SANS Intrusion Detection in Depth
GCIA Practical Assignment
Version 3.2

Etienne Lebel

Sans Ontario – Toronto

May 13 – 18, 2002

Table of Content

Part 1 - Describe the State of Intrusion Detection

Abstract

For this assignment I will look closely at a tool that performs reconnaissance of the remote operating system of a target host. This tool is not new to the community, but uses a reconnaissance technique that no other tools use. It fingerprints operating systems with almost only ICMP packets. The tool that I am talking about is Xprobe[1].

What is OS fingerprinting

Operating system fingerprinting is the ability to identify a remote operating system by analyzing the way it responds to a certain type of packets. It is similar to human fingerprints, because each operating system has its own anomaly in the way it responds to some situations.

History

Xprobe, or X, is a tool designed and actively maintained by Fyodor Yarochkin and Ofir Arkin. It was first released on July 13th, 2001 under the GNU GPL license. A few months later (October), the 0.0.2 version was released. Since that day, a new version of Xprobe called Xprobe2 is under way. The second version of this tool was launched at defcon10 with several additions.

Because the new version is not officially released, release candidate 1, and does not contain as much signature as xprobe-0.0.2, 40 for Xprobe 0.0.2 against 18 for Xprobe2, I will focus my paper on the Xprobe-0.0.2 version.

Introduction

Two categories of tools can be distinguished in the operating system fingerprinting area: passive fingerprinting tools that do not inject packets on the wire (P0F, Siphon, etc.), and active fingerprinting tools that send a number of packets to the target host to determine its operating system (Xprobe, NMAP, Queso, etc.).

Unlike other active fingerprinting tools, Xprobe is based on ICMP. Other tools are based on TCP protocol. They need to generate more packets than Xprobe to achieve their goal. The maximum number of packets that you can inject on the network with Xprobe is 4 toward the target with their 4 replies, for a total of 8

[1] Xprobe - http://www.xprobe.org/

packets. Tools such as NMAP (45 packets) and QUESO (15 packets) are more likely to be noisy than Xprobe. Xprobe is also even more accurate than these two other popular tools.

Methods

All the intelligence behind Xprobe comes from the research of Ofir Arkin in the usage of ICMP in scanning[2]. The goal of Xprobe is to be very stealthy. To achieve this task, it never sends malformed packets to the remote host. It sends packets that are seen on a daily basis on the network, so IDS has a hard time finding them.

Before going any further, we need to know how the intelligence is inserted into Xprobe. All reconnaissance is hard coded into the source file and cannot be modified without recompiling the software. A static tree of decision is built and upon a situation the program executes some other selective tests, depending on the previous results. This way the program does not have to perform each test against all systems, but only those who help it identify an operating system correctly.

Xprobe performs the following verifications:

- ICMP error quoting size

  Represents the number of bits in the ICMP error message of a UDP connection, as described in RFC 1122.

- ICMP error message echoing integrity

  Validates the embedded packet in an ICMP error message for validity in the following fields:
  - IP total length field
  - IP ID
  - 3 fits flags and offset fields
  - IP header checksum
  - UDP header checksum

- Precedence bits

  Checks the Type of service (TOS) and Must Be Zero (MBZ) bytes for validity.

- DF bits echoing

  Is the Don't Fragment bit set in the ICMP error message or was it in the original packet?

- TTL

[2] ICMP Usage in Scanning - http://www.xprobe.org/archive/papers/ICMP_Scanning_v3.0.pdf

Analyzes the response's Time to live

- Code field value

  Validates if the remote operating system changes the Code field in an ICMP echo request. See RFC 792.

- TOS echoing

  Analyzes the value of this field depending on the type of ICMP message.

All these verifications are performed with a UDP packet to a closed port[3], by default 32132, and ICMP Echo request, ICMP Timestamp request, ICMP information request, or ICMP address mask request.

The way Xprobe works is fairly simple. It sends a UDP packet to a closed port and waits for the message of the ICMP port unreachable. After receiving the answer, the program analyzes the received packet, then categorizes the host upon some predefined groups, depending on the result of a test. It then performs another test that will eliminate operating systems in order to obtain the smallest group possible until we send 4 packets, or no other tests are defined.

The major problem with those great methods is that all the logic is stored in the code and not into a database or a text file. So if the behavior of an operating system changes or if we find a new way to identify an operating system, we have to change the original code and recompile the program to add this new definition to it. This problem should be resolved in the new version of Xprobe.

When used to fingerprint a host with a firewall in front of it, Xprobe cannot identify the remote operating system, since the firewall drops the first UDP packet and never sends back an ICMP port unreachable message. Also, if the first UDP packet is sent to a listening UDP, port Xprobe will not be able to identify this host. This is because UDP is a stateless protocol and no answer is given back to the sender. The work around for this last points is that you can use Xprobe with another destination port with the command line option that I will show you later.

Xprobe in action

Now that you know how Xprobe tries to detect a remote operating system, let's look at how it operates in the real world. All tests in this paper are launched from my Mandrake Linux system toward a host that I have on my private LAN. Before doing any test, I downloaded Xprobe from http://www.xprobe.org. Once the package was downloaded, I uncompressed the source file and launched the

[3] IANA Port Assignment - http://www.iana.org/assignments/port-numbers

traditional: configure; make; make install.

The program is very simple to use. You can add up to 6 parameters to Xprobe.

- **-h** Help description
- **-p** The destination port of the first UDP probe (Default: 32132)
- **-i** The interface to use for performing the test
- **-v** To be verbose
- **-o** Log file of the trace
- Target host or network

Here is an example of the program's output on my Linux machine.

```
------------------------------------------------------------------------------------------
./xprobe -h
X probe ver. 0.0.2
------------------
usage: ./xprobe [-p portnum] [-i interface] [-v] host[/netmask] [-o logfile]
------------------------------------------------------------------------------------------
```

Let's look a little closer at what Xprobe can do against Windows XP and OpenBSD3.1 systems.

Xprobe against Windows XP

```
------------------------------------------------------------------------------------------
./xprobe -v 192.168.201.7
X probe ver. 0.0.2
------------------
Interface: eth0/192.168.201.2

LOG: Target: 192.168.201.7
LOG: Netmask: 255.255.255.255
LOG: probing: 192.168.201.7
LOG: [send]-> UDP to 192.168.201.7:32132
LOG: [98 bytes] sent, waiting for response.
TREE: IP total length field value is OK
TREE: Frag bits are OK
LOG: [send]-> ICMP echo request to 192.168.201.7
LOG: [68 bytes] sent, waiting for response.
TREE: Microsoft Windows Family TCP stack
TREE: Other Windows-based OS (ttl: 128)
FINAL:[ Windows 2k. SP1, SP2/Windows XP ]
------------------------------------------------------------------------------------------
```

As we can see, Xprobe fingerprints my Windows XP host correctly. It is unsure about the version of Windows, because Windows 2000 and Windows XP have approximately the same TCP stack. Now look how Xprobe reached this verdict:

First Packet (UPD)

```
08:48:24.703789 192.168.201.2.40879 > 192.168.201.7.32132:  [udp sum ok] udp
70 (DF) (ttl 250, id 35290, len 98)
4500 0062 89da 4000 fa11 e354 c0a8 c902
c0a8 c907 9faf 7d84 004e cec2 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000
```

We see the **UDP** packet is sent to port **32132** with the **Don't Fragment** bit set.

The following response comes from the XP system:

First response (ICMP port unreachable)

```
08:48:24.704062 192.168.201.7 > 192.168.201.2: icmp: 192.168.201.7 udp port
32132 unreachable for 192.168.201.2.40879 > 192.168.201.7.32132:  udp 70 (DF) (ttl
250, id 35290, len 98) (ttl 128, id 53621, len 56)
4500 0038 d175 0000 8001 55f4 c0a8 c907
c0a8 c902 0303 10b8 0000 0000 {4500 0062
89da 4000 fa11 e354 c0a8 c902 c0a8 c907
9faf 7d84 004e cec2}
```

{ } Represents the embedded UDP packet

Xprobe analyzes this answer and the following observation is done. The data embedded in the ICMP packet is equal to 8 bytes (0x38 = 56 bytes – 20 bytes of IP header (icmp) – 8 bytes of ICMP headers = 28 bytes – 20 bytes of the embedded IP Header = **8 bytes of the UDP header**). The **Don't Fragment** bit is echoed in the embedded packet and the **precedence bits are not set**. With this kind of result, Xprobe jumps into another branch, eliminating all OS that set the precedence bit and embed more or less 8 bytes in the ICMP packet. After that Xprobe sends an ICMP Echo Request to the target host to analyze its answer.

Second packet (ICMP ECHO REQUEST)

```
08:48:24.704204 192.168.201.2 > 192.168.201.7: icmp: echo request (DF) [tos
0x6,ECT]  (ttl 250, id 25418, len 68)
4506 0044 634a 4000 fa01 0a0d c0a8 c902
c0a8 c907 087b 459b 45cb 6c1e 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000
```

The second packet is a simple ICMP echo request (PING) with some flags in it.

First Xprobe sets the Type of service flags to 0x6 and the ICMP Code to 0x7b (123). The don't fragment bits is also set.

Second response

```
08:48:24.704363 192.168.201.7 > 192.168.201.2: icmp: echo reply (DF) (ttl 128, id
53622, len 68)
4500 0044 d176 4000 8001 15e7 c0a8 c907
c0a8 c902 0000 4e16 45cb 6c1e 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000
```

Look how Xprobe examines the last packet before giving its conclusion. First it checks the ICMP code of the ICMP Echo Reply. Since the code is zeroed we can confirm that this is a Windows host, because only Windows zeroes that field. So we have to figure out if it's a Windows 98, ME, NT, 2000 or XP system, then it looks at the TTL 80(128). Since the TTL is 128, we can discard Windows 95 because it answers back with a starting TTL of 32. The last check is the precedence bit. Since these bits are zeroed, we can conclude that the remote operating system is a Windows 2000 or XP, because Windows 98, ME and NT don't zero that field.

We saw Xprobe fingerprint a Windows XP box with a total of 4 packets on the network. But could Xprobe do better? Let's look at another trace.

Xprobe against OpenBSD 3.1

```
-----------------------------------------------------------------------------------------
./xprobe 192.168.201.9
X probe ver. 0.0.2
------------------
Interface: eth0/192.168.201.2

LOG: Target: 192.168.201.9
LOG: Netmask: 255.255.255.255
LOG: probing: 192.168.201.9
LOG: [send]-> UDP to 192.168.201.9:32132
LOG: [98 bytes] sent, waiting for response.
FINAL:[  OpenBSD 2.6-2.9 ]

-----------------------------------------------------------------------------------------
```

Note that it took only two packets to Xprobe to identify the remote OS. How can it do that?

First Packet (UPD)

```
10:49:40.081443 192.168.201.2.51601 > 192.168.201.9.32132:  [udp sum ok] udp 70
(DF) (ttl 250, id 11788, len 98)
4500 0062 2e0c 4000 fa11 3f21 c0a8 c902
c0a8 c909 c991 7d84 004e a4de 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000
```

The first packet is exactly the same as the one shown in the Windows XP trace, except that some fields such as the IPID, IP checksum, UDP checksum and the destination host have changed. Therefore I will not examine this packet any further.

First response (ICMP port unreachable)

```
10:49:40.081572 192.168.201.9 > 192.168.201.2: icmp: 192.168.201.9 udp port
32132 unreachable for 192.168.201.2.51601 > 192.168.201.9.32132:  udp 70 (DF) (ttl
250, id 11788, len 78, bad cksum 3f21!) (ttl 255, id 62084, len 56)
4500 0038 f284 0000 ff01 b5e2 c0a8 c909
c0a8 c902 0303 10ce 0000 0000 {4500 004e
2e0c 4000 fa11 3f21 c0a8 c902 c0a8 c909
c991 7d84 004e a4de}
```

The first thing that Xprobe will look at is the **precedence bit**. Since the **precedence bit is not set**, we conclude that the remote system is not a Linux system. The second step is to analyze how many data bytes are embedded in the embedded packet. Here we got a total length (0x**38**) of **56** bytes – 20 bytes (IP header) = 36 bytes – 8 bytes (ICMP header) = 28 bytes – 20 bytes (embedded IP header) = **8 bytes**. After that we look at the echoing total length field in the embedded packet. The first UDP packet had a total length of 0x**62** (**98**) and the one embedded in the ICMP port unreachable is 0x**4e** (**78**). The embedded total length is 20-byte short. It eliminates the AIX and BSDI branch, and then jumps into the tree for OpenBSD, Extreme Switch, NFR appliance and more.

For the next step, it compares the UDP checksum of the first UDP packet 0xa4de with the UDP checksum in the embedded packet. Since both match we eliminate the Extreme Switch and the NFR appliance. We must then compare the **IP checksum** (0x**3f21**) in the first UDP packet with the one embedded in the ICMP packet (0x3f21). With this last test we can conclude that the remote operating system is an OpenBSD one.

With only 2 packets, Xprobe was able to recognize an OpenBSD system and 4 packets to recognize a Windows XP system. Pretty impressive due to the fact that NMAP[4] was not able to identify both of them and that Queso[5] only recognized the OpenBSD system.

Next release

Because Xprobe is still maintained, new releases of the tool show up sometimes and a new one is near its production time. This new release has had several improvements over the first one we looked at in this paper.

- Signature database out of the code to let you add new signatures
- Pluggable modules
- API to design your own modules
- A new way of scoring system against each test. (Called the fuzzy approach)

All these improvements will make Xprobe one of the best operating system fingerprinting tool that we can find on the Internet these days.

How to detect it

Because the packets generated by Xprobe are like others we see on the network each day, it is more difficult to write a signature for that tool. But even with the goal of being stealthy, Xprobe gives us some clue about its presence. Look at the following Snort signatures:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 32132 (content: "|0000 0000|" ; dsize:
70;  classtype:attempted-recon; tag: host,4,seconds; msg: "Xprobe - OS Fingerprinting
! TAG RULE !";)
```

The rules above look for the first packet sent by Xprobe and fire an alarm, and also log the next four packets coming from that host (TAG keyword). So this way you could see all the traffic generated by the probes. This lets you verify if it really is an Xprobe try or something else. Since the Xprobe UDP packets are empty (loaded with zeros), we could also use any destination port, because the default port (32132) can be changed on the command line.

Conclusion

Because of the way Xprobe works, it is one of the most accurate and undetectable active fingerprinting tool you will find on the Internet. This tool is

[4] NMAP - http://www.insecure.org/nmap
[5] Queso - http://packetstormsecurity.org/UNIX/scanners/queso-980922.tar.gz

very effective on a local network without any packet filter device. One of the great assets of Xprobe is that the project is still under development, so new features will come and better techniques should be used. Check out the Xprobe site and give it a spin on our network to give you some idea about how it works.

References

1. Xprobe - http://www.xprobe.org/
2. ICMP Usage in Scanning
   Ofir Arkin- http://www.xprobe.org/archive/papers/ICMP_Scanning_v3.0.pdf
3. IANA Port Assignment - http://www.iana.org/assignments/port-numbers
4. NMAP - http://www.insecure.org/nmap
5. Queso - http://packetstormsecurity.org/UNIX/scanners/queso-980922.tar.gz
6. The Behavior and tools of Today's Hackers by
   Symantec – http://enterprisesecurity.symantec.com/article.cfm?articleid=1398
7. Cracker Tools and techniques by
   Edward Skoudis - http://www.infosecuritymag.com/2002/jul/faster.shtml
8. The tools and methodologies of the scrip kiddie by
   The honynet project – http://www.honeynet.org/papers/enemy/
9. Identifying remote hosts, without them knowing by
   Craig Smith, Peter Grundl – http://www.honeynet.org/papers/finger/
10. A new stealth port scanning method by
    Securiteam - http://www.securiteam.com/securitynews/3D5QFQANFK.html
11. The art of reconnaissance – simple techniques by
    Sai Bhamidipati – http://rr.sans.org/audit/recon.php
12. dumbscan.txt by
    antirez – http://www.kyuzz.org/antirez/papers/dumbscan.html

Practical Part 2 – Network Detects

First Detect – FTP Exploit CWD overflow and WU-FTP file completion overflow

Detection Data

Snort Alert:

```
[**] FTP EXPLOIT CWD overflow [**]
07/06-18:51:37.964488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x23E
134.126.133.162:2103 -> 46.5.180.133:21 TCP TTL:51 TOS:0x0 ID:10390 IpLen:20 DgmLen:560 DF
***AP*** Seq: 0xBB7E1B4C  Ack: 0x8599DDFF  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 30355649 6749585
43 57 44 20 30 30 30 30 30 30 30 30 30 30 30 30  CWD 000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 F0 FC 40 31 07 08 98 5F 08 08 EB 0C  0000..@1..._....
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C 90 90 90 90 90 90 90 90 90 90 90 90  ................
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1  1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59  j.X......1.....Y
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD  j.X........Yj.X.
80 EB 05 E8 ED 0A                                ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] FTP wu-ftp file completion attempt { [**]
07/06-18:51:38.004488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x52
134.126.133.162:2103 -> 46.5.180.133:21 TCP TTL:51 TOS:0x0 ID:10391 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0xBB7E1D48  Ack: 0x8599E008  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 30355653 6749588
43 57 44 20 7E 2F 7B 2E 2C 2E 2C 2E 2C 2E 7D 0A  CWD ~/{.,.,.,.}.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] FTP wu-ftp file completion attempt { [**]
07/06-18:51:38.164488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x49
```

```
134.126.133.162:2103 -> 46.5.180.133:21 TCP TTL:51 TOS:0x0 ID:10399 IpLen:20 DgmLen:59 DF
***AP*** Seq: 0xBB7E1DB0  Ack: 0x8599E13F  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 30355670 6749605
43 57 44 20 7E 7B 0A                              CWD ~{.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

1. Source of Trace

This trace came from the incidents Web site: http://www.incidents.org/logs/RAW/2002.6.6. This file is a Snort alert log file in tcpdump format. The network layout is unknown, however it looks like all alerts come from a sensor that monitors a class B network (46.5.0.0/21). This network belongs to IANA (Internet Assigned Numbers Authority).

```
Look at the ARIN report:
OrgName:    Internet Assigned Numbers Authority
OrgID:      IANA

NetRange:   46.0.0.0 - 46.255.255.255
CIDR:       46.0.0.0/8
NetName:    RESERVED-46
NetHandle:  NET-46-0-0-0-0
Parent:
NetType:    IANA Reserved
Comment:
RegDate:
Updated:    2002-08-23

OrgTechHandle: IANA-ARIN
OrgTechName:   Internet Corporation for Assigned Names and Numbers
OrgTechPhone:  +1-310-823-9358
OrgTechEmail:  res-ip@iana.org
```

2. Detect Generated By

This detect was generated with Snort 1.8.7 on my Mandrake Linux system. The log file was stored in tcpdump format. I replayed the log with Snort and the –r option that allowed me to analyze the tcpdump file. The ruleset that I used was the stable ruleset that can be downloaded from the Snort Web site: http://www.snort.org/dl/signatures/snortrules-stable.tar.gz. Please consult the Snort manual If you are not familiar with the way Snort logs alerts. The Snort rules that triggered the alerts were:

- `alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP CMD overflow"; flags:A+; dsize:>100; content:"CMD "; nocase; classtype:attempted-admin; sid:1621; rev:5;)`

- `alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP wu-ftp file completion attempt {"; flags:A+; content:"~"; content:"{"; reference:cve,CAN-2001-0886; reference:bugtraq,3581; classtype:misc-attack; sid:1378; rev:7;)`

3. Probability that the source address was spoofed

The IP address was definitely not spoofed. Each packet that we saw here was part of an FTP connection and required the establishment of a TCP connection to have the desired effect on the FTP server. This is impossible to perform with a spoofed IP, however, the remote computer can be compromised and the hacker can be using it to perform hacks on the Internet. We also have to consider that those packets are crafted.

4. Description of the attack

It appears that the attacker was scanning hosts on the network to find FTP servers that are vulnerable to the CWD attack and WU-FTPD file completion attack. The attacker always tries 3 types of attacks against each FTP server. The first type of attack only targets the Vermillion FTP server. When performed against the VFTP daemon, the attack allows the user or hacker to execute commands on the remote host, making the FTP server unavailable. In order to launch this attack, the attacker must send 3 CWD commands with more than 504 characters in the command. In this case, the attacker sent 3 CWD commands, but the last 2 did not contain 504 characters, so this first attempt was not the VFTPD exploit.

The 2 other CWD commands demonstrate another type of attack: the “wu-ftp file completion” buffer overflow attack. This type of attack is quite common against the WU-FTPD server 2.60 and 2.61. It utilizes an overflow in the glob function of GLIBC that was not addressed by the FTP daemon. To perform this attack, the logged user has to send a command with the “{“ character. As a result, once the attack is performed, the hacker can execute commands on the remote host like the user running the FTP server, or perform a DOS against this server. In this case, we do not see any evidence of exploit in the command, leading us to believe that it might only be a DOS.

Reference:

CWD Exploit:	-http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-1058
	-http://cgi.nessus.org/plugins/dump.php3?id=10293
File completion:	-http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0886
	-http://www.securityfocus.com/bid/3581

5. Attack Mechanism

To perform his probe, the attacker was able to log into the FTP server anonymously with a normal user name and then launch is 3 commands consecutively. There was no evidence in the log that the user logged into an FTP server to the following address: 46.5.180.133, 46.5.180.134, 46.5.180.135, 46.5.180.151, 46.5.180.153. However, it does not make sense to send these packets to a host that does not run an FTP server. It proves difficult to identify if the attacker used a script to perform his attack, nonetheless, the small interval between each host suggests that it was, in fact, scripted. We must consider that the user has to login, enter his exploit, then validate that it is working before starting on another target.

6. Correlations

The WU-FTP is a well-known FTP server that has proven to have several security flaws during the past few years. This makes it an attractive target for hackers. Also, the IP address of the attacker looks like a familiar IP for FTP used for hacks or scans. After conducting a search on Google and Security Focus, I found a message posted on the "Intrusion Mailing List" by Ken Connelly that shows a large portscan attempt against his network. In his message he mentions the IP 134.126.133.162 as the source of multiple FTP portscans. <http://cert.uni-stuttgart.de/archive/intrusions/2002/07/msg00041.html>. A search on www.dshield.org database revealed 90080 records, 90078 represent different targets. This indicates a mass scanning IP that tries to find vulnerable FTP servers to take control of.

7. Evidence of active targeting

When looking only at the alert log, we are inclined to conclude that these attacks were directly targeted at those 5 IPs mentioned above. However, it is quite difficult to see if the attacker scanned the network beforehand, because I only examined the alert file and not the entire traffic log. A quick review of the past 5 days of alert and portscan files did not provide any trace of scanning or other attacks from that source IP. Perhaps the attacker had already scanned the network before or knew the network services running on each host.

8. Severity

**Since I did not have any information about the network, where the trace came from, and the criticality of the target, I assumed the worst case scenario.*

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

- Criticality= 5 (We assume that those system are critical)
- Lethality=3 (Look like an HTTP and FTP server)
- System Countermeasures = 1 (Assume none)
- Network Countermeasures = 1 (Assume none)

6= (5+3)-(1+1)

Severity: 6

9. Defensive Recommendation

To protect a system against this type of attack, or any other against the WU-FTPD server, I recommend that you use another FTP implementation such as PROFTPD <http://www.proftpd.net> or always to perform updates on your WU-FTP server at <http://www.wu-ftpd.org>. Also, you should restrict the access to your FTP server to only those who truly need it. For the reason that standard FTPs are not secure, I also suggest that you use FTP over SSH or SFTP when performing a file transfer.

10. Multiple Choice Questions

What is the data port for a standard FTP file transfer?

a.21

b.20
c.2121
d.8080

Answer is: b

11. Questions from the Intrusion Mailing List

All questions are from Donald Smith and were received on September 11th, 2002. See appendix A for the original message.

1. When I mentioned that the source IP was not spoofed, Donald asked about the possibility of the man in the middle attack.

I did not previously mention the man in the middle attack because it is so difficult to reproduce. It must be performed against an operating system that has a predictable ISN number. Thus, if the attacker is performing a man in the middle attack, the connection between the two original hosts will be desynchronized and closed immediately. As such, the attacker will not be able to perform this 3 times in a row (for the vermillion FTPD exploit) or will not be able to execute a command after the buffer overflow (WU-FTPD exploit). For these reasons I did not mention it in my original post on the mailing list. To view a list of exactly which operating systems are vulnerable to ISN attacks, please see the following paper.

2. He wanted to know what would happen if only one CWD packet is sent; And if sending one CWD packet will reveal vulnerabilities.

I did not find any place where they discuss only sending one or two packets to the FTP server. I supposed that sending only one CWD command would be inconsequential, because to execute the entire exploit you have to send this command 3 times to the server.

3. Why does the character "~" work and not any other globbing characters?

In effect we find that in addition to the "~" character, the "/" can also be targeted. Accordingly, the globbing function in the WU-FTPD server is the same as the one found in the shell. Globbing serves to expand a named character such as "*","~" as well as many other characters. In reality, the WU-FTPD is performing a part of the globbing in its code. The exploit is due to an incorrect method of catching the error from the globbing function for the following characters: "~" and "/". For that reason, I believe that the "/" character can also be vulnerable to this kind of attack. See Neohapsis **post** for a more in-depth explanation.

Second Detect – MISC UPNP malformed advertisement

Detection Data:

```
Generated by ACID v0.9.6b21 on Sun September 15, 2002 16:43:19

------------------------------------------------------------------------------
#(5 - 62741) [2002-09-04 08:53:47] [CVE/CAN-2001-0876] [CVE/CAN-2001-0877]  MISC UPNP
malformed advertisement - cve CAN-2001-0876 - cve CAN-2001-0877
IPv4: MY.FW.IP.XX -> 239.255.255.250
      hlen=5 TOS=0 dlen=384 ID=65405 flags=0 offset=0 TTL=3 chksum=54149
UDP:  port=9744 -> dport: 1900 len=364
Payload:  length = 356

000 : 4E 4F 54 49 46 59 20 2A 20 48 54 54 50 2F 31 2E   NOTIFY * HTTP/1.
010 : 31 0D 0A 48 6F 73 74 3A 32 33 39 2E 32 35 35 2E   1..Host:239.255.
020 : 32 35 35 2E 32 35 30 3A 31 39 30 30 0D 0A 4E 54   255.250:1900..NT
030 : 3A 75 75 69 64 3A 64 34 38 35 38 65 32 31 2D 32   :uuid:d4858e21-2
040 : 38 37 34 2D 34 34 34 30 2D 62 30 36 64 2D 30 62   874-4440-b06d-0b
050 : 61 35 31 31 66 32 32 66 30 33 0D 0A 4E 54 53 3A   a511f22f03..NTS:
060 : 73 73 64 70 3A 61 6C 69 76 65 0D 0A 4C 6F 63 61   ssdp:alive..Loca
070 : 74 69 6F 6E 3A 68 74 74 70 3A 2F 2F XX XX XX 2E   tion:http://XXX.
080 : XX XX XX 2E XX 2E XX XX XX 3A 32 38 36 39 2F 75   XXX.X.XXX:2869/u
090 : 70 6E 70 68 6F 73 74 2F 75 64 68 69 73 61 70 69   pnphost/udhisapi
0a0 : 2E 64 6C 6C 3F 63 6F 6E 74 65 6E 74 3D 75 75 69   .dll?content=uui
0b0 : 64 3A 64 34 38 35 38 65 32 31 2D 32 38 37 34 2D   d:d4858e21-2874-
0c0 : 34 34 34 30 2D 62 30 36 64 2D 30 62 61 35 31 31   4440-b06d-0ba511
0d0 : 66 32 32 66 30 33 0D 0A 55 53 4E 3A 75 75 69 64   f22f03..USN:uuid
0e0 : 3A 64 34 38 35 38 65 32 31 2D 32 38 37 34 2D 34   :d4858e21-2874-4
0f0 : 34 34 30 2D 62 30 36 64 2D 30 62 61 35 31 31 66   440-b06d-0ba511f
100 : 32 32 66 30 33 0D 0A 43 61 63 68 65 2D 43 6F 6E   22f03..Cache-Con
110 : 74 72 6F 6C 3A 6D 61 78 2D 61 67 65 3D 31 38 30   trol:max-age=180
120 : 30 0D 0A 53 65 72 76 65 72 3A 4D 69 63 72 6F 73   0..Server:Micros
130 : 6F 66 74 2D 57 69 6E 64 6F 77 73 2D 4E 54 2F 35   oft-Windows-NT/5
140 : 2E 31 20 55 50 6E 50 2F 31 2E 30 20 55 50 6E 50   .1 UPnP/1.0 UPnP
150 : 2D 44 65 76 69 63 65 2D 48 6F 73 74 2F 31 2E 30   -Device-Host/1.0
160 : 0D 0A 0D 0A                                        ....
------------------------------------------------------------------------------
```

1. Source of Trace

This detect was retrieved from an ACID 0.9.6b21 console located on the internal network of my company. Currently only one sensor is logging into this console. This sensor is placed between one of our corporate routers and a firewall. It caught all traffic through a spanning port within the switch.

2. Detect was generated by

This alert comes from Snort 1.8.7 (build 128) sensor with a stealth interface that logs into an ACID console located on the internal network. The ruleset that I used for this sensor is based on the stable ruleset found at www.snort.org, with a few of my own minor modifications.

In the ACID report we can see that only the payloads of the packets are left in hexadecimal. All headers have been normalized into more readable form. Let's look at the ACID header:

```
#(5 - 62741) [2002-09-04 08:53:47] [CVE/CAN-2001-0876] [CVE/CAN-2001-
0877]  MISC UPNP malformed advertisement - cve CAN-2001-0876 - cve
CAN-2001-0877
IPv4: MY.FW.IP.XX -> 239.255.255.250
      hlen=5 TOS=0 dlen=384 ID=65405 flags=0 offset=0 TTL=3
chksum=54149
UDP:  port=9744 -> dport: 1900 len=364
Payload:  length = 356
```

Fields	Definition
#(5-62741)	Sensor and alert ID in ACID
[2002-09-04 08:53:47]	Date when this alert was captured
[CVE/CAN-2001-0876] [CVE/CAN-2001-0877]	Reference for that alert
MISC UPNP malformed advertisement	Snort signature that fire this alert
IPv4:	IP header under
MY.FW.IP.XX -> 239.255.255.250	Host involved in the alert
hlen=5	Header length (needs to be multiplied by 4 to get value in bytes)
TOS=0	Type of service bytes
dlen=384	Total length of the IP packet (with the UDP header and payload)
ID=65405	IP ID
flags=0	Fragment Flags (010=Don't fragment, 001 = More Fragment, 000 = normal packet)
Offset=0	Used when fragment packet are seen
chksum=54149	Checksum of the entire IP packet
UDP:	UDP header section
port=9744	Source port
dport: 1900	Destination port
len=364	UDP packet length (Header + payload)
Payload:	Payload section
length = 356	Length of the payload

3. Probability the source address was spoofed

In this case the IP is not spoofed, because this packet was generated by an internal computer that is NAT when passing thought the firewall. Another indication that the packet is not spoofed is that in the payload we can easily see the internal IP of the computer that generated this malformed UPNP packet. It is possible, however, for someone to spoof this kind of packet and generate a crafted IP in the payload. Although in this case I was able to verify that this packet was actually coming from an internal host, because it was from my office network.

4. Description of attack

This packet does not represent an attack in itself, but rather some misconfigured services on a computer that can be exploited at a later time. When reviewing my ACID log, I found this trace that was generated by my laptop. The two services that can be exploited are UPNP

(Universal Plug and Play Device Host) and SSDP (Simple Service Discovery Protocol). By default, Windows XP and any Win98, Win98 SE, Windows ME, and Windows Millennium SE are vulnerable to three types of attacks against those services. These attacks are: system remote exploit via a buffer overflow, DOS, and DDOS. You can find more information about these three types of attacks on the eyes Digital Security website. (http://www.eeye.com/html/Research/Advisories/AD20011220.html)

Here are the advisories about these services:

Microsoft advisory:

- http://www.microsoft.com/technet/security/bulletin/MS01-059.asp

CVE advisory:

- CVE-2001-0876 http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0876
- CVE-2001-0877 http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0877

5. Attack mechanism

The way UPNP works is fairly simple. When a device or a computer becomes live on a network, it sends a NOTIFY packet to a multicast address (239.255.255.250) on port 1900 to announce its presence and its description to other UPNP devices on the network. As a result, other devices know the presence of a new UPNP compatible device and can use it for the service that they provide. However Microsoft's implementation of UPNP and SSDP services encountered some difficulties.

DOS and DDOS

It can be problematic when a malicious user sends a crafted "NOTIFY" directly to a device or computer and states that his description should be downloaded from a specific location. For example, if the location of this description is a CHARGEN server, an endless loop could result, thus consuming all resources on the remote computer or device that tries to retrieve the description of the new UPNP-able system.

This kind of attack can be performed against a specific host or an entire network, depending on the destination of the "NOTIFY" message. If you involve a third party computer in the location of your description, and if you can reach a lot of computers with your crafted packet, you could perform a DDOS on that third party computer or device.

Buffer Overflow

SSDP is the protocol that causes a buffer overflow against Windows XP, 98, 98 SE, Millennium and Millennium SE systems. The SSDP runs on port 5000. The buffer overflow can be performed if a malicious NOTIFY packet is sent to the remote host. This permits the sender to execute commands on the remote device or computer because the SSDP runs with administrative privileges on Windows XP and is part of the operating system on other Windows OS. In this case, the hacker can execute almost any command that he wishes to. The buffer overflow should be placed in the LOCATION of the SSDP message.

6. Correlations

I found references to these particular attacks in the Practical (GCIH) of Chip Calhoun (http://www.giac.org/practical/Chip_Calhoun_GCIH.doc). I also found a script to test your XP computer against this DOS and DDOS vulnerability (http://qb0x.net/exploits/upnp_udp.c) as well as another to test against the SSDP buffer overflow at: (http://packetstormsecurity.org/0112-exploits/XPloit.c). In addition to the above mentioned services, you can also find useful information about this protocol at www.upnp.org.

7. Evidence of active targeting

The absence of any real attacks in this packet, indicates that there were no cases of active targeting. However, due to the NOTIFY packet that my laptop was sending, almost anyone in the office could exploit my computer because I had not installed the latest patch.

8. Severity

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

- Criticality = 5 (This was my laptop that had access to production servers)
- Lethality = 4 (Possibility of Buffer overflow and DOS)
- System Countermeasures = 1 (System not patched)
- Network Countermeasures = 1 (No firewall was running on my laptop)
  (External firewall was not blocking multicast request)

7= (5+4)-(1+1)

Severity: 7

9. Defensive recommendation

All systems should be updated with the following patch:

XP: http://www.microsoft.com/Downloads/Release.asp?ReleaseID=34951
98 or 98 SE: http://www.microsoft.com/Downloads/Release.asp?ReleaseID=34991
Millennium : http://download.microsoft.com/download/winme/Update/22940/WinMe/EN-US/314757USAM.EXE

The UPNP and SSDP services should be disabled if you do not use a UPNP device on your network. Verify with the "netstat –an" command the port 1900 and 5000 are closed. You should also block outbound packets with the destination of 239.255.255.250 on port 1900 on your firewall. The reason for this is that the UPNP service can leak information about your internal network in their NOTIFY messages. Needless to say, a personal firewall should be used on every workstation running any version of Windows.

10. Multiple choice test question

What port does the SSDP (Simple Service Discovery Protocol) service use?

a) 19
b) 1900
c) 5000
d) 8080

Answer: C

Third Detect – WEB-MISC /etc/passwd and http directory traversal

Detection Data

```
Generated by ACID v0.9.6b21 on Wed September 18, 2002 13:11:26

------------------------------------------------------------------------------
#(5 - 181001) [2002-09-07 00:18:07] [arachNIDS/297]  WEB-MISC http directory
traversal - arachnids 297
IPv4: 61.189.200.145 -> XX.XX.XXX.126
      hlen=5 TOS=0 dlen=117 ID=9586 flags=0 offset=0 TTL=107 chksum=61469
TCP:  port=1395 -> dport: 80  flags=***AP*** seq=3309017501
      ack=1708743254 off=5 res=0 win=8760 urp=0 chksum=31918
Payload:  length = 77

000 : 48 45 41 44 20 2F 2E 2E 2F 2E 2E 2F 73 68 61 64   HEAD /../../shad
010 : 6F 77 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73   ow HTTP/1.1..Hos
020 : 74 3A 20 XX XX 2E XX XX 2E XX XX XX 2E 31 32 36   t: XX.XX.XXX.126
030 : 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F   ..User-Agent: Mo
040 : 7A 69 6C 6C 61 2F 35 2E 30 0D 0A 0D 0A            zilla/5.0....
```

1. Source of Trace

This detect was retrieved from an ACID 0.9.6b21 console located on the internal network of my company. Currently only one sensor is logging into this console. This sensor is placed between one of our corporate routers and a firewall. It caught all traffic through a spanning port within the switch.

2. Detect was generated by

This alert comes from Snort 1.8.7 (build 128) sensor with a stealth interface that logs into an ACID console located on the internal network. The ruleset that I used for this sensor is based on the stable ruleset found at www.snort.org, with a few of my own minor modifications.

The Snort rule that set off this alarm is:

a. alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-MISC http directory traversal"; flags:A+; content: "..\\";reference:arachnids,298; classtype:attempted-recon; sid:1112; rev:4;)

In the ACID report we can see that only the payloads of the packets are left in hexadecimal. All headers have been normalized into more readable form. Let's look at the ACID header:

```
#(5 - 181001) [2002-09-07 00:18:07] [arachNIDS/297]  WEB-MISC http
directory traversal - arachnids 297
IPv4: 61.189.200.145 -> XX.XX.XXX.126
      hlen=5 TOS=0 dlen=117 ID=9586 flags=0 offset=0 TTL=107
chksum=61469
TCP:  port=1395 -> dport: 80  flags=***AP*** seq=3309017501
```

```
          ack=1708743254 off=5 res=0 win=8760 urp=0 chksum=31918
   Payload:  length = 77
```

Fields	Definition
#(5-181000)	Sensor and alert ID in ACID
[2002-09-07 00:18:07]	Date when this alert was received
[arachNIDS/297]	Signature reference
WEB-MISC http directory traversal - arachnids 297	Snort signature that sets off this alert
IPv4:	IP header under
61.189.200.145 -> XX.XX.XXX.126	Host involved in the alert
hlen=5	Header length (needs to be multiplied by 4 to get the value in bytes)
TOS=0	Type of service bytes
dlen=117	Total length of the IP packet (with the UDP header and payload)
ID=9586	IP ID
flags=0	Fragment Flags (010=Don't fragment, 001 = More Fragment, 000 = normal packet)
Offset=0	Used when fragment packets are seen
TTL=107	Time to live of this packet
chksum=61469	Checksum of the entire IP packet
TCP:	UDP header section
port=1395	Source port
dport: 80	Destination port
flags=***AP***	Flags set – Ack and Push
seq=3309017501	Sequence Number
ack=1708743254	Ack Number
off=5	Number of 32 bits word in TCP header 5x4 = 20 bytes header
Res=0	Reserved bit (ECN)
Win=8760	TCP windows size
Urp=0	Urgent Pointer data
chksum=31918	TCP checksum
Payload:	Payload section
length = 77	Length of the payload

3. Probability that the source address was spoofed

The source IP is probably not spoofed. The attacker tried to obtain specific information on the targeted host. If the source IP was spoofed then the attacker would not be able to read this information unless he can sniff traffic on the network where the spoofed IP is. However, spoofing an entire TCP session is not possible because it requires the three-way hand shake, and an HTTP request is based on the TCP. The system that belongs to this IP could be compromised and the attacker could launch his attack from this insecure computer. We could conceive of the possibility of session hijacking or a man in the middle attack, but due to the level of difficulty to achieve, I decided eliminate this scenario.

4. Description of attack

In this type of attack, the hacker tries to read private files from the file system. Judging from the trace that I found here, the user did not try to execute an arbitrary command on the server. However, other attack methods, such as CodeRed or Nimda, use directory traversal to execute commands on the remote host and thus gain control. In our case the attacker tried to see the shadow files that contain the encrypted passwords for each user. With this information, the attacker can attempt locally to brute force each password. If the hacker succeeds in finding one password, then he can try to log into the server through ssh, telnet, rsh, etc. If the attacker gains access to the server, then he can use the server to exploit other systems.

Reference: http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids297&view=event

5. Attack mechanism

Once the three-way handshake is made the attacker places a normal HTTP query that searches for the specific file that resides outside of the Document Root of the web server, by using the ".." notation. This is achieved by attempting to go up by two directories and then entering the "etc" directory to request the file shadow. Thus, if the web server is vulnerable to this attack, the hacker can see all the encrypted passwords of all the users on this system. This only works if the document root is two directories under the root directory (example: /www/htdocs).

6. Correlations

To verify that this attack was a false positive, I verified the log file of this web server running Apache version 1.3.23 on Linux. After checking the error_log file, I found that this IP was making numerous directory traversal attempts. I verified these log files to make sure that the request for the shadow file had failed.

```
[client 61.189.200.145] Invalid URI in request HEAD /../../shadow HTTP/1.1
```

Look at the other directory traversal that I found in my log file

```
[client 61.189.200.145] Invalid URI in request HEAD /../../etc/passwd HTTP/1.1
[client 61.189.200.145] Invalid URI in request HEAD /../../etc/passwd HTTP/1.1
[client 61.189.200.145] Invalid URI in request HEAD /../../../../etc/passwd
HTTP/1.1
[client 61.189.200.145] Invalid URI in request HEAD /../../../etc/passwd HTTP/1.1
```

I was troubled by the fact that my IDS did not catch these attacks. After looking at my Snort logs I found that I had unplugged my sensor from the network in order to perform an upgrade. My sensor had remained unplugged while these queries were hitting my web server, thus I did not detect them.

7. Evidence of active targeting

After finding other traces of attacks in the Apache log file coming from that address, I figured that it was probably just scanning software that was checking this server. Hence, it was not active targeting, but merely a reconnaissance run to identify vulnerable web servers.

8. Severity

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

- Criticality = 5 (Staging web server in DMZ where other important server reside)
- Lethality = 1 (Old directory traversal to see the shadows and password file)
- System Countermeasures = 4 (Invulnerable Apache version but not latest)
- Network Countermeasures = 5 (Firewall only let port 80 goes in on that server)

-3= (5+1)-(4+5)

Severity: -3

9. Defensive recommendation

In the current case, there is no possible defense against this type of attack. Since this attack will never be successful against the Apache server 1.3.23 and higher, the only recommendation that I can make is to upgrade to the latest version of Apache due to the known vulnerability in version under 1.3.26.

- http://www.cert.org/advisories/CA-2002-17.html
- http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0392

10. Multiple choice test question

When did the famous CodeRed worm hit the Internet?

a. June 2000
b. February 1999
c. September 2001
d. July 2001

Answer: D

Practical Part 3 – Analyze This

Overview

A University contacted me to investigate on their Snort logs. They wanted me to analyze five days of their logs and produce a report on abnormal events that I could have found in it. The rules were strict: I would not get help from them and I would not be able to ask questions about the network design and the Snort ruleset used in their sensors. I would have to download files from their site, analyze them and send the University a copy of my report. To respect the privacy of the University, we sanitized the first two digits of all internal IPs with 172.16. In my report I had to mention the files I selected, the top talker during the period of analysis, analyze the top alerts, scans, out of specifications lists, compromised servers and correlations between them.

Selected File

After searching on the University site for files, I decided to go with the most recent data that I could work with. The five consecutive files that I worked with are:

Alerts	Scans	OOS
alert.020801.gz	scans.020801.gz	oos_Aug.1.2002
alert.020802.gz	scans.020802.gz	oos_Aug.2.2002
alert.020803.gz	scans.020803.gz	oos_Aug.3.2002
alert.020804.gz	scans.020804.gz	oos_Aug.4.2002
alert.020805.gz	scans.020805.gz	oos_Aug.5.2002

These five alert files weighted 291 Megs uncompressed and had 2,593,013 alert lines. The scan files weighted 257 Megs and had 4,110,199 scan lines. The Out of Specification (OOS) files weighted 456 k and counted 9,870 lines. After starting my analysis, I found out that the OOS files were not named properly. They always contained the previous day's data. So actually my OOS files start on July 31st, 2002 through August 4th, 2002.

Alert Format:

The Alert files contain alerts from a Snort sensor that logs alerts in fast mode. Each alert contains the following information:

- **Date and time where this alert was caught**
- **Signature of the alert**
- Information about hosts involved in the alert
    - **Source IP**
    - **Source port**
    - **Destination IP**
    - **Destination Port**

Example:

```
08/01-23:58:19.650455 [**] UDP SRC and DST outside network [**] 3.0.0.99:137 ->
10.0.0.1:137
```

Scan Format:

These files had the same format of a standard portscan.log file from the Snort output plugging. For those unfamiliar with this file format, here is the information that can be found in those files and some examples:

- **Date and time when the scan was caught**
- Information about hosts involved in the alert
    - **Source IP**
    - **Source port**
    - **Destination IP**
    - **Destination Port**
- **Protocol or Scan Type**
- **Flags Sets (only for TCP scan)**

Example:

```
Aug  2 07:38:55 216.51.82.188:4085 -> 172.16.132.17:1433 SYN ******S*
Aug  2 00:07:03 172.16.165.24:6257 -> 24.82.175.49:6257 UDP
```

OOS Format:

The OOS files contain entries that don't respect the RFC, probably misconfigured flags, invalid checksum, bad size, etc. Each packet contains the following information:

- **Date and time when the packet was caught**
- Information about hosts involved in the OOS packet
    - **Source IP**
    - **Source Port**
    - **Destination IP**
    - **Destination Port**
- **Protocol (always TCP)**
- **Time To Live**
- **Type Of Service**
- **IP ID**
- **Fragment Flag**
- **TCP Flags**
- **Sequence Number**
- **Ack Number**
- **Window Size**
- **TCP Options**

Example:

```
07/31-19:44:50.801909 209.163.19.41:43028 -> MY.NET.88.162:51450
TCP TTL:106 TOS:0x0 ID:55554 DF
21S*R*** Seq: 0xD09E0BD9   Ack: 0xE04ABC0A   Win: 0XC50A
```

```
TCP Options => EOL EOL EOL EOL EOL EOL SackOK

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+
```

Top Talker List

#	Top 10 Sources IP	Alerts	# Destinations
1	***172.16.100.208***	1438614	107124
2	***172.16.84.234***	482388	480380
3	3.0.0.99	51379	1
4	63.250.213.12	32144	1
5	***172.16.81.37***	27085	2
6	194.98.189.139	8375	5456
7	***172.16.85.74***	6991	9
8	80.137.90.34	6899	45
9	***172.16.111.230***	6089	1
10	***172.16.111.231***	6059	1

#	Top 10 Destinations IP	Alerts	# Sources
1	10.0.0.1	51386	1
2	216.241.219.28	39484	5
3	233.28.65.148	32139	1
4	192.168.0.216	24208	5
5	233.2.171.1	17945	100
6	152.163.210.84	6458	4
7	233.28.65.173	4977	1
8	207.200.86.97	4759	8
9	***172.16.104.204***	4493	1529
10	209.10.239.135	3631	5

#	Top 10 Sources IP	Scans
1	***172.16.70.200***	2439486
2	***172.16.84.234***	478403
3	***172.16.100.208***	170345
4	***172.16.70.207***	137226
5	***172.16.82.2***	127792
6	***172.16.165.24***	104553
7	***172.16.83.150***	90049
8	***172.16.137.7***	49208
9	***172.16.70.133***	42744
10	***172.16.81.27***	31926

#	Top 10 destination IP	Scans
1	216.254.108.19	27885
2	204.183.84.240	12633
3	204.183.84.225	10562
4	67.68.113.139	9179
5	62.229.74.253	8704
6	216.254.108.22	8663
7	140.192.175.183	8301
8	66.130.178.166	7476
9	210.187.110.110	6810
10	12.245.28.142	6602

#	Top 5 OOS sources P	#
1	***68.32.126.64***	650
2	***62.76.241.129***	345
3	***209.16.70.75***	214
4	***212.35.180.17***	83
5	***65.210.154.210***	48

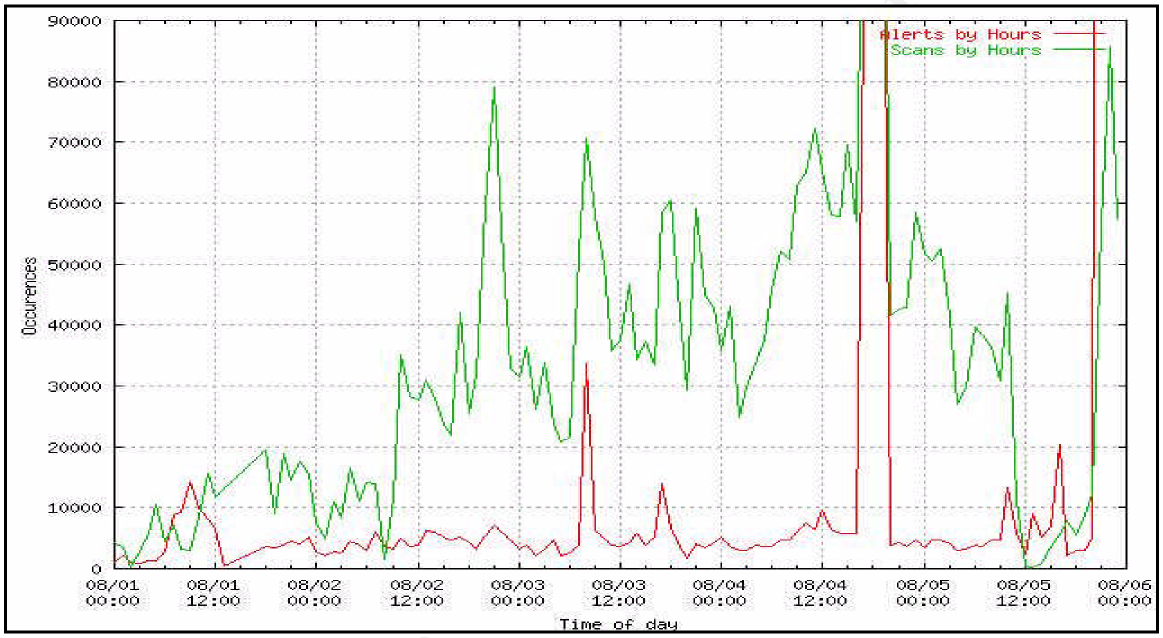
#	Top 5 OOS destination IP	#
1	**172.16.6.7**	658
2	**172.16.97.217**	241
3	**172.16.97.238**	104
4	**172.16.100.217**	95
5	**172.16.253.20**	85

First look at the alerts and scans distribution during these five days. There is the list of day analyzed and the day of the week they match for 2002.

Day	Day of the week	Alerts	Scans	OOS
August 1, 2002	Thursday	92155	176360	42

August 2, 2002	Friday	104600	622287	6779
August 3, 2002	Saturday	134971	986120	3039
August 4, 2002	Sunday	603511	1625720	5
August 5, 2002	Monday	1567776	699712	5

The volume of alerts during the periods seems stable except for the last two nights were we got a peak that doesn't look normal. But this peak is also significant in the scans during the same period. Maybe it is a repetitive event that triggers an alert or a server that was compromise during this period. We can conclude that this sensor generates lots of alerts. Lets look at the alerts and scans distribution over the past 5 days.



We can clearly see that the University received more scans than alerts during this period. We can also see two peaks for alerts and scans on the fourth and fifth days.

Signatures

Names

This is the list of alerts that we sorted out of the five days. The ruleset used to generate these alerts seems to be modified with some home rules. During these 5 days of data, there were 53 different types of alerts and a total of 2,229,959 alerts.

	Top 10 alert	#
1	NIMDA – Attempt to execute cmd from campus host	874,175
2	spp_http_decode: IIS Unicode attack detected	492,442
3	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	481,114
4	NIMDA – Attempt to execute root from campus host	122,833
5	UDP SRC and DST outside network	106,847
6	spp_http_decode: CGI null byte attack detected	53,560

7	SMB name wildcard	30,074
8	TFTP - External UDP connection to internal tftp server	24,212
9	External RPC call	14,576
10	Watchlist 000220 IL-ISDNNET-990517	11,916
11	Possible trojan server activity	4,113
12	SUNRPC highport access! =	2,543
13	IRC evil – running XDCC	2,053
14	Watchlist 000222 NET-NCFC	1,305
15	EXPLOIT x86 NOOP	1,293
16	Queso fingerprint	1,120
17	SNMP public access	927
18	Connect to 515 from outside	788
19	Attempted Sun RPC high port access	730
20	Samba client access	679
21	High port 65535 udp - possible Red Worm - traffic	628
22	IDS552/web-iis_IIS ISAPI overflow ida nosize	314
23	ICMP SRC and DST outside network	259
24	SMB C access	236
25	TFTP - Internal UDP connection to external tftp server	173
26	beetle.ucs	166
27	Port 55850 tcp - Possible myserver activity - ref. 010313-1	147
28	Incomplete packet fragments discarded	136
29	Null scan!	106
30	NMAP TCP ping!	88
31	EXPLOIT x86 setuid 0	58
32	Tiny Fragments - possible hostile activity	53
33	EXPLOIT x86 stealth noop	48
34	High port 65535 tcp - possible Red Worm - traffic	44
35	STATDX UDP attack	42
36	EXPLOIT x86 setgid 0	38
37	Port 55850 udp - Possible myserver activity - ref. 010313-1	18
38	SMB CD...	13
39	TCP SRC and DST outside network	13
40	***172.16.30.4*** activity	11
41	External FTP to HelpDesk ***172.16.70.50***	11
42	HelpDesk ***172.16.70.50*** to External FTP	11
43	HelpDesk ***172.16.70.49*** to External FTP	9
44	External FTP to HelpDesk ***172.16.70.49***	8
45	TFTP - External TCP connection to internal tftp server	6
46	EXPLOIT NTPDX buffer overflow	5
47	HelpDesk ***172.16.83.197*** to External FTP	4
48	Back Orifice	3
49	DDOS shaft client to handler	3
50	RFB - Possible WinVNC - 010708-1	3
51	SYN-FIN scan!	2
52	Traffic from port 53 to port 123	2
53	***172.16.30.3*** activity	1

We can easily conclude that most of the attacks launched from or against the University are UNICODE and IIS-related attacks. Unicode is a standard to represent almost every character into one table[6]. This type of attack stands for 88% of the entire attack list for these five days. Are included in the UNICODE and IIS attacks:

- "*NIMDA - Attempt to execute cmd from campus host*"
- "*NIMDA – Attempt to execute root from campus host*"
- "*spp_http_decode: IIS Unicode attack detected*"
- "*IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize*"

These types of alerts are the number one type of attacks against windows servers[7] these days over the Internet.

NIMDA - Attempt to execute cmd from campus host and NIMDA – Attempt to execute root from campus host

The NIMDA[8] worm is one of the most famous worms we've found on the Internet over the last year. This worm is distributed by email or while people are surfing the Internet, and via an IIS Unicode bug. Once the client or the IIS server is infected, the worm starts searching the Internet to infect other computers or IIS servers. The signatures we got here come from when the worm tries to propagate itself around the Internet via the IIS Unicode bug. Let's take a look at who triggers those alerts:

#	Top 5 sources IP	Alerts	# Destinations
1	***172.16.100.208***	996999	107128
2	***172.16.83.176***	1	1
3	***172.16.82.87***	1	1
4	***172.16.70.169***	1	1
5	***172.16.70.16***	1	1

As we can clearly see, all alerts are coming from ***172.16.100.208***. This host does not seem to target a host in particular, due to its high number of destination. This is clearly a host that is infected with NIMDA and is currently scanning the Internet and its neighbors to find other vulnerable systems to infect.

We can also identify when this system was compromised by displaying the number of alerts (not only NIMDA) a day generated by ***172.16.100.208***.

Days	1	2	3	4	5
Alerts	1	0	0	1	1438612

The NIMDA alert was targeted by HTTP GET queries against other hosts on port 80. If we can get the IIS http log, we will surely see requests such as this:

- *"GET /scripts/root.exe?/c+dir HTTP/1.0" 404 210 "-""-"*

[6] What is Unicode - http://www.unicode.org/unicode/standard/WhatIsUnicode.html
[7] Sans / FBI top 20 - http://www.sans.org/top20/
[8] CERT advisory NIMDA worm - http://www.cert.org/advisories/CA-2001-26.html

- *"GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232 "-" "-"*

These log traces are unsuccessful probes for vulnerable IIS servers due to the 404 HTTP status code[9] returned by the server. But with a vulnerable IIS server, this code will be an HTTP 200 ok code. With that code, you know that your server is vulnerable to this bug. You can also use some tool to scan you IIS Web server to figure this out.[10]

Alerts during the first days were only probes for RPC and SMB vulnerabilities that do not look positive. But on the last days, an alert for a connection on port 515 (LPD)[11] was surely not positive, due to the Windows worm we are analyzing. Shortly after that, we saw a high number of TFTP requests to an external TFTP server. A couple of minutes later, we started seeing a lot of alerts for NIMDA and other directory traversal or Unicode attacks going from and toward this host.

The TFTP request was done once the host was compromised. Moreover, the top source IP for the "*TFTP – Internal UDP connection to external tftp server*" signature is ***172.16.200.208***. This allows us to assume that this host was compromised and a root kit or something similar was uploaded to this computer. If the University had had more logs, we would have suggested checking what were those TFTP requests.

It is clear that this host has been compromised and needs to be taken out of the network for further investigation. If your site is running some IIS server, you should always get the latest patch you can find on the www.microsoft.com site for those systems.

spp_http_decode: IIS Unicode attack detected

A CodeRed, NIMDA or a directory traversal can represent IIS Unicode attacks. This type of attack is by far the most popular against IIS servers. It exploits a flaw when IIS translates Unicode characters. To get more information about some IIS Unicode attacks, look at the following paper that explains some of the biggest Unicode flaws in IIS servers. Normally, these alerts are generated by compromised servers or when users are searching for vulnerable IIS servers. In our case, we can see that compromised computers generated some of our alerts. Other IPs should be investigated further to see if they are coming from compromised computers.

#	Top 5 sources IP	Alerts	# Destinations
1	***172.16.100.208***	436048	250788
2	***172.16.85.74***	6982	10
3	80.137.90.34	6888	45
4	***172.16.152.19***	2885	40
5	***172.16.153.145***	2826	37

We can still see that most of the alerts are coming from the ***172.16.100.208*** computer that seems to be compromised. With these alerts, we can reinforce that this host is compromised and was actively scanning the Internet during these days. The second top talker for this type of alert can be very suspicious, because it is ranked 7th in the top 10 alert sources with 6,991 alerts. This lets only 9 other types of alerts against that host as a source. In these

[9] HTTP status code - http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html
[10] HFNetChk - http://ntbugtraq.ntadvice.com/default.asp?sid=1&pid=55&did=37
[11] LDP bug from sans top 20 list - http://www.sans.org/top20/ - U7

9 types of alerts, 8 are "*possible Trojan server activity*" and one "*SMB Name Wildcard*". But all these alerts come after the first Unicode alert triggered by this host. Maybe this host was already compromised before we started. We recommend the University to investigate all servers that generate lots of IIS Unicode alerts. We also recommend that all IPs that fire these alerts several times be analyzed too.

Correlation:

Keith Alexander and Carlin Carpenter also saw this type of attack**.**

IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize

ISAPI is a module loaded by default into IIS to interact with indexing service. This part of the IIS server had flaws when we requested a file with the ".ida" extension with some bad parameters. These parameters generated a buffer overflow on the system. Lots of worms and scanners use this technique to compromise or detect vulnerable IIS servers. The Code Red version one is one of them. So let's take a look at the top source IP to see if we got a worm on a server that tries to multiple IIS vulnerabilities against the Internet.

#	Top sources IP	Alerts	# Destinations
1	***172.16.84.234***	481114	480380

As we can see, only one IP was triggering this alert, and we saw lots of alerts from it. All these alerts occurred on the fourth. Maybe this computer was compromised on that day.

Let's take a look at other alerts against this host. We got 4 alerts for "*Possible trojan server activity*" against that host. Source IPs of those alerts were 80.62.155.240 and 217.136.63.141. Those IPs were not listed in the top 10 talkers list. We only saw a scan that triggered the Trojan signature against the University's subnet on those IPs on the second and fourth day. The alert on the fourth day was from 217.136.63.141. Oddly enough, we only saw ISAPI alerts on that day as well. The first Trojan alert was fired at 9:30 and the first ISAPI alert at 17:30. The gap between these first two alerts is quite big, but I suggest the University to investigate the scan of that host.

Host information about 217.136.63.141 is:

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html
inetnum:       217.136.48.0 - 217.136.63.255
netname:       BE-SKYNET-20010125
descr:         Belgacom Skynet SA/NV
descr:         ADSL BAS Antwerpen TL GO/PLUS
country:       BE
admin-c:       SN2068-RIPE
tech-c:        SN2068-RIPE
rev-srv:       ns.ripe.net
rev-srv:       ns1.skynet.be
```

```
rev-srv:       ns2.skynet.be
rev-srv:       ns3.skynet.be
rev-srv:       ns4.skynet.be
status:        ASSIGNED PA
mnt-by:        SKYNETBE-MNT
changed:       piet@skynet.be 20010203
source:        RIPE
route:         217.136.0.0/16
descr:         SKYNETBE-CUSTOMERS
origin:        AS5432
notify:        noc@skynet.be
mnt-by:        SKYNETBE-MNT
changed:       jfs@skynet.be 20010125
source:        RIPE
role:          Skynet NOC administrators
address:       Belgacom Skynet  SA/NV
address:       rue colonel Bourg 124
address:       B-1140 Brussels
address:       Belgium
phone:         +3227061311
fax-no:        +3227269311
email:         ripe@skynet.be
admin-c:       JFS1-RIPE
tech-c:        PDH16-RIPE
nic-hdl:       SN2068-RIPE
remarks:       -----------------------------------------
remarks:       Abuse notifications to: abuse@skynet.be
remarks:       Network problems to: noc@skynet.be
remarks:       Peering requests to: peering@skynet.be
remarks:       -----------------------------------------
notify:        noc@skynet.be
mnt-by:        SKYNETBE-MNT
changed:       ripe@skynet.be 20010907
source:        RIPE
```

We also got the information from dshield.org[12] about this host and only found one report. Therefore we cannot conclude that this host is the source of the problem.

UDP SRC and DST Outside Network

We see an important number of alerts with this signature in the alert files. The host involved in those alerts does not seem to be part of the internal University network range nor of the HOME_NET variable of the Snort sensor that captures this traffic. The source IP for all the alerts was 3.0.0.99 and the destination 10.0.0.1.

When we looked at the source, we saw that it was ranked third in the top 10 alert sources. It was probably spoofed, since all traffic was UDP. This type of traffic is easy to spoof because of how the protocol works. The 3.0.0.99 IP belongs to General Electric and is not listed on the www.dshield.com Web site. Most likely all these packets were spoofed or misrouted.

[12] Dshield – http://www.dshield.org

3.0.0.99 IP information:

```
OrgName:    General Electric Company
OrgID:      GENERA-9

NetRange:   3.0.0.0 - 3.255.255.255
CIDR:       3.0.0.0/8
NetName:    GE-INTERNET
NetHandle:  NET-3-0-0-0-1
Parent:
NetType:    Direct Assignment
NameServer: ns.ge.com
NameServer: ns1.ge.com
NameServer: ns2.ge.com
Comment:
RegDate:    1988-02-23
Updated:    2002-09-26

TechHandle: GET2-ORG-ARIN
TechName:   General Electric Company
TechPhone:  +1-518-612-6672
TechEmail:  genictech@ge.com
```

The destination IP is not a public IP that we can view on the Internet. The 10.0.0.0/255.0.0.0 range is reserved to private use only and is not routed on the Internet as described in the Address Allocation RFC 1918[13]. Maybe the University had a subnet with an IP range starting with 10.X.X.X and someone tried to exploit it with NETBIOS attacks. All alerts trigged by this IP had this signature, except for 2 others against ports 31 and 11.

To avoid this kind of alert or misuse, I recommend that the University check the configuration of this sensor to verify that this destination IP does not belong to them. It should also check if this source IP is not routed through their network by any possible way (VPN, misconfigured router, etc.) or if any administrator had any info about this source or destination IP or range.

Correlation:

John Jenkinson and Rick Yuen also had this source IP in their "*UDP SRC and DST outside network*" signature.

Scans

As we already saw with the alerts, a huge amount of scans were seen on the fourth day of analysis. During the whole period, we saw lots of scan types, but two types of scan were seen more often. Those were SYN and UDP scan.

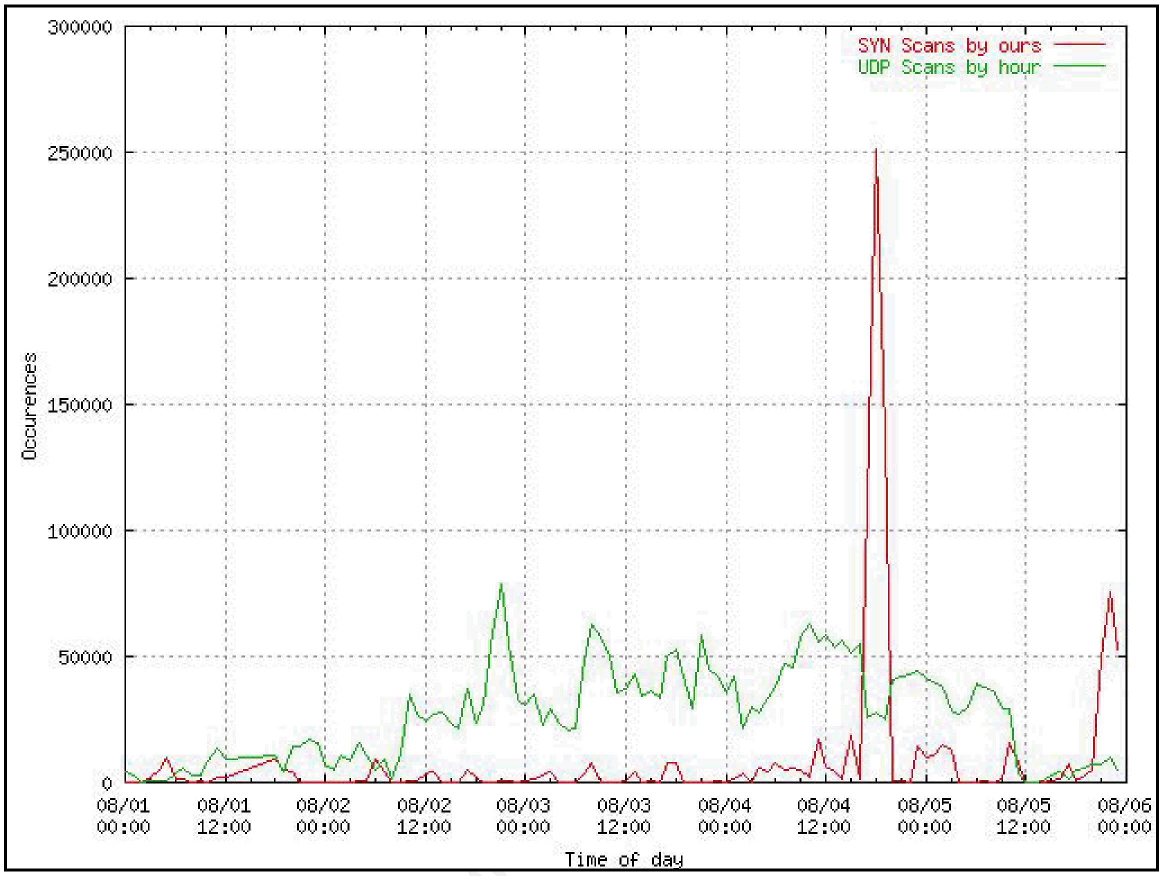
#	Scan Types	#	%

[13] RFC 1918 - http://www.isi.edu/in-notes/rfc1918.txt

1	UDP	3112432	75,7%
2	SYN	997436	24,9%
3	VECNA	71	Less than 1%
4	INVALIDACK	60	Less than 1%
5	NULL	59	Less than 1%
6	NOACK	52	Less than 1%
7	UNKNOWN	18	Less than 1%
8	SYNFIN	5	Less than 1%
9	FIN	4	Less than 1%
10	FULLXMAS	4	Less than 1%
11	XMAS	4	Less than 1%
12	NMAPID	1	Less than 1%

As shown in the table, other types of scans do not seem too dangerous or used. We will focus on the SYN and UDP scans, because they represent 99,6% of all scans.

Let's take a look at the scan distribution over the period of time before doing any analysis:



UDP Scan

#	Top 5 UDP Source	Scans
1	***172.16.70.200***	131364
2	***172.16.82.2***	92939
3	***172.16.70.207***	87534
4	***172.16.83.150***	36804
5	***172.16.165.24***	36009

130.85.70.200

Almost all scans from this IP was on the destination port 41170 and source port 4946. After searching the standard port list[14], we did not find anything useful about those ports. But a search with the google search engine helped us figure out that port 41170 UDP is used by Peer-to-Peer (P2P) software similar to Napster, Gnutella and KaZaA called Blubster[15]. We tried to figure out how Blubster works. Every client listens on port 41170 and all requests go on this port. We can find more information on the Blubster protocol, called "MANOLITO"[16], by reading this paper.

Because all data come from a University, we believe that a student or someone working for the University used this computer. The scan is part of the protocol, because when you launch the client, it tries to connect with clients from the same source port (in our case 4946). During these five days of analysis, we saw one other computer using Blubster with a high number of scans and with the ***172.16.70.180*** IP.

The University should be blocking this port and should investigate those two IPs to remove the Blubster software installed on the computers.

172.16.82.2

Below is the distribution for this IP's source port. We can clearly see that most of the activity was on ports 12300 and 12203.

Source Port	#
12300	83656
12203	12203
137	17

Those ports are unconventional and are not listed in the IANA port list[17]. But after some research, we discovered that those two ports were related to online gaming, particularly online server ports. We can therefore assume that someone installed a gaming server on that host. This computer needs to be removed and cleaned before other students us it.

[14] IANA port list - http://www.iana.org/assignments/port-numbers
[15] Blubster – http://www.blubster.com
[16] Blubster MANOLITO protocol - http://www.blubster.net/php/print.php?sid=34
[17] IANA port list - http://www.iana.org/assignments/port-numbers

172.16.70.207

As with the previous IP, we can clearly see that this computer is running an online gaming server. Just by looking at the source port of each scan entry we got in the scan files. University should investigate this computer before anyone else uses it.

source port	#
12300	79873
12203	7661

172.16.83.50 and 172.16.165.24

We investigated these two IP together because they got the same type of traffic during these five days. They seem to overuse port 6257. After performing a little search on www.dshield.org, we discovered that this port was used along with the WinMX[18] sharing program. This software is very similar to the other P2P software we analyzed before (Blubster, KaZaA, Morpheus and Gnutella). We suggest that the University investigate these two computers to see if they had some of these software installed. They can cause a lot of trouble because the files users download can be very suspicious and can contain viruses or other things you don't want on your network.

SYN Scan

#	Top 5 SYN source	Scans
1	***172.16.84.234***	478439
2	***172.16.100.208***	170067
3	24.138.61.171	11266
4	216.228.171.81	10627
5	80.137.90.34	9457

172.16.84.234

This IP was previously analyzed in the Alert section. We already know that it was ranked second in the top 10 sources of alerts and was the only one that fired the "*IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize*" signature. We already know that this alert was fired against Web servers. There are also a lot of worms that cause your infected computer to start scanning the Internet to find other victims. By watching the destination port of each scan, we can see if the scan and alert entries are related to the same packet.

Destination port	#
80	478420
1443	6
21	3
12345	3

With this last check we can clearly saw that this computer was actively scanning the

[18] WinMX – http://www.winmx.com

Internet for very well know port. Of course the port 80 stand for web server. Port 1443 stand for MS-SQL server, 21 for FTP and 12345 NetBus and Trend Micro Office scan. But we can clearly saw that this host was primarily scanning web server. So we reinforce the need to investigate this probably compromised computer.

172.16.100.208

As the previous one, this IP was also another member of the top ten sources of alerts. It was ranked number one in the top ten with the following signatures:

- "*NIMDA - Attempt to execute cmd from campus host*"
- "*NIMDA – Attempt to execute root from campus host*"
- "*spp_http_decode: IIS Unicode attack detected*"

Again we can correlate the previous analysis we did for this IP by watching the destination port this IP was scanning. We can verify that it was scanning to find other vulnerable Web servers across the Internet.

Destination port	#
80	169947
445	49
139	49
25	14
1433	4
21	3
111	1

So this little table shows that this computer was actively scanning the Internet to infect other computers. This is why we reinforce the fact that the University must remove this computer from the network and investigate it, or even reinstall it.

24.138.61.171

This is the first time we saw this IP in the top list. But which type of scan did this computer do against us? To find out, we sorted out all destination ports related with this IP in the scan files and found that this IP only scanned port 80. This represents 11,266 scans against our subnet. After sorting out all IP destinations, we saw that this IP was only scanning our network, because no IP for our subnet received more than 3 scans from this IP.

So this IP belong to:

```
OrgName:    Access Cable Television
OrgID:      ACCA

NetRange:   24.138.0.0 - 24.138.79.255
CIDR:       24.138.0.0/18, 24.138.64.0/20
NetName:    ACCESS-BLK1
NetHandle:  NET-24-138-0-0-1
Parent:     NET-24-0-0-0-0
NetType:    Direct Allocation
NameServer: EUROPA.ACCESSCABLE.NET
NameServer: PEGGY.ACCESSCABLE.NET
Comment:
RegDate:    1997-09-05
Updated:    2002-07-24

TechHandle: JP1495-ARIN
TechName:   Potvin, Jeff
TechPhone:  +1-902-469-9540
TechEmail:  jpotvin@accesscable.com

OrgName: Access Cable Television
OrgID:   ACCA
Address: 190 Victoria Rd Dartmouth Nova Scotia B2Y 4A4
Country: CA
Comment:
RegDate: 1997-09-05
Updated: 2002-07-24
```

This means that the IP came from a home Internet broadband distributor, probably from scripts kiddies trying his last scanning tool against your range. After searching the www.dshield.org Web site, we didn't find any report about this IP.

216.228.171.81 and 80.137.90.34

As the previous IP, these two ones belong to a cable and broadband Internet provider. So maybe it was some scripts kiddies again trying his latest port scanner against the University network. After looking at the destination port of all scanning activities for those IPs, we discovered that they were only scanning port 80. After a check at the www.dshield.org Web site, we did not find any report against these two IPs.

IP 216.228.171.81 information:

```
CustName:   bend cable communications
Address:    63090 sherman road bend, OR 97701
```

```
Country:    US
Comment:
RegDate:    2000-04-01
Updated:    2001-04-20

NetRange:   216.228.168.0 - 216.228.172.255
CIDR:       216.228.168.0/22, 216.228.172.0/24
NetName:    BCCI228-DOCSIS
NetHandle:  NET-216-228-168-0-1
Parent:     NET-216-228-160-0-1
NetType:    Reassigned
Comment:
RegDate:    2000-04-01
Updated:    2001-04-20
```

IP 80.137.90.34 information:

```
inetnum:      80.128.0.0 - 80.146.159.255
netname:      DTAG-DIAL16
descr:        Deutsche Telekom AG
country:      DE
admin-c:      DTIP-RIPE
tech-c:       ST5359-RIPE
status:       ASSIGNED PA
remarks:
*****************************************************************
remarks:      * ABUSE CONTACT: abuse@t-ipnet.de IN CASE OF HACK
ATTACKS,      *
remarks:      * ILLEGAL ACTIVITY, VIOLATION, SCANS, PROBES, SPAM,
ETC.         *
remarks:
*****************************************************************
notify:       auftrag@nic.telekom.de
notify:       dbd@nic.dtag.de
mnt-by:       DTAG-NIC
changed:      auftrag@nic.telekom.de 20020108
source:       RIPE

route:        80.128.0.0/11
descr:        Deutsche Telekom AG, Internet service provider
origin:       AS3320
mnt-by:       DTAG-RR
changed:      bp@nic.dtag.de 20010807
source:       RIPE

person:       DTAG Global IP-Adressing
address:      Deutsche Telekom AG
address:      Bayreuther Strasse 1
address:      D-90409 Nuernberg
address:      Germany
phone:        +49 911 68909856
e-mail:       ripe.dtip@telekom.de
nic-hdl:      DTIP-RIPE
mnt-by:       DTAG-NIC
changed:      ripe.dtip@telekom.de 20020717
source:       RIPE
```

```
person:        Security Team
address:       Deutsche Telekom AG
address:       Techniknieder1assung Schwaebisch Hall
address:       D-89070 Ulm
address:       Germany
phone:         +49 731 100 84055
fax-no:        +49 731 100 84150
e-mail:        abuse@t-ipnet.de
nic-hdl:       ST5359-RIPE
notify:        auftrag@nic.telekom.de
notify:        dbd@nic.dtag.de
mnt-by:        DTAG-NIC
changed:       auftrag@nic.telekom.de 20010321
source:        RIPE
```

Out of Specification

As mentioned before, the data found in those files do not meet the RFC standard. All packets that do not meet the standard should be viewed as malicious. But sometimes poor software makers do not read the RFC standard and normal traffic could be viewed as OOS traffic. Malicious packets can be used to:

- Bypass filtering devices (ACL)
- Perform denial of service against some operating system (WinNuke)
- Fingerprint (NMAP, Queso, Xprobe)

All OOS files only contain packets that were launched against the University subnet. Our analysis will be focused on the destination. One of the major things we discovered in the OOS file is that all packets had non-standard TCP flags with respect to RFC 793. But this RFC is very old and does not have all the new additions to the TCP implementation.

Let's take a look at some of the various flags combination we got:

Flags	#
21S*****	1604
2*SFR**U	2
2*SF****	2
21S*R**	2
21S***A*	2

As we can see, one pattern appears more often than others. If we simply follow the old RFC or the TCP/IP illustrated Volume1, these packets do not meet the standard because they set the two reserved flags (21), called the ECN flags. Originally, those flags could never be set in any packet. But since the RFC3168, they can now be set to tell other routers or hosts that congestion occurs on the line. Right now, not too many OS implement the ECN flags. This is why it is still not common to see those flags set.

When a device or a computer is ECN aware, it sets these two flags in their initial SYN. If the other side is ECN aware, it will reply with the ECE (1) bit set. Otherwise these two flags must be set to zero. Right now, only Linux supports ECN flag bits. Generally it is used to perform OS fingerprinting. Because we don't know if the University had hosts and routers supporting ECN bits, we will consider this traffic as non-standard and malicious.

68.32.126.64

The top source IP for the OOS files did not occur in any of the alert or scan files during these five days. All these 650 malicious packets were directed on port 110, which stands to be the POP3 mail service. These packets were probably shown in the OOS file because of a non-standard TCP flags sequence. The TCP flags that were set were the two ECN flags and SYN flag. This host is probably performing fingerprinting or is an ECN aware host that tries to pick up its mail from the University POP3 server.

IP information:

```
CustName:   Comcast Cable Communications, Inc.
Address:    3 Executive Campus Cherry Hill NJ 08002
Country:    US
RegDate:    2002-06-15
Updated:    2002-06-15

NetRange:   68.32.112.0 - 68.32.143.255
CIDR:       68.32.112.0/20, 68.32.128.0/20
NetName:    JUMPSTART-BALTIMOR-A3
NetHandle:  NET-68-32-112-0-1
Parent:     NET-68-32-0-0-1
NetType:    Reassigned
Comment:
RegDate:    2002-06-15
Updated:    2002-06-15
```

After reviewing the www.dshield.org Web site, we discovered that 2 reports of malicious attacks were logged, but no ports were linked to these two reports. It could be a student who is picking up his mail from his cable modem over the POP3 server.

62.76.241.129

Moreover, the second IP listed in the top 5 of the OOS did not show up in the alert and scan files. After analyzing the packets logged by this host, we did not find any suspicious IP ID, sequence number, type of service value, and time to live. As for previous IPs, only the TCP flags were suspicious. The flags sequence was the same as the previous IP. Both ECN and SYN flags were set. The two target hosts were MY.NET.97.217 and MY.NET.238. Only one destination port (113) was targeted. Port 113 is the IDENT port and is normally assigned with the Identification Server[19]. IDENT is used to identify incoming users for the following protocol: FTP, POP mail and HTTP. Maybe some users at the University were doing FTP, POP of HTTP transactions with servers that looked for IDENT and were ECN aware.

IP information:

```
inetnum:      62.76.240.0 - 62.76.243.255
netname:      UDMEDU-NET
descr:        Internet Center of Udmurt State University
```

[19] IDENT - ftp://ftp.isi.edu/in-notes/rfc1413.txt

```
descr:        ul. Universitetskaja, 1, k.6, Izhevsk, Russia
country:      RU
admin-c:      BGA4-RIPE
tech-c:       DMIR-RIPE
status:       ASSIGNED PA
notify:       dm@uni.udm.ru
mnt-by:       ROSNIIROS-MNT
changed:      ip-dbm@ripn.net 20000128
source:       RIPE

route:        62.76.240.0/22
descr:        UDMEDU-NET
origin:       AS13094
mnt-by:       UDSU-MNT
changed:      dm@uni.udm.ru 20010124
source:       RIPE

person:       Basil G. Ananin
address:      ul. Universitetskaja, 1, k.6, room 320
address:      Internet Center of UdSU
address:      Izhevsk, Russia
phone:        +7 3412788697
fax-no:       +7 3412788697
e-mail:       anan@uni.udm.ru
nic-hdl:      BGA4-RIPE
notify:       dm@uni.udm.ru
notify:       ip-reg@ripn.net
changed:      dm@uni.udm.ru 20000128
source:       RIPE

person:       Dmitry N. Mironov
address:      1, ul. Universitetskaja
address:      Izhevsk
address:      Russia
phone:        +7 3412 751758
fax-no:       +7 3412 788697
e-mail:       ddd@uni.udm.ru
nic-hdl:      DMIR-RIPE
notify:       ddd@uni.udm.ru
changed:      ddd@uni.udm.ru 19981020
source:       RIPE
```

After looking at the www.dshield.org Web site to find any evidence against that host, we only found 3 reports of attack with no port associated with it. But with some manual testing, we discovered that this IP is serving as FTP server and is sending back an IDENT query when you connect to it with ECN flags. We also noticed in the FTP banner that this host is a Linux server. As mentioned before, this OS supports ECN. Maybe the two destinations that belong to the University were only fetching FTP stuff to a Russian FTP server. After looking at the services offered by this server over the Internet, we found out that this server is very open. Maybe the University should check the two destination servers or workstations to figure out if they are not compromised or contains malicious software.

209.116.70.75

This source IP was already listed in the alert and scan files. We found 645 entries for the "*Queso Fingerprint*" signature. Queso[20] is like NMAP[21] and Xprobe[22]. It is designed to identify the remote operating system by sending malformed packets and then analyze how each responded to those malformed packets.

All Queso attempts performed from this host were toward port 25, which stands for the SMTP port. We identified 8 destinations that were scanned by this source host.

They are:

#	Destination	# packets
1	**MY.NET.100.217***	95
2	**MY.NET.6.47**	23
3	**MY.NET.6.40**	23
4	**MY.NET.6.34**	18
5	**MY.NET.253.11**	18
6	**MY.NET.6.35**	16
7	**MY.NET.253.43**	14
8	**MY.NET.139.230**	8

* ***Listed in the top 5 OOS destination***

IP information:

```
CustName:   Red Hat, Inc.
Address:    4518 South Miami Blvd. Suite #100 Durham NC 27703
Country:    US
RegDate:    2002-09-23
Updated:    2002-09-23

NetRange:   209.116.70.64 - 209.116.70.95
CIDR:       209.116.70.64/27
NetName:    INFLOW-18773-5591
NetHandle:  NET-209-116-70-64-1
Parent:     NET-209-116-68-0-1
NetType:    Reassigned
Comment:
RegDate:    2002-09-23
Updated:    2002-09-23
```

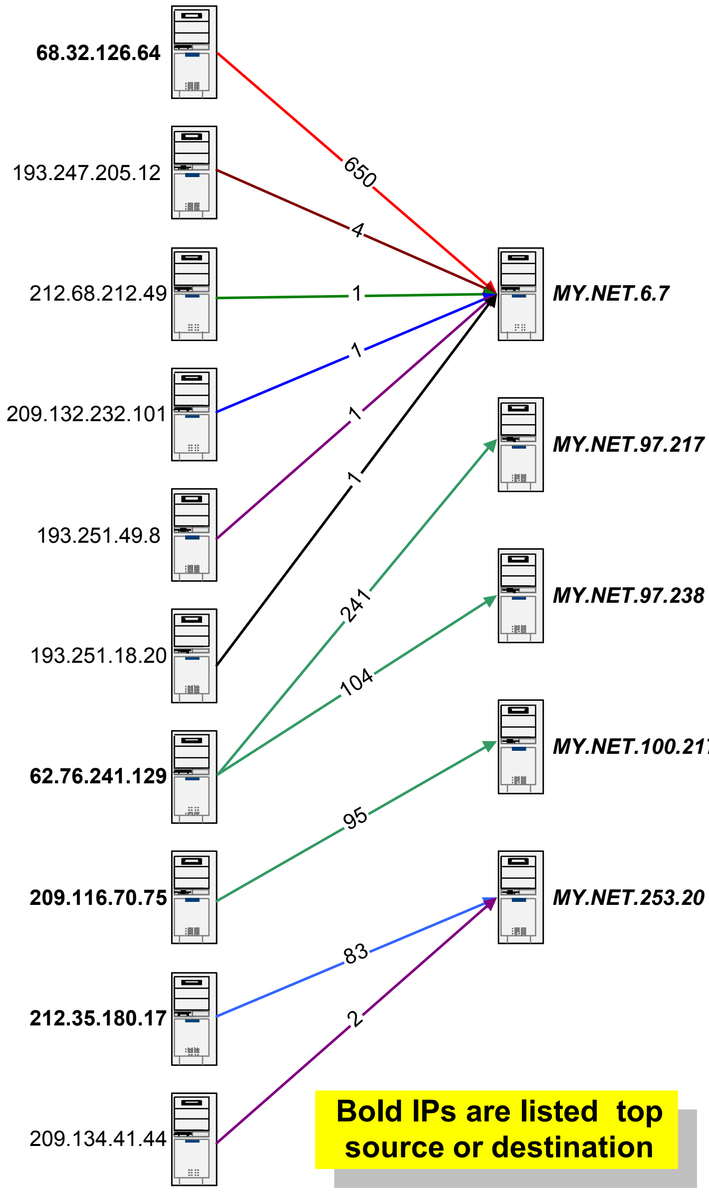
We also checked why all these packets were in the OOS section. It seemed to be like for the previous IP, because of the ECN flags in the SYN packet. Maybe someone from Red Hat was scanning some of the University IPs from an ECN aware server.

Linked Graph

In order to see if we rounded up almost all major top sources and destinations from the OOS file, we built up a linked graph showing all packets against the top 5 destinations to verify

[20] Queso - http://packetstormsecurity.org/UNIX/scanners/queso-980922.tar.gz
[21] NMAP - http://www.insecure.org/nmap/
[22] Xprobe - http://www.xprobe.org

how many top sources we would see.



We can clearly see that those OOS packets came from ECN aware hosts and probably should not be found in the OOS files. Normally we should not see packets with abnormal flags such as SYN+FIN, or any combination not found in RFC 793 and other subsequent TCP RFC. To prevent these types of packets to be in your network, be sure to set up a good firewall in front of all your internal or critical computers. Today, almost every firewall can discard those abnormal packets. Check your firewall documentation for further information.

Correlations

All my correlations are in my document.

Defense Recommendations

After reviewing all alert and scan files, it is important to mention that most of the alerts we got during these five days were related to IIS Web servers. It is very important for the University staff to check and patch all IIS Web servers it has on the campus. As mentioned before, you can also scan you server with tools you can find on the Internet. If you are not restrained to use Microsoft IIS Web server, we suggest to use the Apache[23] Web server. It is one of the most used and more secure Web server you can find on the Internet.

We noticed that the University gets weird traffic with some sources and destinations outside their IP range. So it will be important that the University staff investigate this traffic and drill down where it came from. It can come from a VPN or from a misconfigured router. This traffic seems to be inoffensive and can consume University bandwidth.

Since in a University students must use computers, a lot of peer-to-peer software seems to be used. We recommend blocking P2P software with their firewalls if feasible. For example, the Blubster P2P program uses a static UDP port to fetch and search files across the Internet. By blocking this port, you can disable this software by only adding one rule to you firewall. So investigation against those types of software should be done.

Analysis Process

To process all this data, I first tried to use SnortSnarf[24] from Silicon Defense[25]. Due to the size of all files, it became impossible for my computer to process all this data in a one shot pass. So I've searched through the GIAC practical to find some other tool or scripts. I did not find any scripts that fulfilled my needs. I have to admit that I did not search thoroughly. So I created my own Perl and Bash scripts. Those scripts were inspired from other scripts that I found in some others practicals. Because I used the same script for almost all my work, it was changing a lot and I was not able to get all versions of it. So I will not show you my script for parsing alert and scan.

Parse Alert and Scan Files

I used and modified the script of Gary Smith[26]. As mentioned above, I did not have the

[23] Apache - http://httpd.apache.org
[24] SnortSnarf - http://www.silicondefense.com/software/snortsnarf/
[25] Silicon Defense - http://www.silicondefense.com/

proper version of each script, so please refer the Mr. Smith's scripts.

Parse OOS Files

Retrieve all Flags Sequence

```
cat oos_Aug.*  | grep "Seq:" | cut -d" " -f1 | sort | uniq -c | sort -rn >
oos_flags.txt
```

Retrieve all Packets by Source IP

This script creates a comma-separated file for each source IP.

Here is how to use it:

cat oos_Aug.* | perl oos.parse.pl

The script:

```
#!/bin/perl

#first create a directory called like the dir variable

#07/31-03:01:26.781451 4.64.202.110:22690 -> MY.NET.88.162:1607
#TCP TTL:106 TOS:0x0 ID:55554  DF
#21S*R*** Seq: 0xD09E0BD9   Ack: 0xE04ABC0A   Win: 0xC50A
#TCP Options => EOL EOL EOL EOL EOL EOL SackOK

#    1    2    3    4    5    6   7   8 9   10   11  12  13  14
#   DATE,SRC,SPORT,DST,DPORT,TTL,TOS,ID,DF,FLAGS,SEQ,ACK,WIN,OPT\n";
$dir="oos";
`rm -f $dir/*`;
while ( <> )
{
    chomp;
    if (( $_ !~ /\+=/ ) and ( $_ !~ /^\s*$/ ) )
    {
       #Date line
       if ( $_ =~ /^\d+\/\d+/ )
       {

           #print "Date line is : $_\n";
           ($date,$src1,$dump,$dst1) = split /\s/ ;
           ($src,$sport) = split /:/, $src1;
           ($dst,$dport) = split /:/, $dst1;
           #open DATAF, ">$dir/$src.csv";
           #print DATAF "$header" if ! -f $dir/$src.csv;
           #close DATAF;

           #print "SRC $src SPORT $sport -> DST $dst DPORT $dport\n";

       }
       elsif ( $_ =~ /^TCP TTL/ )
       {
           #print "IP HEADER line is $_\n";
           ($proto,$ttl,$tos,$id,$ff) = split /\s+/;
           #print "PROTO: TCP\n";
           #print "  TTL $ttl TOS $tos ID $id Frag $ff\n";
```

[26] Gary Smith Practical - http://www.giac.org/practical/Gary_Smith_GCIA.zip

```
        }
        elsif ( $_ =~ /Seq:/ )
        {

            #print "flags line is $_\n";
            ($flags,$seq,$seq1,$ack,$ack1,$win,$win1) = split /\s+/;
            #print "FLAGS $flags SEQ $seq1 ACK $ack1 WIN $win1\n";
        }
        elsif ( $_ =~ /^TCP Options/ )
        {
            #print "TCP Options line is $_\n";
            ($dump,$opt) = split />/;
            #print "OPTIONS $opt\n";
            open DATAF, ">>$dir/$src.csv";
            print DATAF
"$date,$src,$sport,$dst,$dport,$ttl,$tos,$id,$ff,$flags,$seq1,$ack1,$win1,$opt\n";
            close DATAF;
        }
        else
        {
           print "Discarded line : $_\n";
        }
    }
}
```

References

1. Dshield – www.dshield.org
2. Unicode – http://www.unicode.org/unicode/standard/WhatIsUnicode.html
3. Sans Top 20 list – http://www.sans.org/top20/
4. Cert – http://www.cert.org/advisories/CA-2001-26.html
5. HTTP Status code – http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html
6. HFNetchk – http://ntbugtraq.ntadvice.com/default.asp?sid=1&pid=55&did=37
7. Address Allocation RFC – http://www.isi.edu/in-notes/rfc1918.txt
8. IANA port list – http://www.iana.org/assignments/port-numbers
9. Blubster
   - Main site : http://www.blubster.com
   - Manolito Protocol – http://www.blubster.net/php/print.php?sid=34
10. WinMX – http://www.winmx.com
11. IDENT RFC – ftp://ftp.isi.edu/in-notes/rfc1413.txt
12. Queso – http://packetstormsecurity.org/UNIX/scanners/queso-980922.tar.gz
13. NMAP – http://www.insecure.org/nmap
14. Xprobe – http://www.xprobe.org
15. Apache – http://httpd.apache.org
16. SnortSnarf – http://www.silicondefense.com/software/snortsnarf/
17. Snort – http://www.snort.org
18. ACID – http://www.andrew.cmu.edu/~rdanyliw/snort/snortacid.html
19. Microsoft – http://www.microsoft.com
20. ARIN WHOIS server - http://ws.arin.net/cgi-bin/whois.pl
21. RIPE WHOIS server - http://www.ripe.net/perl/whois
22. GNU plot - http://www.gnuplot.org/
23. GNU plot tutorial
   - http://www.duke.edu/~hpgavin/gnuplot.html
   - http://physics.ucsc.edu/~farris/helpwithgnuplot.html
24. GNU plot manual –

25. - http://www.comnets.rwth-aachen.de/doc/gnu/gnuplot37/gnuplot.html
26. Tom Rodriguez
    Understanding IIS Unicode Vulnerabilities - http://216.239.51.100/search?q=cache:K83fHGDuwW0C:www.infosecalliance.com/resources/whitepapers/iis-unicode-vuln.pdf+IIS+Unicode+attack+detected&hl=en&ie=UTF-8
27. Keith Alexander Practical - http://www.giac.org/practical/KeithAlexander_GCIA.zip
28. Carlin Carpenter Practical - http://www.giac.org/practical/Carlin_Carpenter_GCIA.doc
29. John Jenkinson Practical - http://www.giac.org/practical/John_Jenkinson_GCIA.doc
30. Rick Yuen Practical - http://www.giac.org/practical/Rick_Yuen_GCIA.doc
31. Gary Smith Practical - http://www.giac.org/practical/Gary_Smith_GCIA.zip
32. TCP/IP Illustrated Volume 1 by Richard Stevens
33. Visio - http://www.microsoft.com/office/visio/default.asp
34. Excel - http://www.microsoft.com/office/excel/default.asp
35. Word - http://www.microsoft.com/office/word/default.asp

Appendix A – Intrusion Mailing list reply by Donald Smith

```
-----Original Message-----
From: Smith, Donald [mailto:Donald.Smith@qwest.com]
Sent: September 11, 2002 19:22
To: Etienne Lebel; intrusions@incidents.org
Subject: RE: LOGS: GIAC GCIA version 3.2 Practical Detect - CWD
exploit
an d WU -FTP file completion exploit




> -----Original Message-----
> From: Etienne Lebel [mailto:elebel@recruitsoft.com]
> Sent: Wednesday, September 11, 2002 9:10 AM
> To: intrusions@incidents.org
> Subject: LOGS: GIAC GCIA version 3.2 Practical Detect - CWD
> exploit and
> WU -FTP file completion exploit
>
>
> Excuse me of sending on the same day the same detect,
>
> but my detect was already done.  And after reading the
>
> previous detect we didn't come to the same observation.
>
> So it's all up to you to verify who got the right analysis.
There maybe more then one right answer. There often is.

>
>
>
> So give me your comment about my detect.
>
>
>
> Thanks
>
>
>
>
>
> ---------------------------------------------------------------
> ----------
>
>
>
>
>
> Detection Data
>
>
>
> Snort Alert:
```

```
>
>
>
> [**] FTP EXPLOIT CWD overflow [**]
>
> 07/06-18:51:37.964488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
>
> len:0x23E
>
> 134.126.133.162:2103 -> 46.5.180.133:21 TCP TTL:51 TOS:0x0 ID:10390
>
> IpLen:20 DgmLen:560 DF
>
> ***AP*** Seq: 0xBB7E1B4C  Ack: 0x8599DDFF  Win: 0x16D0  TcpLen: 32
>
> TCP Options (3) => NOP NOP TS: 30355649 6749585
>
> 43 57 44 20 30 30 30 30 30 30 30 30 30 30 30 30  CWD 000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
>
> 30 30 30 30 F0 FC 40 31 07 08 98 5F 08 08 EB 0C  0000..@1..._....
>
> EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
>
> EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
>
> EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
>
> EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
```

```
>
> EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
>
> EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
>
> EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
>
> EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
>
> EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
>
> EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
>
> EB 0C EB 0C 90 90 90 90 90 90 90 90 90 90 90 90  ................
>
> 31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1  1.C..tQ.-....P..
>
> 6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59  j.X......1.....Y
>
> 6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD  j.X........Yj.X.
>
> 80 EB 05 E8 ED 0A                                ......
>
>
>
> =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
> =+=+=+=+=+
>
> =+
>
>
>
> [**] FTP wu-ftp file completion attempt { [**]
>
> 07/06-18:51:38.004488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
>
> len:0x52
>
> 134.126.133.162:2103 -> 46.5.180.133:21 TCP TTL:51 TOS:0x0 ID:10391
>
> IpLen:20 DgmLen:68 DF
>
> ***AP*** Seq: 0xBB7E1D48  Ack: 0x8599E008  Win: 0x1920  TcpLen: 32
>
> TCP Options (3) => NOP NOP TS: 30355653 6749588
>
> 43 57 44 20 7E 2F 7B 2E 2C 2E 2C 2E 2C 2E 7D 0A  CWD ~/{.,.,.,.}.
>
>
>
> =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
> =+=+=+=+=+
>
> =+
>
>
>
> [**] FTP wu-ftp file completion attempt { [**]
```

```
>
> 07/06-18:51:38.164488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
>
> len:0x49
>
> 134.126.133.162:2103 -> 46.5.180.133:21 TCP TTL:51 TOS:0x0 ID:10399
>
> IpLen:20 DgmLen:59 DF
>
> ***AP*** Seq: 0xBB7E1DB0  Ack: 0x8599E13F  Win: 0x1920  TcpLen: 32
>
> TCP Options (3) => NOP NOP TS: 30355670 6749605
>
> 43 57 44 20 7E 7B 0A                             CWD ~{.
>
>
>
> =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
> =+=+=+=+=+
>
> =+
>
>
>
> 1. Source of Trace:
>
>
>
> This trace came from the incidents web site.
>
> http://www.incidents.org/logs/RAW/2002.6.6.  This file is a
> snort alert
>
> log file in tcpdump format.  The network layout is unknown,
> but it looks
>
> like all alerts come from a sensor that monitors a class B network
>
> 45.5.0.0/21.  This network belongs to Interop Shop Network.
> Look at the
>
> ARIN report:
>
>
>
> OrgName:    Interop Show Network
>
> OrgID:      ISN
>
>
>
> NetRange:   45.0.0.0 - 45.255.255.255
>
> CIDR:       45.0.0.0/8
>
> NetName:    SHOWNETA
>
> NetHandle:  NET-45-0-0-0-1
```

```
>
> Parent:
>
> NetType:    Direct Assignment
>
> NameServer: DNS.INTEROP.NET
>
> NameServer: SOLARIS.CC.VT.EDU
>
> Comment:
>
> RegDate:    1991-09-09
>
> Updated:    1998-11-11
>
>
>
> TechHandle: JM1174-ARIN
>
> TechName:   Martin, Jim
>
> TechPhone:  5-5691
>
> TechEmail:  jim@daedelus.com
>
> 2. Detect Generated By:
>
>
>
> This detect was generated with snort 1.8.7 on my Mandrake
> Linux system.
>
> The log file was store in tcpdump format. I replay the log with
snort
>
> and the -r option that let me the ability to analyze tcpdump
> file.  The
>
> ruleset that I used was the stable ruleset that can be downloaded
from
>
> the snort website.
>
> http://www.snort.org/dl/signatures/snortrules-stable.tar.gz
> If you are
>
> not familiar with the way snort log alert you should read the snort
>
> manual.  The snort rules that trigger the alerts were:
>
>
>
> *     alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP CMD
>
> overflow"; flags:A+; dsize:>100; content:"CMD "; nocase;
>
> classtype:attempted-admin; sid:1621; rev:5;)
>
```

```
>
>
> *     alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP wu-ftp
>
> file completion attempt {"; flags:A+; content:"~"; content:"{";
>
> reference:cve,CAN-2001-0886; reference:bugtraq,3581;
>
> classtype:misc-attack; sid:1378;  rev:7;)
>
>
>
> 3. Probability that the source address was spoofed:
>
>
>
>       The IP address is definitely not spoofed.   Because each
packet
>
> we saw here are parts of a FTP connection and required the
> establishment
>
> of a TCP connection to have the desired effect on the FTP
> server.  Which
>
> is impossible to perform with a spoofed IP.   But the remote
computer
```

not impossible. If the system has a predictable ISN. Or if your anywhere
between the victim and host you wish to spoof you can preform a man in the middle style
attack.

```
>
> can be compromised and the hacker can be using it to perform is
damage
True
>
> on the Internet.  We also have to consider that those packets are
>
> crafted ones, but if it's right, those packets were inoffensive.

See above about spoofing

>
>
>
> 4. Description of the attack:
>
>
>
>       This look like the attackers was scanning some hosts on the
>
> network to find vulnerable FTP server to the CWD exploit and WU-
FTPD
>
> file completion exploit.  The attacker always tries 3 types of
```

```
attacks
>
> against each FTP server.  The first type of attack only applies to
the
>
> Vermillion FTP server.  When perform against the FTP daemon the
attack
>
> can let the user execute command on the remote host and then the
FTP
>
> server became unavailable.  But to perform this attack, the
attacker
>
> must send 3 CWD commands with more than 504 characters in the
command.
>
> In that case the attacker sends 3 CWD commands, but the 2 others
don't
>
> contain 504 characters, so this first attempt was not the
> VFTP exploit.
```

What happens if you only send one?
Would it reveal a vulnerability without exploiting it?

```
>
> So we can now eliminate the VFTP exploit.  But the 2 others
> CWD command
>
> show another type of attack.  The "wu-ftp file completion" buffer
>
> overflow attack. This one is a common attack against the
> WU-FTPD server
>
> 2.60 and 2.61.  This attack uses an overflow in the glob function
of
>
> GLIBC that was not handled by the FTP daemon.  To perform this
attack,
>
> the logged user has to send a command with the "{" character.  So
once
```

Why? Won't any globbing character work?

```
>
> the attack is performed, the hacker can execute command on the
remote
>
> host as the user running the FTP server or perform a DOS against
this
>
> server.  But in that case we didn't see any evidence of exploit in
the
>
> command.  So maybe its only a DOS.
>
```

```
>
>
> Reference:
>
> CWD Exploit:
>
> -http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-1058
>
>
>
> -http://cgi.nessus.org/plugins/dump.php3?id=10293
>
>
>
> File completion :
>
> -http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0886
>
>                       -http://www.securityfocus.com/bid/3581
>
>
>
> 5. Attack Mechanism:
>
>
>
>       To perform is probe the attacker was able to log into the FTP
>
> server (with a normal user or anonymously).  Then the
> attacker launch is
>
> 3 type of attack consecutively.  In the log I didn't have any
evidence
>
> that the user log into a FTP server at the following address:
>
> 46.5.180.133, 46.5.180.134, 46.5.180.135, 46.5.180.151,
46.5.180.153.
>
> But it doesn't make sense to send those packets to a host that
doesn't
>
> run an FTP server.  It's pretty hard to identify if the attacker
use a
>
> script to perform is attack, but it look like it was scripted,
because
>
> the interval between each host is fairly small.  We have to
consider
>
> that the user have to login, then enter is exploit, validate If it
is
>
> working before starting with a new target.
>
>
>
```

```
> 6. Correlations:
>
>
>
>       The WU-FTP is a well know FTP server and had several security
>
> flaws during the past years.  Also the IP address of the
> attacker looks
>
> like a familiar IP for FTP hacks or scan over the Internet.  After
a
>
> search on google.com and security focus, I found a links that show
>
> evidence about this IP.  A message posted on the Intrusion
> Mailing list
>
> by Ken Connelly that show a large portscan attempt against its
network
>
> range.  In that list we saw the source IP 134.126.133.162
> performing and
>
> FTP portscan.
>
> <http://cert.uni-stuttgart.de/archive/intrusions/2002/07/msg00
> 041.html>.
>
> Also after a search on www.dshield.org I found that 90080 records
are
>
> stored in the dshield database and those records are against 90078
>
> targets.  So this represents a mass scanning IP that try to find
>
> vulnerable FTP server to take control of.
>
>
>
> 7. Evidence of active targeting:
>
>
>
>       When looking only at the alert log, we can conclude that
these
>
> attacks were directly targeted to those 5 IP mention above.  But
it's
>
> pretty difficult to see if the attacker had scanned the
> network before,
>
> because we only got the alert file and not the entire traffic log.
>
> After a quick review of the past 5 days of alert and the
> portscan files
>
> that were generated when I review the log.  I didn't find any trace
```

```
of
>
> scanning or other attack from that source IP.  Maybe the attacker
had
>
> already scanned the network before or he had the knowledge the
network
>
> services running on each host.
>
>
>
> 8. Severity:
>
>
>
> *Since we didn't have any information about the network where
> the trace
>
> came from and the criticality of the target we will assume the
worst
>
> case scenario.
>
>
>
> Severity = (Criticality + Lethality) - (System
> Countermeasures + Network
>
> Countermeasure)
>
>
>
> Criticality= 5     (We assume that those system are critical)
>
> Lethality=3        (Looks like an HTTP and FTP server)
>
> System Countermeasure = 1 (Assume none)
>
> Network Countermeasure = 1 (Assume none)
>
>
>
> 6= (5+3)-(1+1)
>
>
>
> Severity: 6
>
>
>
> 9. Defensive Recommendation:
>
>
>
>       To protect you against this type of attack or any further one
>
> that could occur against the WU-FTPD server.  I recommend you to
```

```
use
>
> another implementation of FTP, like PROFTPD
><http://www.proftpd.net> or
>
>to always perform updates of your WU-FTP server at
>
><http://www.wu-ftpd.org>.  Also restrict the access to your FTP
server
>
>to only people that need it.  Because FTP is not a secure file
transfer
>
>protocol, I also suggest you to consider using FTP over SSH or SFTP
to
>
>perform data transfer when it's possible.
>
>
>
>10. Multiple Choice Questions:
>
>
>
>What is the data port for a standard FTP file transfer?
>
>
>
>a.21
>
>b.20
>
>c.2121
>
>d.8080
>
>
>
```

>Answer is: b