=time, $4 = srcip, $5 = nohost, $6 =
totaltime, $7 = TCPno, $8 = UDPno
            $sql  = "INSERT INTO $tablename (datestamp, src, totaltime, hosts,
tcp, udp) VALUES (";
            $sql .= "'$year-$1-$2 $3', '$4', $5, $6, $7, $8)";
               $query = $dbh->prepare($sql) or die "Couldn't prepare
statement: " . $dbh->errstr . "\nQuery string was: $sql\n";
            $query->execute();
      }
}
```

Import UDP scans

```
#!/usr/bin/perl -w
# Based on work by Jason Lam

#--[ Settings ---------------
$driver         = "mysql";
$database       = "giac";
$host           = "localhost";
$user           = "giac";
$password       = "mekmitasdigoat";
$tablename      = "udpscan";
$year           = "2004";
$month              = "04"; # Yes, very nasty, but it works fine for now
#---------------------------

use DBI;

$file = $ARGV[0];
open ALERT, $file or die "Could not open file";

$dsn = "DBI:$driver:database=$database;host=$host;";
$dbh = DBI->connect($dsn, $user, $password) or die "Database error";

$tablecreate = "CREATE TABLE IF NOT EXISTS $tablename ("
            . "`id`        INT NOT NULL AUTO_INCREMENT,"
            . "`datestamp`        DATETIME NOT NULL,"
            . "`srchost`   TEXT NOT NULL,"
            . "`srcport`   INT NOT NULL,"
            . "`dsthost`   TEXT NOT NULL,"
            . "`dstport`   INT NOT NULL,"
             . "PRIMARY KEY (`id`) )";

my $query = $dbh->prepare($tablecreate) or die "Could not prepare statement";
```

```
$query->execute() or die "Could not create table";

$query = $dbh->prepare("DELETE FROM $tablename") or die "Could not delete
records from table";
$query->execute();

while ($line=<ALERT>) {
       if ($line =~ /^([\w\W]+).*(\d+) (\d+:\d+:\d+) ([\d\w\W.]+):(\d+) ->
([\d\w\W.]+):(\d+) UDP/) {
                # UDP Scan
                # $1 = Month, $2=date, $3=time, $4 = srcip, $5 = srcport, $6 =
dstip, $7 = dstport
             $day = sprintf '%02d', $2;
             $sql  = "INSERT INTO $tablename (datestamp, srchost, srcport,
dsthost, dstport) VALUES (";
             $sql .= "'$year-$month-$day $3', '$4', $5, '$6', $7)";
             $query = $dbh->prepare($sql) or die "Couldn't prepare statement: "
. $dbh->errstr . "\nQuery string was: $sql\n";
             $query->execute();
        }
}
```

Import TCP scans

```
#!/usr/bin/perl -w
# Based on work by Jason Lam

#--[ Settings ---------------
$driver         = "mysql";
$database       = "giac";
$host           = "localhost";
$user           = "giac";
$password       = "mekmitasdigoat";
$tablename      = "tcpscan";
$year           = "2004";
$month              = "04"; # Yes, very nasty, but it works fine for now
#----------------------------

use DBI;

$file = $ARGV[0];
open ALERT, $file or die "Could not open file";

$dsn = "DBI:$driver:database=$database;host=$host;";
$dbh = DBI->connect($dsn, $user, $password) or die "Database error";

$tablecreate = "CREATE TABLE IF NOT EXISTS $tablename ("
            . "`id`        INT NOT NULL AUTO_INCREMENT,"
            . "`datestamp`        DATETIME NOT NULL,"
            . "`srchost`   TEXT NOT NULL,"
            . "`srcport`   INT NOT NULL,"
            . "`dsthost`   TEXT NOT NULL,"
            . "`dstport`   INT NOT NULL,"
            . "`type`             TEXT NOT NULL,"
             . "PRIMARY KEY (`id`) )";

my $query = $dbh->prepare($tablecreate) or die "Could not prepare statement";
$query->execute() or die "Could not create table";

$query = $dbh->prepare("DELETE FROM $tablename") or die "Could not delete
records from table";
$query->execute();
```

```
while ($line=<ALERT>) {
       if ($line =~ /^([\w\W]+).*(\d+) (\d+:\d+:\d+) ([\d\w\W.]+):(\d+) ->
([\d\w\W.]+):(\d+) (.+)/) {
                # UDP Scan
                # $1 = Month, $2=date, $3=time, $4 = srcip, $5 = srcport, $6 =
dstip, $7 = dstport, $8 = type
             $day = sprintf '%02d', $2;
             $sql  = "INSERT INTO $tablename (datestamp, srchost, srcport,
dsthost, dstport, type) VALUES (";
             $sql .= "'$year-$month-$day $3', '$4', $5, '$6', $7, '$8')";
             $query = $dbh->prepare($sql) or die "Couldn't prepare statement: "
. $dbh->errstr . "\nQuery string was: $sql\n";
             $query->execute();
        }
}
```