Global Information Assurance Certification Paper

GIAC Certified Intrusion
Analyst (GCIA)
Practical Assignment
Version 4

Harro Gaastra

Table of Contents

List of Figures

Abstract

This paper is the result of the GIAC GCIA practical assignment version 4.0 The assignment consists of three parts:
In part 1 an executive summary of the paper is given. The aim is to bridge the gap between the technical content and management and help management to make an informed decision.
In part two different detects taken from a set of intrusion detection log files are analysed. Further information is provided about the most active attackers and the most targeted services is given. The analysis is concluded with a set of defensive recommendations.
Part three describes the used platforms and the analysis process used during the process of creating this paper.

Document Conventions

When you read this practical assignment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

`command`	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
`filename`	Filenames, paths, and directory names are represented in this style.
`computer output`	The results of a command and other computer output are in this style
URL	Web URL's are shown in this style.
Quotation	A citation or quotation from a book or web site is in this style.

Part One: Executive Summary

The University has asked for a security audit based on an analysis of log files from their intrusion detection system and other security point devices. This paper and the recommendations are based on 5 days worth of log files from the 14th of November to the 18th of November 2002.
In these 5 days a total of 843 events were registered on the intrusion detection system. From these 843 events a total of 363 events can be described as scanning activity where the internal network is scanned for vulnerable or compromised systems.
Although a lot of worrying and potentially harmful attacks have been detected there is no evidence based on the log information that any of the internal systems have been compromised. To exclude the possibility of infected or compromised systems more information is needed. It is recommended that all attacked systems are investigated for any signs of breach of security.
It is quite clear that a university network is very dependant on its possibility to share information in every conceivable way. The broad array of users with different interests en different demands ask for an open yet secure environment. Open networks bring huge risks, and a balance has to been found between providing services and securing the network to ensure that services can be provided.
The security policy should address general issues like anti-virus solutions, patch management, change procedures, network access and authorized usage of computing and network resources. Each system should then be configured in compliance with the security policy and users should be made aware of the policy.
Based on the analysis of the available log files it is highly recommended that some form of filtering on the border router is established. The solution would be to allow only inbound traffic for those ports that are necessary for network operations and blocking all other incoming traffic. This will reduce the number of attacks on hosts in the protected network and reduce the amount of alerts generated by the intrusion detection system. The intrusion detection sensor should be tuned accordingly.

Due to the lack of information about current defensive measures, established security policy and general network layout it is difficult to give more detailed advice on network and system countermeasures. A more extensive review of the current security policy and defensive measures is recommended.

Part Two – Detailed Analysis

Scenario description

The given scenario is that I have been asked to provide a security audit for a University by analyzing logs from an intrusion detection system and security point devices. For this assignment a set of log files has been selected from the available log files at the website of Sans.org [1].

The following (raw) log files were used for the assignment:
2002.10.14
2002.10.15
2002.10.16
2002.10.17
2002.10.18

These log files contain a total of 7985 packets for analyzing, ranging from the 14th of November to the 18th of November 2002. No other log files are available for this timeframe.

According to the README file accompanying the log files, all the files are a result of a Snort instance running in binary logging mode, so only packets that violated the ruleset of the snort sensor appear in these logs.
IP addresses of the protected network and the checksums in the log files have been changed. Furthermore all SMTP,ICMP, DNS and web traffic has been removed from the logs.

Relational analysis

To be able to analyze the log files, a basic understanding of the network layout is required. The assumed network topology is based on the results of the steps in the analysis process described in part three:

1) All the traffic in the log files is between two distinct network interfaces with the following MAC addresses:
   - 00:00:0c:04:b2:33
   - 00:03:e3:d9:26:c0

   Based on the IEEE OUI assignments both devices are Cisco devices.
2) The Snort sensor must be listening in between these two devices**.**
3) Based on a dump of all IP addresses in the packets with 00:00:0c:04:b2:33 as the source MAC address and a second dump of all IP addresses in the packets with 00:03:e3:d9:26:c0 as the source MAC address the protected network must be behind the second network

[1] The available log files are files from 2002.

interface. The address range of the internal network is 170.129.0.0/16, the external interface seems to be connected to the Internet.

Based on the obtained information the network looks like this

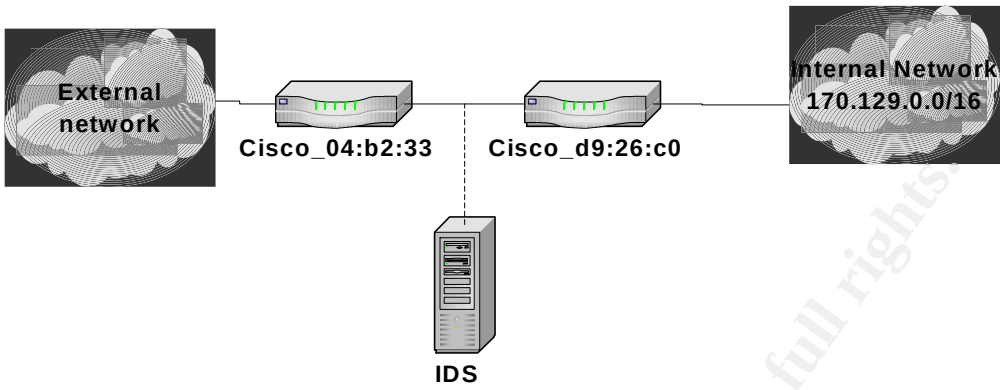


Figure 1: Network Layout

The port information obtained from the log file suggests that device 1 (Cisco_04:b2:33) is a border router. It is unclear whether device 2 has some sort of port filtering enabled.
Further traffic suggests a Web-proxy server in the protected network with IP 170.129.50.120, a Apache web server with IP address 170.129.50.3 and two ftp servers with IP's 170.129.50.4 and 170.129.50.5.

Overview of identified Detects

Using Snort with the http-preprocessor disabled a total of 832 detects were registered:

Detect	Occurances	Classification
P2P Outbound GNUTella client request	2	policy-violation
SHELLCODE x86 inc ebx NOOP	15	shellcode-detect
SHELLCODE x86 NOOP	34	shellcode-detect
SHELLCODE x86 0x90 NOOP unicode	2	shellcode-detect
SHELLCODE x86 unicode NOOP	2	shellcode-detect
SHELLCODE x86 0xEB0C NOOP	2	shellcode-detect
SCAN Proxy Port 8080 attempt	130	attempted-recon
SCAN Squid Proxy attempt	115	attempted-recon
SCAN SOCKS Proxy attempt	114	attempted-recon
BLEEDING-EDGE SCAN NMAP -sA	2	attempted-recon
MISC source port 53 to <1024	1	bad-unknown
SCAN FIN	1	attempted-recon
RPC portmap mountd request UDP	16	rpc-portmap-decode
MISC Tiny Fragments	1	bad-unknown
BAD-TRAFFIC same SRC/DST	46	misc-activity
BACKDOOR Q access	143	misc-activity
BAD-TRAFFIC tcp port 0 traffic	181	misc-activity
(snort_decoder): Tcp Options found with bad lengths	1	attempted-recon
BAD-TRAFFIC ip reserved bit set	23	misc-activity

(snort_decoder) WARNING: TCP Data Offset is less than 5!	3	(null)

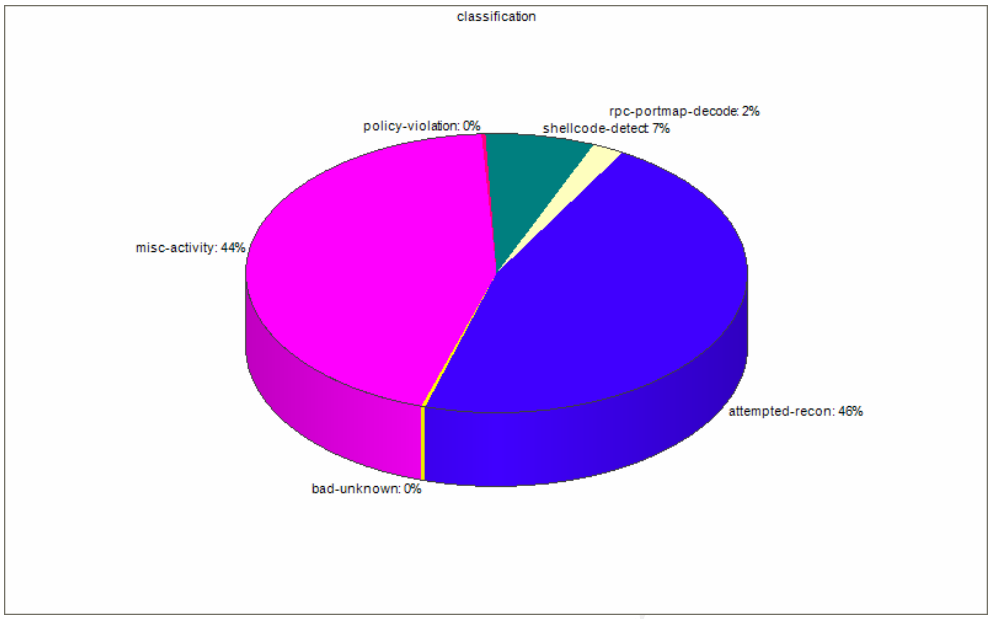


Figure 2: Detects by classification

A large part of the detects is related to scanning activity, where most of the scans originate from two IP addresses (202.108.254.200 and 202.108.254.204). These scans are targeted at possible proxy servers in the protected network.

Based on a the available data the probably most interesting detects are:
a) Backdoor Q access: 143 occurrences in 5 days targeting a large number of different hosts in the protected network.
b) SHELLCODE x86 0xEB0C NOOP: Two detects from two different sources, both aimed towards possible FTP servers.
c) Bad-Traffic TCP port 0 traffic: The large number of 'Bad traffic TCP port 0 traffic' detects are most likely related to scanning activity. All of these detects come from a small range of IP addresses (211.47.255.20-211.47.255.24) and target a total of 12 different IP addresses in the protected network range during 4 days.

Detect 1 : Backdoor Q access

Sample Snort alert (1 of 143)

```
[**] [1:184:6] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
11/14-16:29:14.826507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
255.255.255.255:31337 -> 170.129.172.186:515 TCP TTL:15 TOS:0x0 ID:0 IpLen:20
DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
```

TCPDump output of the packet that triggered the alert:

```
16:29:14.826507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, IPv4, length 60: IP (tos
0x0, ttl  15, id 0, offset 0, flags [none], proto 6, length: 43)
255.255.255.255.31337 > 170.129.172.186.printer: tcp 3
        0x0000:  4500 002b 0000 0000 0f06 5492 ffff ffff  E..+......T.....
        0x0010:  aa81 acba 7a69 0203 0000 0000 0000 0000  ....zi..........
        0x0020:  5014 0000 09ba 0000 636b 6f00 0000       P.......cko...
```

Attack Description

The detects shows that packets are sent from a spoofed IP address (255.255.255.255) to a number of hosts. In the analyzed log files a total of 143 different hosts were targeted during 5 consecutive days[2]. No host was targeted twice in this timeframe.
The alert point to the Q Trojan as target of the detect, the data is consistent with other detects of the Q Trojan.
Each packet has both the ACK and RST flags set and is sent to port 515.
The large number of detects suggests that the network is being scanned for infected hosts; the packets would act as a stimulus for the Q Trojan to perform a certain action based on the data in the packet or the makeup of the packet itself. It is unclear what kind of response this would be.
The timing of the scan is quite slow, on average two packets per hour.
It is unclear why the destination port of 515 (Unix Line Printer Daemon) has been chosen: Although on quite a few UNIX systems this port would be open the attacker could have chosen a port which would stand out less in the traffic. Since the Trojan listens to raw IP packets only, the destination port does not matter.
There is no data available if any of the targeted hosts makes an outbound connection as a result of the detected packets.
Based on the paper by Les Gordon about the Q Trojan the detect is consistent with a pre 2.0 version of the Q Trojan.
No further information about the senders operating system could be obtained from the log files using p0f as a fingerprinting tool.

Reason for selection

A compromised host would pose a severe risk for the network. The fact that traffic originates from 255.255.255.255 stands out from the rest of the detects. Compared to the rest of the detects this detect could have quite an impact on the network. The number of detects and the fact that the same attack has been detected during several days are more reasons to make this detect a prime target for further analysis.

Detect was generated by

The detect was generated by Snort version 2.1.3 using the log files provided. The rule that triggered the detect is:

[2] See Appendix A for a list of targeted hosts

```
alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR Q
access"; dsize:>1; flags:A+; flow:stateless;
reference:arachnids,203; classtype:misc-activity; sid:184;
rev:6;)
```

Packet data was extracted from the log file by using TCPdump
`tcpdump -Xnnevvr all.dump 'host 255.255.255.255'`

Possibility the source address was spoofed

It is absolutely certain that the source IP address is spoofed as there is no legitimate traffic originating from any broadcast address, including 255.255.255.255.
The address 255.255.255.255 is reserved for broadcasts on a local network and should never be forwarded. From our tcpdump output it is clear that the packet was send from the external network, the source MAC address is 00:03:e3:d9:26:c0 which is the border router.
Evidence that the packet is crafted is given by the source port (31337, the 'eleet' port), the TTL's of 14 and 15 and the fact that both the ACK and RST flags are set. Furthermore each packet has a IP ID of zero, without the DF flag set.

Attack Mechanism

The Q Trojan affects mostly UNIX systems (a Windows port has been created) and uses raw IP packets to communicate. All versions of Q are created as a client/server pair, where different client server/pairs are unable to establish a connection. The Q 'remote access tool' has been developed by Mixter and available from his web site. The latest version at the time of writing this paper is 2.4
The server component will listen for any traffic on an infected host and will initiate an action based on the packet send by the client component. The actions include executing privileged commands and opening shells on certain ports. In this case it is unclear what the response to the detected packets would be.
Since the Trojan listens to the raw IP packets only the Q messenger does not require a response. With a broadcast address like 255.255.255.255 as the source IP and by setting the reset flag on the packet the attacker ensures that there will be no response from the Q server. The data send with the packet might be used as a trigger for an action by the server component.

Correlations

During my investigation into the workings of the Q Trojan is ran into a paper about the Q trojan from Les Gordon, available at
www.sans.org/resources/idfaq/qtrojan.php
A CVE for the Q Trojan is available at
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660

Evidence of Active Targeting

The high number of targets in the analysed log files does not suggest that the attacker is actively targeting. It appears to be a scan of large parts of the protected network 170.129.0.0/24. A short look at older available log files shows that the scanning has been going on for quite some time.

Severity

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

The severity for this detect is (2+4) – (1+2) = 3

Criticality (2): There is no evidence suggesting other connections to the targeted systems than the spoofed packets from 255.255.255.255. Furthermore there are no logs which show connections from the targeted hosts. The fact that the role of the target systems is unclear adds a point to criticality
Lethality (4): Although it is unclear what the reaction from an infected system would be, the target of the Q Trojan is to provide root level access on the infected host. Any compromised host would be a severe risk to the infected machine and probably the entire network. One point is deduced for the fact that it is not a targeted attack
System countermeasures (1): It is unclear whether any countermeasures are present on the targeted hosts.
Network countermeasures (2): The fact that traffic to port 515 from a source address with 255.255.255.255 passes through the router suggests that it is an open infrastructure. The fact that no NETBIOS (ports 137-139) traffic is detected suggests that at least some filtering mechanism is available. The packet was detected by an intrusion detection sensor, so some network defense is present.

Detect 2: SHELLCODE x86 0xEB0C NOOP

Sample Detect (one of two detects)

```
[**] [1:1424:6] SHELLCODE x86 0xEB0C NOOP [**]
[Classification: shellcode-detect] [Priority: 1]
11/17-14:54:27.776507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
len:0x23E
165.154.7.2:1982 -> 170.129.50.4:21 TCP TTL:46 TOS:0x0 ID: 35277
IpLen:20 DgmLen:560 DF
***AP*** Seq: 0x473ADE51  Ack: 0x719E0279  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1055711 3964041
```

TCPDump output of packet that triggered the alert:

```
14:54:27.776507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, IPv4, length 574: IP
(tos 0x0, ttl  46, id 35277, offset 0, flags [DF], proto 6, length: 560)
165.154.7.2.1982 > 170.129.50.4.ftp: tcp 508
        0x0000:  4500 0230 89cd 4000 2e06 37d9 a59a 0702  E..0..@...7.....
        0x0010:  aa81 3204 07be 0015 473a de51 719e 0279  ..2.....G:.Qq..y
        0x0020:  8018 7d78 282a 0000 0101 080a 26ad 2644  ..}x(*......&.&D
        0x0030:  0045 e5c8 4357 4420 3030 3030 3030 3030  .E..CWD.00000000
        0x0040:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
```

```
0x0050:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x0060:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x0070:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x0080:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x0090:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x00a0:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x00b0:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x00c0:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x00d0:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x00e0:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x00f0:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x0100:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x0110:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x0120:  3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
0x0130:  3030 3030 3030 3030 f0fc 4031 0708 985f  00000000..@1..._
0x0140:  0808 eb0c eb0c eb0c eb0c eb0c eb0c eb0c  ................
0x0150:  eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c  ................
0x0160:  eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c  ................
0x0170:  eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c  ................
0x0180:  eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c  ................
0x0190:  eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c  ................
0x01a0:  eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c  ................
0x01b0:  eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c  ................
0x01c0:  eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c  ................
0x01d0:  eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c  ................
0x01e0:  eb0c eb0c eb0c eb0c 9090 9090 9090 9090  ................
0x01f0:  9090 9090 31db 43b8 0b74 510b 2d01 0101  ....1.C..tQ.-...
0x0200:  0150 89e1 6a04 5889 c2cd 80eb 0e31 dbf7  .P..j.X......1..
0x0210:  e3fe ca59 6a03 58cd 80eb 05e8 ed0a ca59  ...Yj.X........Y
0x0220:  6a03 58cd 80eb 05e8 edff ffff ffff ff0a  j.X.............
```

Attack Description

Two attacks took place in the selected timeframe from 14th of november to the 18th of November 2002. On the 16th the first detect is made: at 16:41 Shellcode is detected in a packet originating from 163.24.239.8. The second detect is made on the 17th at 13:54 and shows 165.154.7.2 as the source address.
Both packets trigger a “SHELLCODE x86 0xEB0C NOOP” alert in snort. Looking at all the packets originating from the two source addresses both show two additional packets sent to the server. The short interval between these packets (less then a second) shows that some form of automation or scripting is used. The following listing shows the two additional packets from the second detect:

```
14:54:27.836507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, IPv4, length 82: IP (tos
0x0, ttl  46, id 35321, offset 0, flags [DF], proto 6, length: 68)
165.154.7.2.1982 > 170.129.50.4.ftp: tcp 16
        0x0000:  4500 0044 89f9 4000 2e06 3999 a59a 0702  E..D..@...9.....
        0x0010:  aa81 3204 07be 0015 473a e04d 719e 0482  ..2.....G:.Mq...
        0x0020:  8018 7d78 151b 0000 0101 080a 26ad 264a  ..}x........&.&J
        0x0030:  0045 e5cd 4357 4420 7e2f 7b2e 2c2e 2c2e  .E..CWD.~/{.,.,.
        0x0040:  2c2e 7d0a                                ,.}.
14:54:28.186507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, IPv4, length 73: IP (tos
0x0, ttl  46, id 35599, offset 0, flags [DF], proto 6, length: 59)
165.154.7.2.1982 > 170.129.50.4.ftp: tcp 7
        0x0000:  4500 003b 8b0f 4000 2e06 388c a59a 0702  E..;..@...8.....
        0x0010:  aa81 3204 07be 0015 473a e0b5 719e 05b9  ..2.....G:..q...
        0x0020:  8018 7d78 85b3 0000 0101 080a 26ad 266e  ..}x........&.&n
        0x0030:  0045 e5f2 4357 4420 7e7b 0a              .E..CWD.~{.
```

A first investigation into the packets show that both target the FTP port (21) on two different systems in the protected network. By looking for more clues about the detects using Google shows the same packets being detected by older versions of snort as a “FTP EXPLOIT CWD overflow”.
These detects point to a vulnerability in the Vermillion FTP server version 1.23 by Arcane Software. To exploit the vulnerability the attacker has to send three packets with a payload length of 504 bytes to the server. The log files show that in fact three packets are sent to the server but the first packet has a payload of only 508 bytes, the second just 16 bytes an third 7 bytes. This rules out that this attack is a DOS attack against the Vermillion FTP server.

A discussion about the same attack point to another possible cause for the packet: the 7350wurm exploit
The source code for this exploit is widely available. A comparison between the dump of the first packet and part of the source code shows a match in the shellcode being sent to the server:

Packet
```
0x01f0:  9090 9090 31db 43b8 0b74 510b 2d01 0101
0x0200:  0150 89e1 6a04 5889 c2cd 80eb 0e31 dbf7
0x0210:  e3fe ca59 6a03 58cd 80eb 05e8 ed0a ca59
ox0220:  6a03 58cd 80eb 05e8 edff ffff ffff ff0a
```

Shellcode from 7350wurm.c
```
unsigned char   x86_wrx[] =
        "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
        "\x31\xdb\x43\xb8\x0b\x74\x51\x0b\x2d\x01\x01\x01"
        "\x01\x50\x89\xe1\x6a\x04\x58\x89\xc2\xcd\x80\xeb"
        "\x0e\x31\xdb\xf7\xe3\xfe\xca\x59\x6a\x03\x58\xcd"
        "\x80\xeb\x05\xe8\xed\xff\xff\xff";
```

The source code reveals that this is an exploit targeted at the WU-ftpd daemon to gain remote root access. The other detected packets are consistent with the characteristics of this exploit.

Reason for selection

The analysed log files contain quite a few shellcode detects, with two detects by two different sources targeting two different systems in the protected network standing out from the rest. Snort itself gives the detect a high priority (priority 1) which make the detects very interesting for some further analysis.

Detect was generated by

The detect was generated by Snort version 2.1.3 using the selected log files. The rule that triggered the detect is:
```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 0xEB0C NOOP"; content:"|EB 0C EB 0C EB 0C EB
0C EB 0C EB 0C EB 0C EB 0C|"; classtype:shellcode-detect;
sid:1424; rev:6;)
```

Packet data was extracted from the log file by using TCPdump
`tcpdump -Xnnevvr all.dump 'src host 165.154.7.2'`

Possibility the source address was spoofed

Since the attacker tries to gain root access at the server, he/she will not want to spoof the IP address in the first place. In all the packets the ACK|PUSH flags are set, which would indicate an established connection with the servers. The sequence number of the packets in question seem consistent for both detects, as well as the IP ID's.
The first detect from 163.25.239.8 shows a TTL of 44, the second detect coming from 165.154.7.2 shows a TTL of 46 for all three packets.
Based on these characteristics it is highly unlikely that the IP address was spoofed.

Attack Mechanism

The WU-FTP daemon has globbing[3] capabilities which allow users to specify filenames and locations the same way as it is done in shells. The globbing code in WU-FTPD is vulnerable due to a bug in handling certain commands starting with '~{.'
The attacker can exploit this vulnerability by placing shellcode in the right locations of the heap using FTP commands and then sending a command which would be handled by the vulnerable globbing code. This would cause WU_FTP to execute arbitrary code with the privileges of WU-FTPD, which in most cases would be root.

Correlations

CVE entry for buffer overflow in Vermillion FTP Daemon VFTPD 1.23
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-1058
CVE Entry for globbing vulnerability in WU-FTPD
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0550
CERT has an advisory for the vulnerability at:
http://www.cert.org/advisories/CA-2001-33.html
Both IP addresses are registered in the Dshield databases at
http://www.dshield.org
Source code for the 7350 exploit:
(http://packetstormsecurity.nl/removed/7350wurm.c)

Evidence of Active Targeting

The two attacks are clearly targeted. In both cases a single IP was targeted in the protected network. There is no evidence for other activity originating from these IP addresses.

[3] For more information about 'globbing': http://www.cert.org/advisories/CA-2001-07.html

Severity

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

The severity for this detect is (4+5) – (1+2) = 6

Criticality (4): Since this is a FTP server, criticality is high. We do not know the contents or use of the FTP server, so in my view this would give it a criticality of 4.
Lethality (5): The attaker tries to gain root access to the server, which gives it a lethality rating of 5 immediately.
System countermeasures (1): There is no evidence for any countermeasures on the server and we do know based on the way the exploit works that anonymous access is enabled on the server. There is no evidence that suggests the system was or was not compromised.
Network countermeasures (2): We see from the rest of the log files that the border router is hardly filtering anything, but we do have an IDS in place which did detect the shellcode.

Detect 3: Bad Traffic TCP port 0 traffic

Sample set of Snort alerts

```
[**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/15-14:36:26.406507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x42
211.47.255.24:41866 -> 170.129.195.40:0 TCP TTL:46 TOS:0x0 ID:0 IpLen:20
DgmLen:52 DF
******S* Seq: 0xD8010CF5  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/15-14:36:29.296507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x42
211.47.255.24:41866 -> 170.129.195.40:0 TCP TTL:46 TOS:0x0 ID:0 IpLen:20
DgmLen:52 DF
******S* Seq: 0xD8010CF5  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/15-14:36:35.286507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x42
211.47.255.24:41866 -> 170.129.195.40:0 TCP TTL:46 TOS:0x0 ID:0 IpLen:20
DgmLen:52 DF
******S* Seq: 0xD8010CF5  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/15-14:36:47.306507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x42
211.47.255.24:41866 -> 170.129.195.40:0 TCP TTL:46 TOS:0x0 ID:0 IpLen:20
DgmLen:52 DF
******S* Seq: 0xD8010CF5  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
```

TCPDump output of packets that triggered the alerts:

```
14:36:26.406507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, IPv4, length 66: IP (tos
0x0, ttl  46, id 0, offset 0, flags [DF], proto 6, length: 52)
211.47.255.24.41866 > 170.129.195.40.0: tcp 0
        0x0000:  4500 0034 0000 4000 2e06 0cd2 d32f ff18  E..4..@....../..
        0x0010:  aa81 c328 a38a 0000 d801 0cf5 0000 0000  ...(............
        0x0020:  8002 16d0 8fd4 0000 0204 05b4 0101 0402  ................
        0x0030:  0103 0300                                ....
14:36:29.296507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, IPv4, length 66: IP (tos
0x0, ttl  46, id 0, offset 0, flags [DF], proto 6, length: 52)
211.47.255.24.41866 > 170.129.195.40.0: tcp 0
        0x0000:  4500 0034 0000 4000 2e06 0cd2 d32f ff18  E..4..@....../..
        0x0010:  aa81 c328 a38a 0000 d801 0cf5 0000 0000  ...(............
        0x0020:  8002 16d0 8fd4 0000 0204 05b4 0101 0402  ................
        0x0030:  0103 0300                                ....
14:36:35.286507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, IPv4, length 66: IP (tos
0x0, ttl  46, id 0, offset 0, flags [DF], proto 6, length: 52)
211.47.255.24.41866 > 170.129.195.40.0: tcp 0
        0x0000:  4500 0034 0000 4000 2e06 0cd2 d32f ff18  E..4..@....../..
        0x0010:  aa81 c328 a38a 0000 d801 0cf5 0000 0000  ...(............
        0x0020:  8002 16d0 8fd4 0000 0204 05b4 0101 0402  ................
        0x0030:  0103 0300                                ....
14:36:47.306507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, IPv4, length 66: IP (tos
0x0, ttl  46, id 0, offset 0, flags [DF], proto 6, length: 52)
211.47.255.24.41866 > 170.129.195.40.0: tcp 0
        0x0000:  4500 0034 0000 4000 2e06 0cd2 d32f ff18  E..4..@....../..
        0x0010:  aa81 c328 a38a 0000 d801 0cf5 0000 0000  ...(............
        0x0020:  8002 16d0 8fd4 0000 0204 05b4 0101 0402  ................
        0x0030:  0103 0300
```

Attack Description

The analysed log files show a total of 181 detects during 4 days (there were no detects on the 14th of November), in which 12 different IP addresses were targeted by 5 different sources.

The following figure shows the number of scans per day:

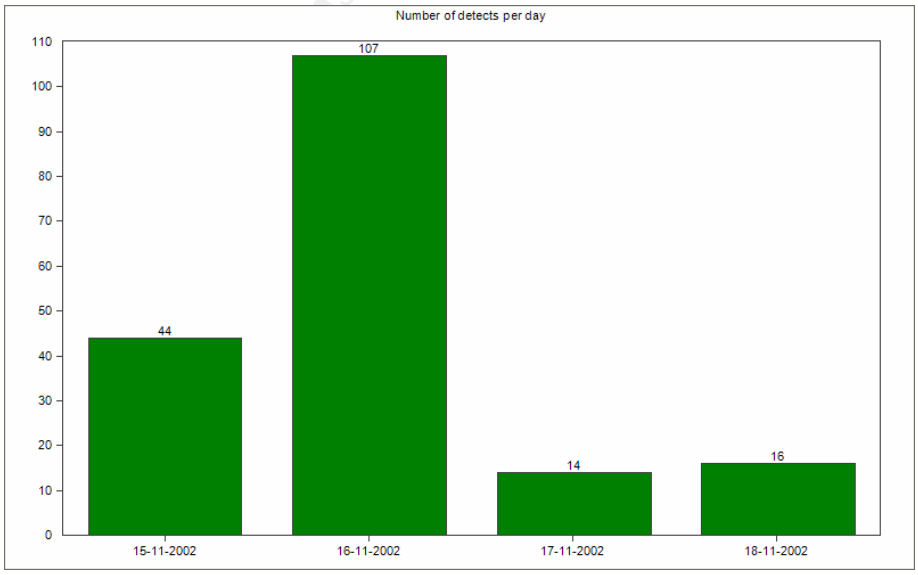


Figure 3: Number of scans per day

In all cases the packets come in groups of four packets towards the same target and are send in intervals of 3, 6 and 12 seconds. The interval matches the respected behavior of retry packets and indicates that the attacker did not get a response from the targeted IP's.
After 4 packets the source port is changed. In each case a single IP is targeted by the same IP with 4 groups of 4 packets. No IP was targeted from more than one source address.
All packets are send with an IP ID of zero and the Don't Fragment flag is set which is abnormal behavior, as each packet should have a unique IP ID. The IP ID of zero and the destination port of zero are a fair indication that the packets are crafted.
The fact that the packets are send with the retry intervals indicate that the attacker did not get a response from the target. This means that either the packets have been silently dropped by the second router or that the host did not respond.

Reason for selection

The reason for the selection of these detects is the fact that there are a total of 181 detects against a range of hosts in the 170.129.0.0/16 subnet. There are a total of 4 different source addresses for the packets, all belonging to the same subnet. There is no legitimate reason for sending packets with a destination port of zero, so the chance of these detects being false positives is relatively small.

Detect was generated by

Once more the detect was generated by Snort version 2.1.3 using the log files provided. The rule that triggered the detect is:

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg: "BAD-TRAFFIC tcp
port 0 traffic"; flow:stateless; classtype:misc-activity;
sid:524; rev:8;)
```

Possibility the source address was spoofed

As this is a clear example of reconnaissance activity the chance of the source address being spoofed is very small. When passive OS fingerprinting is the goal, the attacker will need an answer to use with the fingerprinting tool.

Attack Mechanism

The reason for the packets seems to be OS fingerprinting. Port zero is a reserved port per rfc 1700 and no normal application should be generating this kind of traffic. In addition to the destination port being zero the IP ID of zero clearly indicates that some form of packet crafting has been done.
The reason for sending a packet to a reserved port is to trigger a response from the receiving host. In case of a reserved port the answer would be a RESET. As different operating systems will respond differently, the attacker can then use

passive fingerprintinging techniques to gain information about the operating system of the victim.
For more information about passive fingerprinting techniques
One of the tools able to craft such a packet is hping2, which uses port zero as the default target port. The following hping2 command would generate a packet similar to the detected packets:

```
hping –N 0 –S –w 5840 –y 170.129.X.X
```

Although the analysis points to hping2 as the source of these packets the packets are not generated by this tool as hping2 lacks the possibility to generate the retry characteristics as shown in the logs. This means that some other packet crafting tool is used. It is highly unlikely that a script is used in conjunction with hping2, as there is nothing to gain from sending multiple packets during a reconnaissance like this.

Correlations

More information about the hping2 utility can be obtained from the hping website at http://www.hping.org
Information about passive os fingerprinting and p0f is available at http://lcamtuf.coredump.cx/p0f.shtml

Evidence of Active Targeting

In the detected logs there is evidence that a total of 12 destinations were targeted in the protected network by 5 different source addresses ranging from 211.47.255.20 to 211.47.255.24. No further traffic from these addresses can be seen in the logs.
The wide range of targets would indicate a broad ‘low and slow’ scan and that no active targeting is taking place

Severity

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

The severity for this detect is (2+1) – (2+2) = -1

Criticality (2): There is not enough information to determine the role of the target systems: this adds a point to criticality.
Lethality (1): Since the attacker just gathers information and the targets did not respond to the probes lethality is set to 1.
System countermeasures (2): Once again, there is not much information to know anything about the systems countermeasures. The fact that there is no evidence of a response would give it at least an additional point on system countermeasures.
Network countermeasures (2): The border router is not filtering any traffic from port zero or isn’t filtering anything at all. The packet might have been dropped by a firewall behind the border router and the IDS, but there is not enough

information to be sure about that. The probe was detected by an intrusion detection sensor, so some network defense is present.

Network Statistics

Top Five talkers

#	Source IP	Number of events	Distinct events
1	202.108.254.200	221	3
(2)	(255.255.255.255)	(143)	(1)
2	202.108.254.204	86	3
3	211.47.255.24	59	1
4	129.118.2.10	49	2
5	211.47.255.20	45	1

The top 5 talkers is based upon the total number of events in the database. The number of detects originating from IP 255.255.255.255 has been singled out, as these detects are definitely from a spoofed IP address.

Top Five targeted Services

#	Destination Port	Number of detects	Known services
1	515	143	(tcp) printer spooler (udp) printer spooler (tcp) lpdw0rm [trojan] lpdw0rm (tcp) Ramen [trojan] Ramen
2	8080	130	(tcp) http-alt HTTP Alternate (see port 80) (udp) http-alt HTTP Alternate (see port 80) (tcp) BrownOrifice [trojan] Brown Orifice (tcp) BrownOrifice trojan] Brown Orifice (tcp) Genericbackdoor [trojan] Generic backdoor (tcp) RemoConChubo [trojan] RemoConChubo (tcp) ReverseWWWTunnel [trojan] Reverse WWW Tunnel Backdoor (tcp) RingZero [trojan] RingZero Often used as proxy server
3	3128	115	(tcp) squid-http Proxy Server (tcp) ReverseWWWTunnel [trojan] Reverse WWW Tunnel Backdoor (tcp) RingZero [trojan] RingZero (tcp) RingZero [trojan] RingZero
4	1080	114	(tcp) socks Proxy Server (udp) socks Proxy Server (tcp) SubSeven2.2 [trojan] SubSeven 2.2 (tcp) WinHole [trojan] WinHole (tcp) WinHole [trojan] WinHole
5	63414	49	None

Once more the results are based on the detected events in the snort database.

Top 3 suspicious external source adresses

Two of the most suspicious external source addresses would be both the addresses which use the 7350wurm exploit in order to compromise FTP servers:

1) 165.154.7.2 : WU_FTPD exploit
http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00260.html shows some active scanning for FTP servers from the same source address. There are several other (Japanese) logs which show activity from this address

2) 163.24.239.8 : WU_FTPD exploit
No further information about this IP address has could be found

The third place goes to 202.108.254.200 and 202.108.254.204 for their massive scanning efforts: In total these two IP's are responsible for more than 300 scans targeting over 100 different hosts in the protected network. They are actively scanning for open proxies on several ports and triggering several alerts:

1080: SCAN Socks Proxy attempt
3128: SCAN Squid Proxy attempt
8080: SCAN proxy port 8080 attempt

Correlations from previous practicals

There are quite a few GCIA practicals and discussions about the Q Trojan, most of which describe similar attacks and have similar findings.
In the discussion about his detect (http://cert.uni-stuttgart.de/archive/intrusions/2004/01/msg00148.html) ,Mike Shannon draws conclusions which are similar to mine. Pete Storm does a nice job in his practical (http://www.giac.org/practical/GCIA/Pete_Storm_GCIA.pdf) in rounding up all rumors about the Q Trojan and agrees with my findings that the detect may be a Trojan trigger.
Of course there is the very extensive paper by Les Gordon available at http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.pdf

Concerning the Shellcode / WU-FTPD exploit a few papers have been committed to GIAC.
Ronny Rieveld has a few very interesting links in his paper about the 3750wurm exploit (http://www.giac.org/practical/GCIA/Ronny_Rietveld_GCIA.pdf)

There are several GCIA practical postings available which have an analyses of the port Zero detects. Eric Evans's GIAC GCIA Practical assignment 3.4 available at http://www.giac.org/practical/GCIA/Eric_Evans_GCIA.pdf has some interesting information about the use of port 0 by the Linux operating system.
For further correlation see
GCIA practical detect by Barbara Morgan
http://www.dshield.org/pipermail/intrusions/2002-september/005400.php
GCIA practical detect #1 from Simon Ktung (?)
http://cert.uni-stuttgart.de/archive/intrusions/2003/09.msg00030.html

Insights into internal machines

There are no indications of compromised machines or anomalous activity from systems in the protected network. Since no further data is available, this does not mean that there are no compromised systems. It is therefore highly recommended to check for any compromises as a result of the various detects. Though it is unclear whether any of the two targeted FTP servers was running a vulnerable version of the WU_FTP daemon, they should be thoroughly investigated for any signs of compromise.

Defensive recommendations

A general measure to ensure that all internal systems are kept secure is to establish a security policy for these systems and to enforce and regularly review the policy.
The first defense would be to block inbound ports at the border router. In the analysis it is clearly visible that packets on several ports pass through the border router without being filtered. A general solution is to allow only inbound traffic for those ports that are necessary for network operations, for example HTTP (port 80), FTP (20,21) DNS (port 53) etc. The router should be configured to block all inbound traffic originating from reserved addresses like broadcast addresses. The number of IDS alerts should decline drastically as a result of this filtering on the border router.
Based on the network layout it is highly probable the border router is not under the control of the owners of the protected network, but is probably owned by an ISP. In this case it is highly recommended to get a hold of the owner of the router and ask them to update it according to the recommendations above.

With the use of laptops and forms of portable media and a large number of users comes the risk of an attack from the inside. The intrusion detection logs do not provide any evidence for these attacks, as the system only monitors traffic between the internal network and the external network, but it is widely recognized that these types of attacks do take place. Therefore and from a 'defense in depth' perspective it is recommended that all systems in the internal network have some kind of port filtering installed and other measures are taken to ensure that none of the systems gets compromised.

- Some sort of port filtering (personal firewall) should be applied on all systems in the protected network.
- All systems should have an up-to-date anti-virus solution
- All systems should be updated with the latest patches
- Disable any unnecessary services
- Critical systems should have some form of host based IDS and a file integrity tool like Tripwire installed.

Other specific countermeasures for the analyzed detects would be:
- Blocking access to port 515 on the border router (Q Backdoor)
- Configuring the border router to drop packets originating from reserved addresses and coming into the network from the outside (Q Backdoor)
- Disabling anonymous access to the FTP servers (SHELLCODE detect)
- Restrict access to the FTP servers to trusted external IP's (SHELLCODE detect)
- Patch vulnerable FTP servers (SHELLCODE detect)
- Configure the border router to drop all packets targeting port zero (Bad Traffic TCP port 0 traffic)

Part Three – Analysis Process

Used platforms

For the analysis of the log files two different systems were used:

1) Windows XP system, with mysql database and custom analysis software
2) Linux based system (Fedora Core 2) with snort 2.1.3 and the latest ruleset from http://www.snort.org installed. To be able to pick up all detects all rules were enabled in the snort configuration file and additional rules from http://www.bleedingsnort.org were added.

TCPdump and ethereal are available on both systems

General actions

To conveniently process the log files the selected log files were merged into a single file using mergecap:

```
Mergecap –w ./all.dump ./2002.10.1*
```

mapping the network

The MAC addresses can be dumped from the log files using the –e switch with tcpdump. Using the following commands all Mac addresses in the log entries are displayed:

```
tcpdump –ner all.dump
```

Explanation of used switches:
-n: Do not convert host addresses to names
-e: Print the link level header for each packet
-r: read from file
This will yield the following result (snipped)

```
16:28:16.826507 00:00:0c:04:b2:33 > 00:03:e3:d9:26:c0,
IPv4, length 645: IP 170.129.50.120.62872 >
64.154.80.45.http: tcp 591
16:28:17.026507 00:00:0c:04:b2:33 > 00:03:e3:d9:26:c0,
IPv4, length 644: IP 170.129.50.120.62872 >
64.154.80.45.http: tcp 590
[…]
```

Using the '`cut`' command, only parts of each line of the output can be displayed. Cut will cut the output in different field based on a delimiter given by the user using the –d switch. By using the –f switch each individual field can be displayed By piping the tcpdump command to the cut command, a list of the source MAC addresses can be obtained by cutting the file with a single space as delimiter and displaying field number two:

```
tcpdump –ner all.dump | cut –d ' ' –f2
```

By using a third and a fourth command, 'sort' and 'uniq' all unique source MAC addresses can be displayed from this list:

```
tcpdump –ner all.dump | cut –d ' ' –f2 | sort | uniq
```

A similar list of destination MAC addresses can be obtained by letting the cut command displaying only the fourth field

```
tcpdump –ner all.dump | cut –d ' ' –f4 | sort | uniq
```

(The same result can be obtained by loading the file into Ethereal and using the statistics→endpoint list→ethernet command in the menu)

In this case only two MAC addresses can be seen, a quick check using tcpdump filters confirms that there is only traffic between these interfaces:

```
tcpdump –ner all.dump 'ether src 0:0:c:4:b2:33 and ether dst not 00:03:e3:d9:26:c0'
tcpdump –ner all.dump 'ether src 00:03:e3:d9:26:c0 and ether dst not 0:0:c:4:b2:33'
```

As expected, both commands return zero results.
Based on information provided by IEEE OUI assignments, both devices are Cisco devices
The next step in mapping the network is to look at the source IP addresses from which packets are received on the two identified interfaces. Once more tcpdump is used to obtain a list of the source IP addresses, this time a filter is used for the MAC address. After using the cut command, the IP address will be in field 11 of the tcpdump output. Additional steps are necessary to get the IP address, since the source port is also part of field 11. (The field has the format 'IP.port')

```
tcpdump –ner all.dump 'ether src 0:0:c:4:b2:33' | cut –d ' ' –f11 | cut –d '.' –f 1-4 | sort | uniq
tcpdump –ner all.dump 'ether src 0:3:e3:d9:26:c0' | cut –d ' ' –f11 | cut –d '.' –f 1-4 | sort | uniq
```

The same step is repeated for the destination IP addresses on both interfaces (Field 13):

```
tcpdump –ner all.dump 'ether src 0:0:c:4:b2:33' | cut –d ' ' –f13 | cut –d '.' –f 1-4 | sort | uniq
tcpdump –ner all.dump 'ether src 0:3:e3:d9:26:c0' | cut –d ' ' –f13 | cut –d '.' –f 1-4 | sort | uniq
```

By running these commands with filters for both the identified MAC addresses a list with unique IP addresses for each device can be obtained. From the results it can be concluded that there is a single IP subnet behind the 0:3:e3:d9:26:c0 interface (170.129.0.0/16)

To see whether any of the devices filters incoming traffic once more TCPdump is used to show all the ports on which traffic is passed through the first Cisco device:

```
tcpdump –ner all.dump 'ether src 0:3:e3:d9:26:c0' | cut –d ' ' –f13 | cut –d '.' –f 5 | sort | uniq
```

This results in a total of 650 different ports, both low numbered ports (80, 81, 560) and high numbered ports. Based on the number of open ports it is not very likely that this device has filtering rules and is probably a simple router. There is not enough information in the logs to determine whether device 2 has any filtering enabled.

Getting events

Since the information in the log files is obtained from a Snort IDS sensor running in binary logging mode, the first step to obtain further information about any events is done by running the log files through a snort sensor. In this case Snort version 2.1.3 is used[4]. Snort is configured to output to a mysql database
The database is filled using the following command

```
Snort –c /etc/snort/snort.conf –r all.dump –k none –h
170.129.0.0/16
```

Explanation of used switches
-c <file>: Tells snort which config file to load
-r <file>: Read data from file
-k none: checksum mode is set to none: This solves any troubles with mangled checksums in the sanitized log files
-h : Sets the HOME_NET variable to the protected network

To obtain more information about the attackers systems, a passive OS fingerprinting tool was used on the collected data,
OS Fingerprinting
To obtain more information about the operating system of the source of the detects, a passive operating fingerprinting tool named p0f is used.

```
P0f –f /etc/p0f/p0f.fp –s all.dump -N
```

Explanation of used switches
-f <file>: Reads fingerprints from the specified file
-s <file>: Read packets from tcpdump file
-N: Logs only source IP and OS data

Analysis: ACID.NET

For further analysis of the detected events, a self-developed software tool is used: The tool (named ACID.NET) is based on the ACID (Analysis Console for Intrusion Databases) tool and uses the ACID tables in the snort database[5]. ACID itself is quite useful for analyzing and viewing events but it lacks speed and certain features. To overcome these obstacles a piece of custom software has been written in VB.Net.

[4] See 'Used platforms'
[5] The tool has been in development as a small project since January 2004 and is still in Beta.

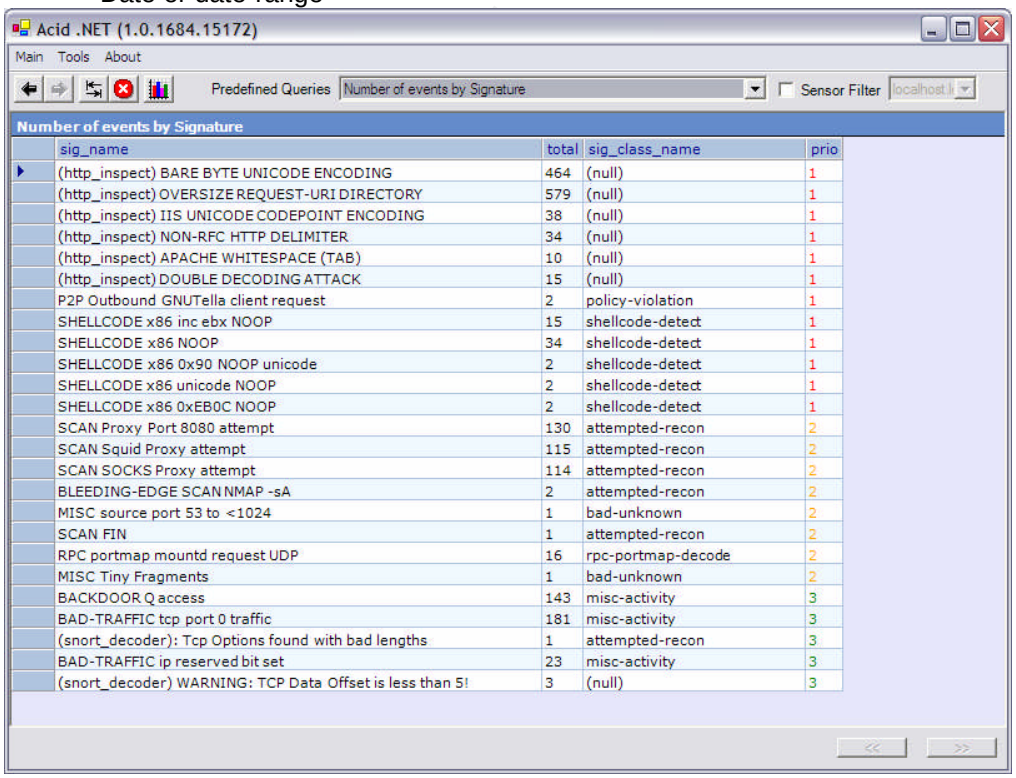
Although much of the functionality is similar to that of the ACID console, the custom software is much faster and easier to configure and more geared towards working with multiple sensors. Both textual and graphical displays of data can be generated using the software.

The statistics, overview of the events and the graphics in Part II of this assignment were generated by the ACID.net tool.
Besides the opportunity to 'drill down' into the data and to write custom queries, the tool has the following standard queries:

- Number of events by signature and sensor
- Number of events by signature
- Number of events per sensor
- Top source IP addresses
- Top destination IP addresses
- Top source ports
- Top destination ports
- All category 1 events
- All category 2 events
- All category 3 events
- Number of events per day
- Number of events per month
- Number of events by signature classification
- Events where source IP has hit multiple sensors

Standard filters for the queries are

- Sensor
- Number of entries to return
- Date or date-range



sig_name	total	sig_class_name	prio
(http_inspect) BARE BYTE UNICODE ENCODING	464	(null)	1
(http_inspect) OVERSIZE REQUEST-URI DIRECTORY	579	(null)	1
(http_inspect) IIS UNICODE CODEPOINT ENCODING	38	(null)	1
(http_inspect) NON-RFC HTTP DELIMITER	34	(null)	1
(http_inspect) APACHE WHITESPACE (TAB)	10	(null)	1
(http_inspect) DOUBLE DECODING ATTACK	15	(null)	1
P2P Outbound GNUTella client request	2	policy-violation	1
SHELLCODE x86 inc ebx NOOP	15	shellcode-detect	1
SHELLCODE x86 NOOP	34	shellcode-detect	1
SHELLCODE x86 0x90 NOOP unicode	2	shellcode-detect	1
SHELLCODE x86 unicode NOOP	2	shellcode-detect	1
SHELLCODE x86 0xEB0C NOOP	2	shellcode-detect	1
SCAN Proxy Port 8080 attempt	130	attempted-recon	2
SCAN Squid Proxy attempt	115	attempted-recon	2
SCAN SOCKS Proxy attempt	114	attempted-recon	2
BLEEDING-EDGE SCAN NMAP -sA	2	attempted-recon	2
MISC source port 53 to <1024	1	bad-unknown	2
SCAN FIN	1	attempted-recon	2
RPC portmap mountd request UDP	16	rpc-portmap-decode	2
MISC Tiny Fragments	1	bad-unknown	2
BACKDOOR Q access	143	misc-activity	3
BAD-TRAFFIC tcp port 0 traffic	181	misc-activity	3
(snort_decoder): Tcp Options found with bad lengths	1	attempted-recon	3
BAD-TRAFFIC ip reserved bit set	23	misc-activity	3
(snort_decoder) WARNING: TCP Data Offset is less than 5!	3	(null)	3

Figure 4: ACID.NET: Overview of detected events

Besides giving overviews of detects the tool also allows for a more detailed view of single events:

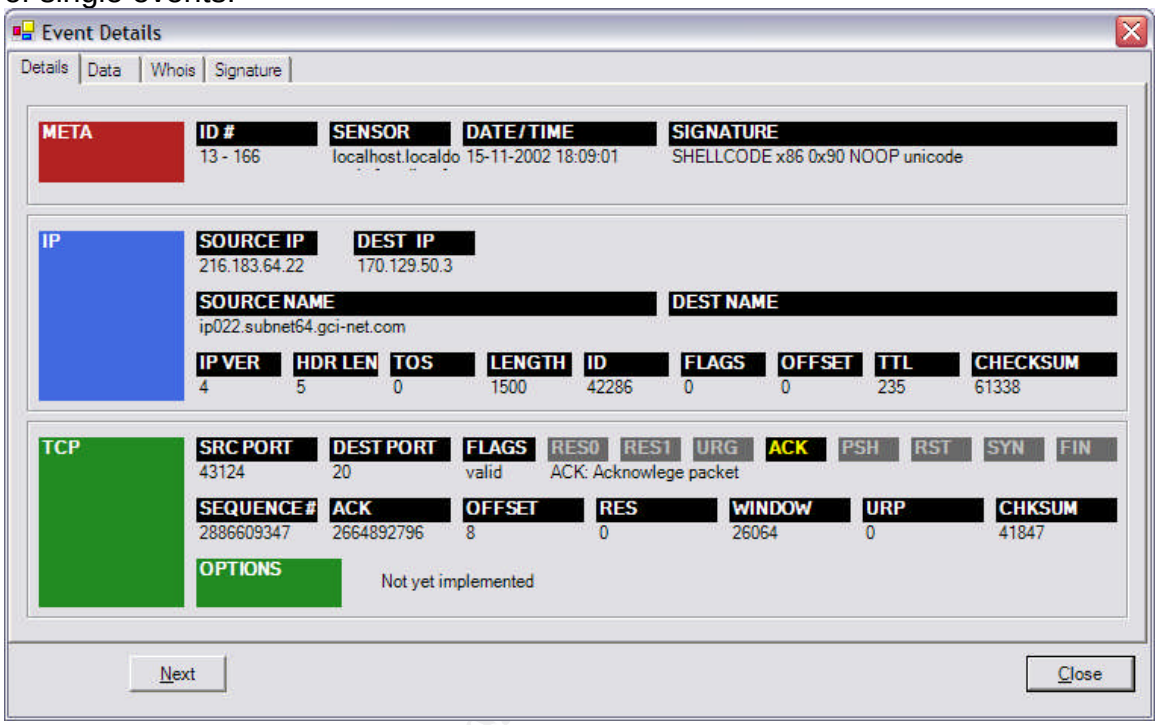


Figure 5: ACID.NET: Example event details for TCP packets

Although the tool is still in beta stage it shows much promise in helping to analyze and correlate different events. In the process of analyzing the data the tool has shown a few shortcomings which will be addressed in the short future:

-the ability to filter all data on a single event type
-the ability to obfuscate Source and Destination Ip addresses in the output

References

Other practicals and related material:

[1] Baxter, John, Discussion of GCIA practical detect, September 2002, http://www.dshield.org/pipermail/intrusions/2002-september/0054
[2] Baltes, Craig, GCIA practical , October 2002 http://www.giac.org/practical/GCIA/Craig_Baltes_GCIA.doc
[3] Shannon, Mike, GCIA practical, march 2004 (http://www.giac.org/practical/GCIA/Mike_Shannon_GCIA.pdf
[4] Storm, Pete, GCIA practical, December 2003 (http://www.giac.org/practical/GCIA/Pete_Storm_GCIA.pdf
[4] Gordon, Les, GCIA practical , December 2002 http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.pdf
[5] Rietveld, Ronny, GCIA practical , January 2003 http://www.giac.org/practical/GCIA/Ronny_Rietveld_GCIA.pdf
[6] Evans, Eric, GCIA practical , January 2004 http://www.giac.org/practical/GCIA/Eric_Evans_GCIA.pdf
[7] Morgan, Barbara, GCIA practical detect , September 2002 http://www.dshield.org/pipermail/intrusions/2002-september/005400,php

Other material

[1] Raw log files available for the scenario, provided by sans.org http://isc.sans.org/logs/raw
[2] Snort Intrusion Detection System, Snort.org, http://www.snort.org
[3] 'Bleeding' snort rules, bleedingsnort.com, http://www.bleedingsnort.org
[4] IEEE OUI assignments, Institute of Electrical and Electronics Engineers website http://standards.ieee.org/regauth/oui/index.shtml
[5] Common vulnerability and exposures database, Mitre.org http://cve.mitre.org/
[6] Gordon, Les, Intrusion Detection Faq, What is the Q Trojan?, 2002 http://www.sans.org/resources/idfaq/qtrojan.php
[7] Mixter, Code for Q 'remote access tool' http://mixter.void.ru/code.html
[9] CERT® Advisory CA-2001-33 Multiple Vulnerabilities in WU-FTPD, CERT.org http://www.cert.org/advisories/CA-2001-33.html
[10] TESO Security, 7350wurm.c source code http://packetstormsecurity.nl/removed/7350wurm.c
[11] hping2 packet crafting tool, hping.org http://www.hping.org
[12] p0f passive OS fingerprinting tool, p0f website http://lcamtuf.coredump.cx/p0f.shtml
[13] Miller, Toby, Rating the enemy, "How to identify the enemy" http://www.koot.biz/docs/overig/how_to_identify_the_enemy.html

[14] Caswell, Brian e.a., Snort 2.0 Intrusion detection, Syngress Publishing Inc, 2003
[15] Northcutt, Stephen e.a., Intrusion Signatures and Analysis, New Riders, 2001
[16] Northcutt, Stephen, Network Intrusion Detection An analyst's handbook, New Riders 1999

Appendix A

Targeted hosts for Q backdoor access			
170.129.26.65	170.129.195.178	170.129.176.42	170.129.4.175
170.129.100.206	170.129.200.206	170.129.178.16	170.129.40.192
170.129.106.120	170.129.201.124	170.129.178.39	170.129.41.171
170.129.112.183	170.129.201.142	170.129.185.19	170.129.45.82
170.129.117.222	170.129.207.122	170.129.185.91	170.129.49.119
170.129.119.210	170.129.214.158	170.129.186.20	170.129.52.209
170.129.129.128	170.129.216.226	170.129.19.190	170.129.53.148
170.129.129.188	170.129.220.126	170.129.19.28	170.129.56.82
170.129.130.130	170.129.222.145	170.129.192.22	170.129.57.163
170.129.137.174	170.129.222.156	170.129.197.57	170.129.57.187
170.129.140.105	170.129.230.201	170.129.2.149	170.129.57.211
170.129.148.109	170.129.1.102	170.129.200.84	170.129.65.138
170.129.148.110	170.129.1.20	170.129.203.58	170.129.65.179
170.129.153.108	170.129.10.221	170.129.205.6	170.129.65.227
170.129.153.131	170.129.103.3	170.129.208.79	170.129.66.181
170.129.153.135	170.129.106.86	170.129.209.67	170.129.68.126
170.129.153.221	170.129.115.50	170.129.209.73	170.129.69.141
170.129.155.128	170.129.119.45	170.129.211.38	170.129.69.158
170.129.156.132	170.129.122.35	170.129.216.96	170.129.70.150
170.129.157.148	170.129.129.38	170.129.22.182	170.129.72.205
170.129.159.157	170.129.13.177	170.129.222.41	170.129.72.94
170.129.161.133	170.129.131.2	170.129.227.19	170.129.73.83
170.129.161.211	170.129.132.79	170.129.227.97	170.129.76.209
170.129.165.132	170.129.134.5	170.129.23.128	170.129.77.14
170.129.165.156	170.129.134.87	170.129.23.133	170.129.77.205
170.129.167.203	170.129.135.94	170.129.23.189	170.129.77.88
170.129.169.107	170.129.140.19	170.129.230.46	170.129.78.58
170.129.172.186	170.129.142.93	170.129.24.44	170.129.79.93
170.129.173.207	170.129.146.14	170.129.25.196	170.129.80.5
170.129.178.195	170.129.146.62	170.129.27.13	170.129.83.228
170.129.181.145	170.129.15.176	170.129.30.34	170.129.89.164
170.129.181.151	170.129.150.38	170.129.32.155	170.129.89.87
170.129.190.188	170.129.156.91	170.129.33.54	170.129.94.129
170.129.190.224	170.129.166.7	170.129.33.72	170.129.94.77
170.129.193.103	170.129.166.76	170.129.38.133	170.129.95.144
170.129.194.187	170.129.171.53	170.129.38.78	

Last Updated: July 22nd, 2017

Upcoming Training

SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201709,	Sep 11, 2017 - Oct 18, 2017	vLive
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Baltimore Fall 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS Scottsdale SEC503	Scottsdale, AZ	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced