Global Information Assurance Certification Paper

GCIA Practical
v. 3.3

by
Travis Bow
December 31 2003

Table of Contents

Part 1 "Describe the State of Intrusion Detection"

Introduction: "Slammer" (Written June 24, 2003)

Part 1 of this practical will attempt to provide a discussion of the Microsoft SQL Slammer worm. The discussion will entail some background on SQL Slammer worm, method of infection, capabilities of the SQL Slammer worm in terms of what it can do to hosts and networks, analysis on what Snort rules triggers this alert, and in conclusion remediation and closing thoughts. The SQL Slammer worm was chosen because I was interested in the stimulus that caused the massive onslaught of denied firewall logs and thousands of IDS alerts that are still prevalent today. The Slammer worms' aliases are otherwise known as WORM_SQLP1434.A (Trend Micro), Sapphire (F-Secure), W32.SQLExp.Worm (Symantec), Worm.SQL.Helkern (Kaspersky) and W32/SQLSlammer.worm (Network Associates). Snort, was the tool of choice during the analysis phase of assignment #1.

Background:

Internet Security Systems (ISS) on January 25, 2003 released a security alert regarding the Microsoft SQL Slammer worm propagation that stated, "Billions of attacks have been detected in the last 12 hours…."[1]. ISS also noted that SQL Slammer infected, "…over 200,000 Microsoft SQL Server installations in less than 10 minutes."[2] Basically, within less than 24 hours the SQL Slammer worm had infected thousands of hosts generating tremendous amounts of Internet/network traffic. The SQL Slammer worm uses a vulnerability found in various Microsoft products. The key products are Microsoft SQL Server and Microsoft SQL Desktop Engine (MSDE).

What is Microsoft SQL Server? According to Microsoft, "…SQL Server 2000 is a fully enterprise-class database product, providing core support for Extensible Markup Language (XML) and Internet queries."[3], "With the lowest implementation and maintenance costs in the industry….".[4] What is Microsoft SQL Desktop Engine? MSDE is a basically a "Desktop" version of SQL Server when instances of a non-enterprise class database product is required. MSDE is often times integrated into third party applications that do not require SQL server.

[1] http://www.iss.net/issEn/delivery/xforce/alertdetail.jsp?oid=21824
[2] https://gtoc.iss.net/documents/summaryreport.pdf (January – March 2003)
[3] http://www.microsoft.com/sql/evaluation/overview/default.asp
[4] http://www.microsoft.com/sql/evaluation/overview/default.asp

The SQL Slammer worm takes advantage of a stacked-based buffer overflow vulnerability (CVE CAN-2002-0649)[5] found in the Microsoft SQL Resolution Service. The Resolution Service purpose is to aid versions of SQL server with the ability to host multiple instances of SQL server on a particular host. Many times organizations make use of multiple instances of SQL Server on one host to minimize hardware and maintenance costs. For a particular machine to host multiple instances the use of unique names for each instance is required. The Resolutions Service is able to assist with hosting multiple instances by providing "…. a way for clients to query for the appropriate network endpoints to use for a particular SQL Server instance."[6] The default SQL Server instance will reside on TCP port 1433 while the Resolution Service resides on UDP port 1434 commonly referred to as the Microsoft SQL Monitor port. Once the default instance assumes control of TCP port 1433, other instances of SQL server must use another port. When a MS SQL client attempts to connect to a non-default instance the client uses the Resolution Service Port (UDP1434). The Resolution Service then determines if it contains the instance requested. If the instance is registered, it then responds to the client with information including the port the requested instance resides on. Normally clients will send a first payload byte of 0x02 to the Resolution Service port to discover how they should connect to the server, followed by 16 bytes of data that includes the name of the SQL server instance. Unfortunately, the ability to host multiple instances of SQL Server is installed by default, leaving unpatched machines vulnerable.

Method of Infection:

The SQL Slammer worm sends specifically crafted packets with the first byte of payload being 0x04 to the SQL Server Resolution port. This informs the server that the next fifteen bytes of data will be the named instance requested, followed by 0x00, which specifies the end of the name. However the Slammer worm sends a string of 0x01s that surpasses the 16 byte allocation for the name of the SQL server instance. The buffer that contains the name in this case goes unchecked and dumps the data into memory. At this point the service attempts to open the registry key. The registry the host attempts to open is, "(HKLM\Software\Microsoft\Microsoft SQL Server\……..\MSSQLServer\CurrentVersion)" "(dots in this key denote the Registry key branch SQL Monitor tries to access - for SQLSlammer.worm these bytes are a long series of unprintable 01 symbols following 04 which is a type of request)."[7] Because the buffer goes uncheck the overflowed data will overwrite the stack and the saved return address and enable execution of arbitrary code. According to Network Associates, the vulnerability slammer uses is found in the SSNETLIB.DLL, which handles the 0x04 found in the first byte of payload sent to the Resolution port.

[5] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0649
[6] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms02-039.asp
[7] http://vil.nai.com/vil/content/v_99992.htm

Once the host is compromised, ISS states that the Slammer worm loads Kernel32.DLL and Ws2_32.DLL and starts to send the 376 bytes of exploit to a random set of host on port 1434 UDP. The random IP addresses are derived from a call onto "GetTickCount". From this point the compromised host continuously sends the slammer worm until the SQL server process either shuts down or crashes.

Perhaps another clue for system administrators that infection has occurred on their network would be if they suddenly see large quantities of ICMP Port/Host Unreachable messages. The reasoning is if the worm sends UDP packets destined to invalid hosts or hosts that do not have UPD port 1434 open then ICMP port/host unreachables will follow. In addition, network logs should be displaying large amounts of UDP 1434 traffic.

Listed below is a list of software that is vulnerable to the exploit used by the Slammer worm, a Stack-based buffer overflow, listed by SecurityFocus. [8]

Microsoft Data Engine 2000
+ Akiva WebBoard 6.1
+ BindView bv-Admin for Microsoft Exchange
+ BindView bv-Admin for Windows 7.0
+ BindView bv-Admin for Windows Migration
+ BindView bv-control for Active Directory 7.0.2
+ BindView bv-Control for Internet Security 7.0.1
+ BindView bv-Control for Microsoft Exchange 7.0
+ BindView bv-Control for Microsoft SQL Server 7.0
+ BindView bv-Control for Microsoft SQL Server 7.0.1
+ BindView bv-Control for Windows 7.0.2
+ CARI-RUSCO Secure Perfect 3.0
+ CCH Equity Compliance Insider Reporting Module
+ Collins Medical Plus 2000
+ Computer Associates Unicenter
+ Computer Associates Unicenter RC/Update 6.0
+ Computer Associates Unicenter RC/Update 6.1
+ CSIRO BioLink Software 1.5
+ DATA.TXT Corporation Time Matters 3.0
+ DATA.TXT Corporation Time Matters 4.0
+ Dell OpenManage IT Assistant 5.0
+ Dell OpenManage IT Assistant 6.0
+ Express Metrix Express Software Manager 5.0
+ Express Metrix Express Software Manager 6.0
+ Express Metrix Express Software Manager 6.0.1
+ Express Metrix Express Software Manager 6.0.2
+ Fluke Networks Optiview Network Inspector 5.0
+ HP Openview Internet Services 4.0
+ HP Openview Internet Services 4.5
+ HP Openview Operations for Windows 6.0
+ HP Openview Operations for Windows 7.0
+ HP Openview Operations for Windows 7.1
+ HP Openview Reporter 2.0.2
+ HP Openview Reporter 3.0
+ ISI Infortel for Windows 4.0
+ ISI Infortel for Windows 5.1
+ ISI Infortel for Windows 5.2
+ ISI Infortel for Windows 5.4
+ Journyx Timesheet 2.0
+ Journyx Timesheet 4.5
+ Journyx Timesheet 4.5 m2

[8] http://www.securityfocus.com/bid/5311

+ Journyx Timesheet 4.5 m3
+ Journyx Timesheet 4.6
+ Journyx Timesheet 5.0
+ Microsoft .NET Framework 1.0
+ Microsoft .NET Framework 1.0 SP1
+ Microsoft .NET Framework 1.1
+ Microsoft .NET Framework SDK 1.0
+ Microsoft Application Center 2000
+ Microsoft Biztalk Server 2002 Partner Edition
+ Microsoft FrontPage 2000 Server Extensions SR 1.0
+ Microsoft FrontPage 2000 Server Extensions SR 1.1
+ Microsoft FrontPage 2000 Server Extensions SR 1.2
+ Microsoft FrontPage 2000 Server Extensions SR 1.3
+ Microsoft Great Plains 5.0
+ Microsoft Great Plains 5.5
+ Microsoft Great Plains 5.5.1
+ Microsoft Great Plains 7.0
+ Microsoft Office 2000
+ Microsoft Office 2000 SP2
+ Microsoft Office 2000 SP2
+ Microsoft Office 2000 SR1
+ Microsoft Office 2000 Chinese Version
+ Microsoft Office 2000 Japanese Version
+ Microsoft Office 2000 Korean Version
+ Microsoft Office XP
+ Microsoft Office XP SP1
+ Microsoft Office XP Developer Edition
+ Microsoft Project Central Server
+ Microsoft SharePoint Portal Server 2001
+ Microsoft SharePoint Portal Server 2001 SP1
+ Microsoft SharePoint Team Services
+ Microsoft SQL Server 2000
+ Microsoft SQL Server 2000 SP1
+ Microsoft SQL Server 2000 SP2
+ Microsoft SQL Server 2000 SP3
+ Microsoft Visio 2000 Enterprise Edition
+ Microsoft Visio Enterprise Network Tools
+ Microsoft Visual FoxPro 6.0
+ Microsoft Visual FoxPro 7.0
+ Microsoft Visual FoxPro 7.0 SP1
+ Microsoft Visual Studio .NET Academic Edition
+ Microsoft Visual Studio .NET Enterprise Architect Edition
+ Microsoft Visual Studio .NET Enterprise Developer Edition
+ Microsoft Visual Studio .NET Professional Edition
+ Microsoft Visual Studio .NET Trial Edition
+ Microsoft Windows Server 2003 Standard Edition
+ Microsoft Windows XP Embedded
+ Microsoft Windows XP Embedded SP1
+ MIP NonProfit Series Pro 4.3
+ MIP NonProfit Series Pro 4.4
+ MIP NonProfit Series Pro 4.5
+ NetSupport NetSupport TCO 4.5
+ NetSupport NetSupport TCO 4.5.1
+ Network Associates SupportMagic SQL 4.5
+ Okena StormWatch
+ Peachtree Software Timeslips 6.0
+ Peachtree Software Timeslips 7.0
+ Peachtree Software Timeslips 8.0
+ Peachtree Software Timeslips 9.0
+ Peachtree Software Timeslips 9.0
+ Peachtree Software Timeslips 10.0
+ Peachtree Software Timeslips 11.0
+ QiNetix CommVault Galaxy 4.0.1
+ SalesLogix Corporation SalesLogix 2000.0
+ SmartMax Software MailMax 5.0
+ TeleStream FlipFactory 1.2

+ TeleStream FlipFactory 2.0
+ TeleStream FlipFactory 3.0
+ Veritas Software Backup Exec 9.0
+ VIGILANTe SecureScan NX 2.5
+ Visionary Systems Firehouse Software 3.0.5
+ Visionary Systems Firehouse Software 5.0
+ Visionary Systems Firehouse Software 5.0.2 5
+ Visionary Systems Firehouse Software 5.4
+ Wonderware InTouch 7.11
+ Xerox CentreWare Web 1.0

Microsoft SQL Server 2000 SP2

Microsoft SQL Server 2000 SP1

- Microsoft Windows 2000 Workstation
- Microsoft Windows 2000 Workstation SP1
- Microsoft Windows 2000 Workstation SP2
- Microsoft Windows NT 4.0 SP5
- Microsoft Windows NT 4.0 SP6
- Microsoft Windows NT 4.0 SP6a

Microsoft SQL Server 2000

- Microsoft Windows 2000 Workstation
- Microsoft Windows 2000 Workstation SP1
- Microsoft Windows 2000 Workstation SP2
- Microsoft Windows NT 4.0
- Microsoft Windows NT 4.0 SP5
- Microsoft Windows NT 4.0 SP6
- Microsoft Windows NT 4.0 SP6a

Veritas Software Backup Exec 9.0

RESULTS

The Slammer worm is not necessarily destructive to the compromised host per se. In fact the worm only exists in memory, does not modify any files, and leaves no backdoor. Slammer merely tries to replicate itself. However, with that stated, the Slammer worm generates enormous amounts of traffic, as it is unlikely to scan local subnets. The large amounts of Internet traffic could overwhelm networks causing several types of DOS (Denial of Service) instances. For example, a network type of DOS, where a host attempts to access services on another network, but is unable to because of the limited amount of legitimate traffic that can traverse the networking equipment. The second scenario is the Slammer worm can cause the Resolution process to fail resulting in legitimate hosts unable to access the server, in other words DOS to the application. According to statements made by ISS, Slammer caused a "general slowdown of the entire Internet." [9]

Analysis of Snort Rule

The Analysis of the Snort rule was accomplished utilizing Snort version 1.9.1, IIS 5, ACID v9.6b24 and MYSQL v4.0.12 on a Windows 2000 platform. The alerts and data were captured on April 23, 2003. Two alerts were randomly selected to analyze. Alert #1, displays host 210.58.80.47 attempted connection to host

[9] https://gtoc.iss.net/documents/summaryreport.pdf

my.net.1.1 on UDP port 1434. Alert #2 displays host 24.136.118.40 attempted connection to host my.net.1.1.

Shown below are ACID outputs of Snort alerts with the application layer dump.

Alert #1:

sourceaddr	destaddr	Ver	Hdr	TOS	Len
210.58.80.47	my.net.1.1	4	5	0	404
ID	flags	offset	TTL	chksum	
64429	0	0	115	40037	

sourceport	destport	length
3056	**1434**	384

Slammer Payload
length = **376**

```
000 : 04 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
010 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
020 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
030 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
040 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
050 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
060 : 01 DC C9 B0 42 EB 0E 01 01 01 01 01 01 01 70 AE   ....B.........p.
070 : 42 01 70 AE 42 90 90 90 90 90 90 90 90 68 DC C9   B.p.B........h..
080 : B0 42 B8 01 01 01 01 31 C9 B1 18 50 E2 FD 35 01   .B.....1...P..5.
090 : 01 01 05 50 89 E5 51 68 2E 64 6C 6C 68 65 6C 33   ...P..Qh.dllhel3
0a0 : 32 68 6B 65 72 6E 51 68 6F 75 6E 74 68 69 63 6B   2hkernQhounthick
0b0 : 43 68 47 65 74 54 66 B9 6C 6C 51 68 33 32 2E 64   ChGetTf.llQh32.d
0c0 : 68 77 73 32 5F 66 B9 65 74 51 68 73 6F 63 6B 66   hws2_f.etQhsockf
0d0 : B9 74 6F 51 68 73 65 6E 64 BE 18 10 AE 42 8D 45   .toQhsend....B.E
0e0 : D4 50 FF 16 50 8D 45 E0 50 8D 45 F0 50 FF 16 50   .P..P.E.P.E.P..P
0f0 : BE 10 10 AE 42 8B 1E 8B 03 3D 55 8B EC 51 74 05   ....B....=U..Qt.
100 : BE 1C 10 AE 42 FF 16 FF D0 31 C9 51 51 50 81 F1   ....B....1.QQP..
110 : 03 01 04 9B 81 F1 01 01 01 01 51 8D 45 CC 50 8B   ..........Q.E.P.
120 : 45 C0 50 FF 16 6A 11 6A 02 6A 02 FF D0 50 8D 45   E.P..j.j.j...P.E
130 : C4 50 8B 45 C0 50 FF 16 89 C6 09 DB 81 F3 3C 61   .P.E.P........<a
140 : D9 FF 8B 45 B4 8D 0C 40 8D 14 88 C1 E2 04 01 C2   ...E...@........
150 : C1 E2 08 29 C2 8D 04 90 01 D8 89 45 B4 6A 10 8D   ...).......E.j..
160 : 45 B0 50 31 C9 51 66 81 F1 78 01 51 8D 45 03 50   E.P1.Qf..x.Q.E.P
170 : 8B 45 AC 50 FF D6 EB CA                           .E.P....
```

Alert #2:

sourceaddr	destaddr	Ver	Hdr	TOS	length
24.136.118.40	My.net.1.1	4	5	0	404
ID	flags	offset	TTL	chksum	
16842	0	0	106	42113	

sourceport	destport	length
7346	**1434**	384

length = **376**

```
000 : 04 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
010 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
020 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
030 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
040 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
050 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
060 : 01 DC C9 B0 42 EB 0E 01 01 01 01 01 01 01 70 AE   ....B.........p.
070 : 42 01 70 AE 42 90 90 90 90 90 90 90 90 68 DC C9   B.p.B........h..
080 : B0 42 B8 01 01 01 01 31 C9 B1 18 50 E2 FD 35 01   .B.....1...P..5.
090 : 01 01 05 50 89 E5 51 68 2E 64 6C 6C 68 65 6C 33   ...P..Qh.dllhel3
0a0 : 32 68 6B 65 72 6E 51 68 6F 75 6E 74 68 69 63 6B   2hkernQhounthick
0b0 : 43 68 47 65 74 54 66 B9 6C 6C 51 68 33 32 2E 64   ChGetTf.llQh32.d
0c0 : 68 77 73 32 5F 66 B9 65 74 51 68 73 6F 63 6B 66   hws2_f.etQhsockf
0d0 : B9 74 6F 51 68 73 65 6E 64 BE 18 10 AE 42 8D 45   .toQhsend....B.E
0e0 : D4 50 FF 16 50 8D 45 E0 50 8D 45 F0 50 FF 16 50   .P..P.E.P.E.P..P
0f0 : BE 10 10 AE 42 8B 1E 8B 03 3D 55 8B EC 51 74 05   ....B....=U..Qt.
100 : BE 1C 10 AE 42 FF 16 FF D0 31 C9 51 51 50 81 F1   ....B....1.QQP..
110 : 03 01 04 9B 81 F1 01 01 01 01 51 8D 45 CC 50 8B   ..........Q.E.P.
120 : 45 C0 50 FF 16 6A 11 6A 02 6A 02 FF D0 50 8D 45   E.P..j.j.j...P.E
130 : C4 50 8B 45 C0 50 FF 16 89 C6 09 DB 81 F3 3C 61   .P.E.P........<a
140 : D9 FF 8B 45 B4 8D 0C 40 8D 14 88 C1 E2 04 01 C2   ...E...@........
150 : C1 E2 08 29 C2 8D 04 90 01 D8 89 45 B4 6A 10 8D   ...).......E.j..
160 : 45 B0 50 31 C9 51 66 81 F1 78 01 51 8D 45 03 50   E.P1.Qf..x.Q.E.P
170 : 8B 45 AC 50 FF D6 EB CA                           .E.P....
```

The first step in the analysis process was to investigate the rule that triggered the alert. Shown below is the Snort rule that triggered the alert and the highlighted sections of the rule are used to analyze the data.

```
alert udp $EXTERNAL_NET any -> $HOME_NET 1434 (msg:"MS-SQL Worm
propagation attempt"; content:"|04|"; depth:1; content:"|81 F1 03 01 04 9B 81 F1 01|";
content:"sock"; content:"send"; reference:bugtraq,5310; classtype:misc-attack;
reference:bugtraq,5311; reference:url,vil.nai.com/vil/content/v_99992.htm; sid:2003;
rev:2;)
```

I then proceeded to analyze and then compare the rule to the Snort generated data. Following the rule from left to right, this would substantiates the alerts were not false positives and why the alerts triggered.

Both alerts contained the following: (Highlighted in Yellow)

1. UDP traffic
2. Destination ports of 1434
3. 1st byte of payloads (Depth:1) had a content of 0x04
4. Hex content string of 81 F1 03 01 04 9B 81 F1 01
5. Content of ASCII characters "sock" and "send"

Additional information gathered from an ISS alert[10] states the payload of the SQL Slammer worm is 376 bytes, which was consistent with both Snort alerts.

Trends:

The following graphs depict the activity of Slammer traffic on my home network. My network included one host running SQL server, however this host is behind a firewall that does not permit publicly accessibility to port 1434 UDP. The Snort IDS was located outside the firewall. To create the graphs I utilized ACID to group and sort all of the SQL Slammer alerts together. The signature that triggered these alerts was from the default rule set from Snort (SID=2003). The signature is provided above in the Analysis of Snort Rule section. Fig. 1 displays alert occurrences over a time frame of April 4, 2003 to June 23, 2003. As you there were some days where close to 180 attacks occurred. April 4th to April 6th and May 11th to 17th were the peak time frames. Fig. 2 aggregates the alert data from the same time period and displays when the alerts occur over the time of day. Interestingly enough the bulk of the alerts were generated from 12:00 am to 3:00am and 4:00pm and 7:00pm. During 10:00 am and 1:00pm there were very few alerts.

Fig. 1

[10] http://www.iss.net/issEn/delivery/xforce/alertdetail.jsp?oid=21824

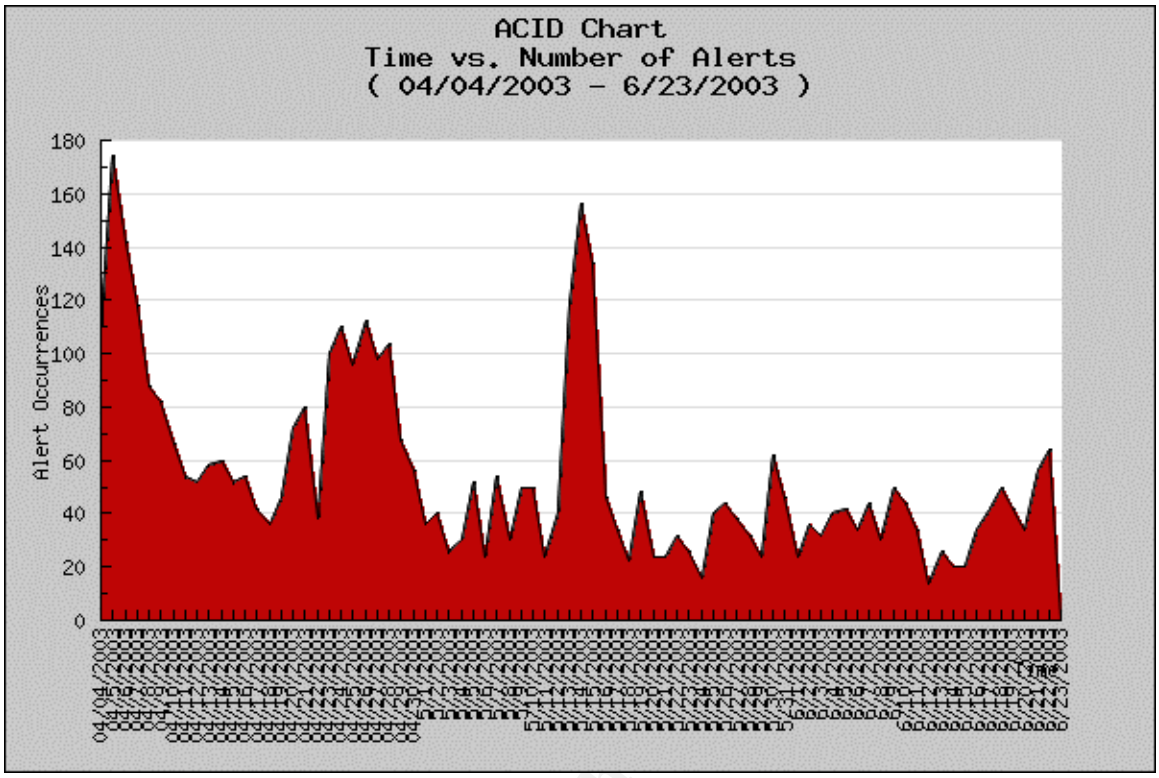
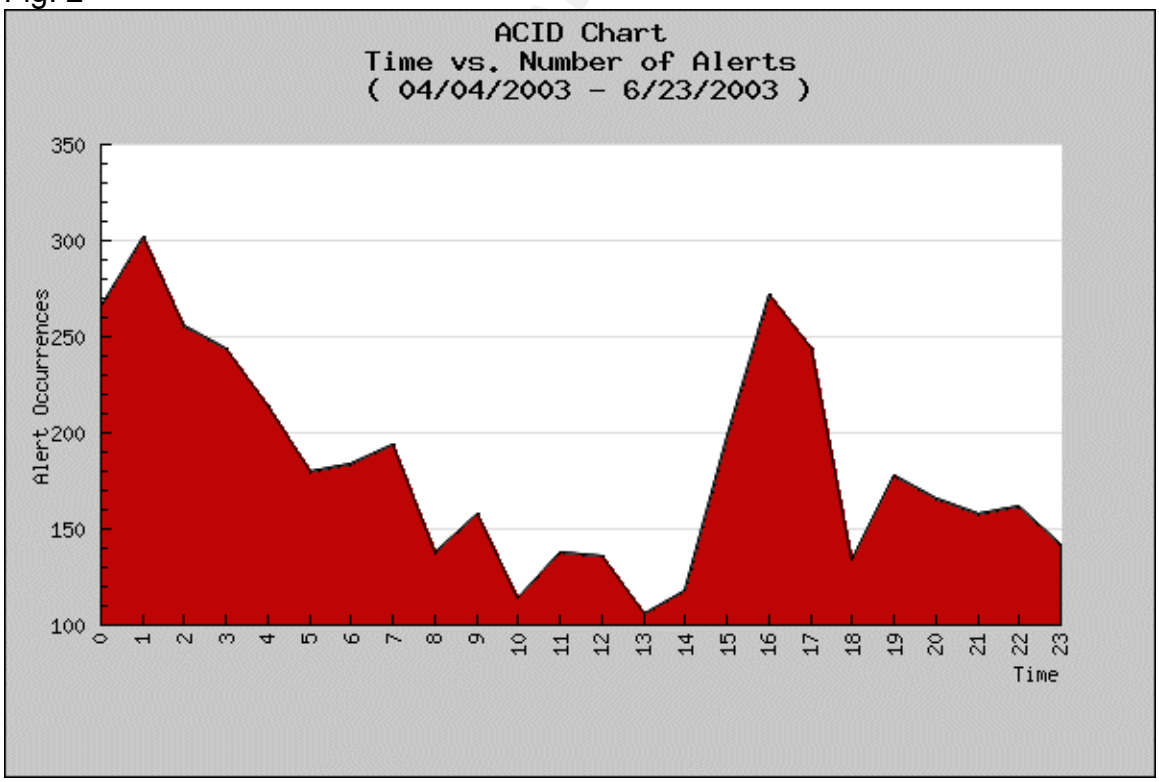


Fig. 2

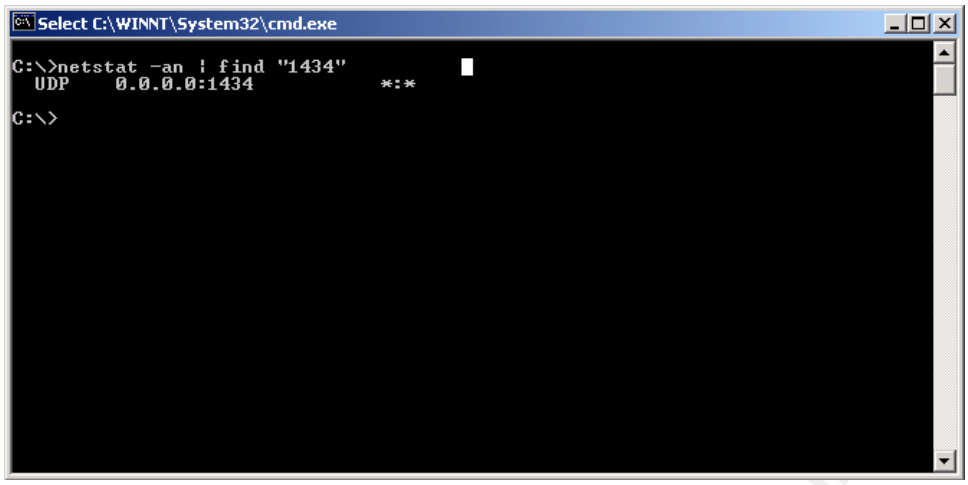


Removal:

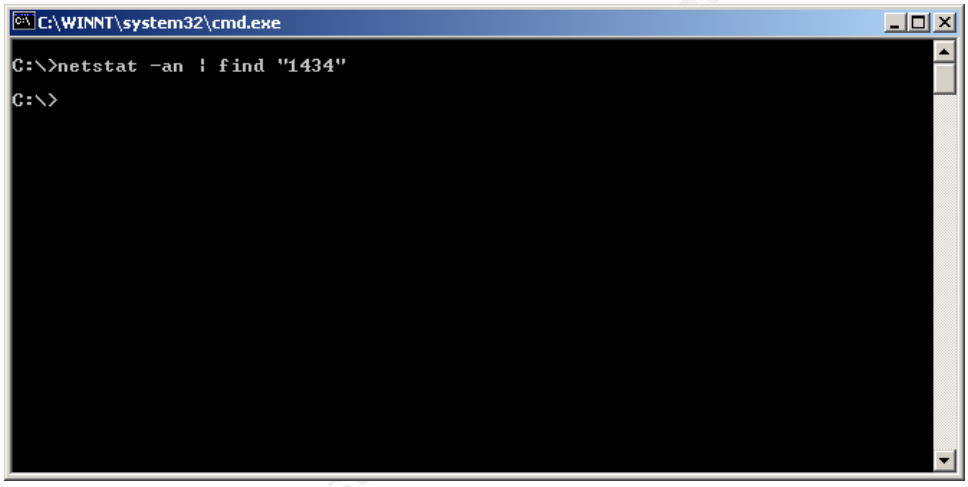
As stated previously, the Slammer worm only exists in memory. A simple reboot of the infected host will result in the removal of infection. However, to prevent reoccurrences system administrators should correct the stack based buffer overflow vulnerability found in the SSNETLIB.DLL file by applying a patch found at http://www.microsoft.com/Downloads/details.aspx?displaylang=en&FamilyID=DCFDCBE9-B4EB-4446-9BE7-2DE45CFA6A89. This patch works for both Microsoft SQL Server and MSDE. Or you can upgrade your version of MSDE to MSDE Release A that is not vulnerable to SQL Slammer worm. To verify that the vulnerable file has been updated make certain that the version of the SSNETLIB.DLL file located in \MSSSQL\BINN folder is 2000.80.636.0 according to Microsoft[11].
The Resolutions Service on port UDP 1434 can be turned off, however clients would need aliases setup so they would know which port to connect for the instance of SQL they are looking for. Another solution is to download and install the latest service pack of SQL Server, which currently is SP3a (July 12, 2003). This service pack gives the system administrators the ability to prevent the SQL host from listening on port1434 if networking is disabled. This vulnerability can be triggered by a single (UDP) datagram which emphasizes the need for proper ingress filtering on routers and firewalls to prevent accessibility to vulnerable hosts and the appropriate egress filters to ensure your infected hosts are unable to compromise others.

If you are not sure if your host is running Microsoft SQL server or MSDE, then you can use the following command on the host to determine if they are. The command used would be netstat –an | find “1434”. Netstat displays TCP/IP connections. The picture below indicates what the response would be from a host that is probably running SQL or MSDE.

[11] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms02-039.asp



The picture below indicates what the response would be from a host that is not running SQL or MSDE.



In addition to this you can use other tools like Nmap, ISS Internet Scanner and others to determine if the UDP port 1434 is listening on your hosts.

The picture below indicates what the response would be from a host that is probably running SQL or MSDE after running the nmap –sU -p 1434 172.16.10.10 command.

```
[root@ID root]# nmap -sU -p 1434 172.16.10.10

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (172.16.10.10):
Port       State       Service
1434/udp   open        ms-sql-m
```

The picture below indicates what the response would be from a host that is not running SQL or MSDE after running the nmap –sU -p 1434 172.16.10.11 command.

```
[root@ID root]# nmap -sU -p 1434 172.16.10.11

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
The 1 scanned port on (172.16.10.11) is: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

You could also run this nmap command nmap -sS -p 1433 172.16.10.10 on TCP port 1433 to discover if a host is probably running SQL or MSDE.

Response if host is running SQL or MSDE

```
[root@ID root]# nmap -sS -p 1433 172.16.10.10

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (172.16.10.10):
Port       State       Service
1433/tcp   open        ms-sql-s

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

Response if host is not running SQL or MSDE

```
[root@ID root]# nmap -sS -p 1433 172.16.10.11

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
The 1 scanned port on (172.16.10.11) is: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
[root@ID root]#
```

Conclusion:

The troubling part of the SQL Slammer worm was not the exploit itself. It was the fact that the protection to prevent infection had been available since July 2002. Quite scary, as this indicated a large percentage of Internet accessible SQL Sever hosts are not “patched”. Also, indicating that there may be many networks are probably not secure and do not have the proper ingress and egress filtering on network appliances.

Even today we still receive hundreds of Slammer attempts on our address space. More information about the SQL Slammer worm and the exploits that it uses can be found at the references below.

References:

1. Network Associates
   URL:
   W32/SQLSlammer.worm. Network Associates. 24 June 2003
   <http://vil.nai.com/vil/content/v_99992.htm>
2. SecurityFocus
   URL:
   Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability. Security Focus. 24 June 2003
   <http://www.securityfocus.com/bid/5311>
3. Snort.org
   URL:
   Snort.org. 24 June 2003
   <http://www.snort.org/snort-db/sid.html?sid=2003>
4. Mitre.org
   URL:
   CAN-2003-0209. Common Vulnerabilities and Exposures. 24 June 2003
   http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0649
5. Sophos
   URL:
   Virus writing on the increase, Bugbear-B worm is major irritant of 2003. Sophos. 24 June 2003
   <http://www.sophos.com/pressoffice/pressrel/uk/20030630topten.html>
6. Internet Security Systems
   URL:
   Microsoft SQL Slammer Worm Propagation. Internet Security Systems. 24 June 2003
   <http://www.iss.net/issEn/delivery/xforce/alertdetail.jsp?oid=21824>
7. Microsoft
   URL:
   SQL Server 2000 Service Pack 3a. Microsoft. 24 June 2003
   <http://www.microsoft.com/sql/downloads/2000/sp3.asp>

Part 2: Network Detects

Detect #1

```
[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:35:10.156507 211.47.255.20:46000 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x121C24B1  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:35:13.156507 211.47.255.20:46000 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x121C24B1  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:35:19.166507 211.47.255.20:46000 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x121C24B1  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:35:31.146507 211.47.255.20:46000 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x121C24B1  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:35:42.176507 211.47.255.20:46319 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x13A6B185  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:35:45.156507 211.47.255.20:46319 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x13A6B185  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:35:51.166507 211.47.255.20:46319 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x13A6B185  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
```

```
[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:36:03.146507 211.47.255.20:46319 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x13A6B185  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:36:14.166507 211.47.255.20:46671 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x1661E114  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:36:17.166507 211.47.255.20:46671 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x1661E114  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:36:23.166507 211.47.255.20:46671 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x1661E114  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:36:35.146507 211.47.255.20:46671 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x1661E114  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:36:46.156507 211.47.255.20:47040 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x18C00FAD  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:36:49.176507 211.47.255.20:47040 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x18C00FAD  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-02:36:55.156507 211.47.255.20:47040 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x18C00FAD  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
```

```
[Classification: Misc activity] [Priority: 3]
11/16-02:37:08.116507 211.47.255.20:47040 -> 170.129.118.169:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x18C00FAD  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:54:53.826507 211.47.255.20:35752 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x3EAAD0C3  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:54:56.806507 211.47.255.20:35752 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x3EAAD0C3  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:55:02.826507 211.47.255.20:35752 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x3EAAD0C3  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:55:14.996507 211.47.255.20:35752 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x3EAAD0C3  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:55:28.826507 211.47.255.20:36015 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x40E17268  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:55:34.826507 211.47.255.20:36015 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x40E17268  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:55:46.826507 211.47.255.20:36015 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x40E17268  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
```


© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
11/16-03:55:57.816507 211.47.255.20:36319 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x4304B38E  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:56:00.826507 211.47.255.20:36319 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x4304B38E  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:56:06.826507 211.47.255.20:36319 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x4304B38E  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:56:18.826507 211.47.255.20:36319 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x4304B38E  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:56:29.836507 211.47.255.20:36576 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x45FCB139  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:56:32.836507 211.47.255.20:36576 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x45FCB139  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:56:39.426507 211.47.255.20:36576 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x45FCB139  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-03:56:50.836507 211.47.255.20:36576 -> 170.129.44.181:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x45FCB139  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:19:51.146507 211.47.255.20:34339 -> 170.129.210.216:0
```

```
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB1A713EE  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:19:57.136507 211.47.255.20:34339 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB1A713EE  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:20:09.126507 211.47.255.20:34339 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB1A713EE  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:20:20.136507 211.47.255.20:34851 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB3AFF15D  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:20:23.146507 211.47.255.20:34851 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB3AFF15D  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:20:29.146507 211.47.255.20:34851 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB3AFF15D  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:20:41.136507 211.47.255.20:34851 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB3AFF15D  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:20:52.146507 211.47.255.20:35382 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB5D0E140  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:20:55.136507 211.47.255.20:35382 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
```

```
******S* Seq: 0xB5D0E140  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:21:01.156507 211.47.255.20:35382 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB5D0E140  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:21:13.126507 211.47.255.20:35382 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB5D0E140  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:21:24.156507 211.47.255.20:35928 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB79B7F19  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:21:33.126507 211.47.255.20:35928 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB79B7F19  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/16-12:21:45.136507 211.47.255.20:35928 -> 170.129.210.216:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xB79B7F19  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
```

1) **Source of Trace:**
   The source of the trace was from http://www.incidents.org/logs/Raw/2002.10.16 Listed below are Tcpdump outputs which gives a better understanding of the topology of the Ethernet network. Description of Tcpdump switches are listed below
   -e Print the link-level header on each dump line.
   -n Don't convert addresses to names
   -r Read packets from file

   tcpdump -ne -r ./2002.10.16 outputted the MAC address shown below in red.

05:16:04.896507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 66: 211.47.255.22.39681 > 170.129.228.55.0: S 1909051444:1909051444(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

After discovering both MAC addresses above, I ran the following tcpdump, grep and wc commands to determine if all of the packets were traversing between these two devices. The –L switch used with grep finds " --files-without-match" and the –l switch used with wc would output the count.

```
[root@ID root]# tcpdump -ne -r ./2002.10.16 | grep -L "0:3:e3:d9:26:c0" | wc -l
     0
[root@ID root]# tcpdump -ne -r ./2002.10.16 | grep -L "0:0:c:4:b2:33" | wc -l
     0
```

From this data, and the information at http://standards.ieee.org/regauth/oui/oui.txt we know that the IDS/packetsniffer is between two Cisco devices.

2) **Detect was generated by:**
The detect was generated by Snort Intrusion Detection System version 2.0.0 (Build 72) with a rule set downloaded on August 31, 2003. The detect was captured in binary format. The command used was "snort -c snort.conf -r 2002.10.16 -l log –b". Description of Snort switches are listed below.

-c Use Rules File
-r Read and process tcpdump file
-l Log to directory
-b Log packets in tcpdump format

Listed below is the signature that triggered the Snort alerts.

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD-TRAFFIC tcp port 0 traffic"; classtype:misc-activity; sid:524; rev:6;)
```

Signature triggered based on the following:
1. Tcp traffic
2. External traffic flowing to internal
3. Destination port 0

3) **Probability the source address was spoofed:**
There is a very low probability that the source address was spoofed. The attacker in this case seemed to be performing reconnaissance, and is trying to elicit a response from the target host. If the attacker spoofed the

source IP address he would never receive a response because the response would travel back to the spoofed IP. The attacker would need a packet-capturing box on the wire in order to analyze the response from the target hosts.

4) **Description of attack:**
This detect is not necessarily an attack, the attacker may be performing reconnaissance for future use. The attacker is sending TCP SYN packets in groups of 4 (consisting of an initial attempt with three retries) to three different hosts on the 170.129.X.X network with a destination port of 0 which is reserved according to RFC 1700 http://www.ietf.org/rfc/rfc1700.txt?number=1700. The TTL values at 46 remained constant while the attacker scanned all three hosts. The frequency that the packets were sent would suggest a doubling of back off times. The 2nd packet is sent 3 seconds after the first, the third packet is sent 6 seconds after the second and the fourth packet is send 12 seconds after the third as seen below in example 1 (actual attack trace). The retransmits could be due to the target host not responding or the attacker is using some type of script that sends the packets out even if the target hosts responds. Based on the Tcpdump trace Example 1 we can speculate that the attacker seems to be using a Linux 2.4 kernel host. This was accomplished by analyzing the Window size, MSS (Maximum Segment Size), DF (Don't fragment), SackOK (Selective Acknowledgement OK), Window Scaling and NOP fields and comparing them to the default values of a Linux kernel 2.4 shown in example 2. They are as follows: 1) Window size - 5840 2) MSS – set to 1460 3) DF – is set 4) SackOK – is set 5) wscale – 0 6) NOP Flag – is set. In addition to these fields, a Linux box with a 2.4 kernel does use a doubling of backoff times. Example 2 is a Linux box with a 2.4 kernel and is shown attempting to connect to host 10.1.1.10 on the telnet port by using the telnet 10.1.1.10 command. Highlighted in red displays the doubling of backoff times. The host 10.1.1.10 was configured to not respond to the syns sent to port 23. Other information to note is the IP ID field remained constant at 0. The default Linux 2.4 TCP\IP stack normally increments by 1. If this was a Linux box with a 2.4 kernel, the attacker could have manipulated this field.

Listed below in example 1 is a snippet of the connections from the attackers host which displays the evidence of TTL, DF, SYN, IP ID, timestamp, src host/port, and dst host/port mentioned above. The command used was windump -nnvv -r \snort\etc\log\snort.log "src host 211.47.255.20" Description of Windump switches are listed below

-nn Don't convert addresses (host addresses, port numbers, etc.) to names
-vv Displays TTL, identification, total length and options
-r Read packets from file

"src host 211.47.255.20" specifies what source IP to search for.

Example 1 – Actually attack trace

```
02:36:46.156507 211.47.255.20.47040 > 170.129.118.169.0: S [tcp sum ok]
415240109:415240109(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF) (ttl 46,
id 0, len 52)
02:36:49.176507 211.47.255.20.47040 > 170.129.118.169.0: S [tcp sum ok]
415240109:415240109(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF) (ttl 46,
id 0, len 52)
02:36:55.156507 211.47.255.20.47040 > 170.129.118.169.0: S [tcp sum ok]
415240109:415240109(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF) (ttl 46,
id 0, len 52)
02:37:08.116507 211.47.255.20.47040 > 170.129.118.169.0: S [tcp sum ok]
415240109:415240109(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF) (ttl 46,
id 0, len 52)
03:54:53.826507 211.47.255.20.35752 > 170.129.44.181.0: S [tcp sum ok]
1051381955:1051381955(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF) (ttl 46,
id 0, len 52)
03:54:56.806507 211.47.255.20.35752 > 170.129.44.181.0: S [tcp sum ok]
1051381955:1051381955(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF) (ttl 46,
id 0, len 52)
03:55:02.826507 211.47.255.20.35752 > 170.129.44.181.0: S [tcp sum ok]
1051381955:1051381955(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF) (ttl 46,
id 0, len 52)
03:55:14.996507 211.47.255.20.35752 > 170.129.44.181.0: S [tcp sum ok]
1051381955:1051381955(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF) (ttl 46,
id 0, len 52)
```

Example 2 – Analyst Telnet Session trace

```
02:14:57.325511 10.1.1.165.34381 > 10.1.1.10.23: S [tcp sum ok]
1685502267:1685502267(0) win 5840 <mss 1460,sackOK,timestamp800922 0,nop,wscale 0>
(DF) [tos 0x10]  (ttl 64, id 33185, len 60)
02:15:00.318075 10.1.1.165.34381 > 10.1.1.10.23: S [tcp sum ok]
1685502267:1685502267(0) win 5840 <mss 1460,sackOK,timestamp801222 0,nop,wscale 0>
(DF) [tos 0x10]  (ttl 64, id 33186, len 60)
02:15:06.318075 10.1.1.165.34381 > 10.1.1.10.23: S [tcp sum ok]
1685502267:1685502267(0) win 5840 <mss 1460,sackOK,timestamp801822 0,nop,wscale 0>
(DF) [tos 0x10]  (ttl 64, id 33187, len 60)
02:15:18.318065 10.1.1.165.34381 > 10.1.1.10.23: S [tcp sum ok]
1685502267:1685502267(0) win 5840 <mss 1460,sackOK,timestamp803022 0,nop,wscale 0>
(DF) [tos 0x10]  (ttl 64, id 33188, len 60)
```

5) Attack mechanism:

The attacker in this case seemed to be performing reconnaissance, and is trying to elicit a response from the target host. If the attacker is able to retrieve a response from the target host, the response could then be analyzed to determine the target operating system. For example, If the attacker is able to determine the target host operating system, they could find specific vulnerabilities associated with the operating system to use as an attack. Another alternative as to why the attacker is sending these packets is perhaps there is a backdoor he is using that has yet to be uncovered. Fortunately after running the following command, windump -nnvv -r \snort\etc\2002.10.16 "dst host 211.47.255.20" we see that the targeted hosts did not respond to the attacker's packets.

6) Correlations:

This detect has been documented by many prior GIAC certified analysts such as Susan Kovacevich and Joe Bowling. Susan's practical can be

found at http://www.giac.org/practical/GCIA/Susan_Kovacevich_GCIA.pdf Joe's practical is not posted yet, although his detect was submitted to incidents.org. Both of their detects were coming from the 211.47.255.X network. Dshield had no record of attacks from 211.47.255.20. Listed below is WHOIS query on APNIC.

WHOIS query on source IP: http://www.apnic.org/search/index.html

```
inetnum:        211.42.0.0 - 211.51.255.255
netname:        KRNIC-KR
descr:          KRNIC
descr:          Korea Network Information Center
country:        KR
admin-c:        HM127-AP
tech-c:         HM127-AP
remarks:        ****************************************
remarks:        KRNIC is the National Internet Registry
remarks:        in Korea under APNIC. If you would like to
remarks:        find assignment information in detail
remarks:        please refer to the KRNIC Whois DB
remarks:        http://whois.nic.or.kr/english/index.html
remarks:        ****************************************
mnt-by:         APNIC-HM
mnt-lower:      MNT-KRNIC-AP
changed:        hostmaster@apnic.net 19991118
changed:        hostmaster@apnic.net 20010606
status:         ALLOCATED PORTABLE
source:         APNIC
person:         Host Master
address:        11F, KTF B/D, 1321-11, Seocho2-Dong, Seocho-Gu,
address:        Seoul, Korea, 137-857
country:        KR
phone:          +82-2-2186-4500
fax-no:         +82-2-2186-4496
e-mail:         hostmaster@nic.or.kr
nic-hdl:        HM127-AP
mnt-by:         MNT-KRNIC-AP
changed:        hostmaster@nic.or.kr 20020507
source:         APNIC
```

Other resources that can be used for information regarding port 0 and fingerprinting at:

http://compnetworking.about.com/library/ports/blports_0.htm
http://www.securiteam.com/securityreviews/5XP0Q2AAKS.html

7) Evidence of targeting

There is no evidence of active targeting. Using the detects from Susan's paper and Joe's detect I concluded that the scans on our three hosts were not targeted but just a portion of a larger scan.

8) Severity

Criticality - Because I am unsure what the attacker is attempting to accomplish by sending these packets a criticality level of 3 is selected. Fortunately based on our logs, there were no responses by the victim machines. However if responses were omitted from the logs the attacker could have gained knowledge of victim machines, which could lead to future attacks. **Criticality = 3**

Lethality - Since I concluded the detects were reconnaissance missions the lethality is low. **Lethality = 2**

System Countermeasures - System countermeasures were high. The victim machines did not respond. The host could have been using iptables or a software firewall to prevent the packets from rising through the TCP/IP stack. Since we cannot be sure of what happened system countermeasures are set to 4.
System Countermeasures = 4

Network Countermeasures – Network countermeasures were high. The victim machines did not respond. Perhaps a firewall or a packet filter prevented the packets from reaching the victim hosts. Since we cannot be sure of what happened network countermeasures are set to 4.
Network Countermeasures = 4

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

(3+2) – (4+4) = -3

9) Defensive recommendation

Monitor the suspicious host that is scanning our network. Perhaps they will continue to probe our IP address space and scan for available services. At border router and firewalls we can configure (ingress filtering) access-lists to deny attackers IP into our network. Deny any reserved IP addresses from entering our network. In addition the network administrator should only permit access to services that are needed. The proper egress filtering will also be of use just incase there is a misconfiguration in the ingress filtering. Be sure virus definitions, IDS signatures and firewall firmware are updated. Patch all systems.

10) Multiple choice test question

If you send a SYN packet to a destination port of 0 on a Windows or Unix box , what response would you receive ? ** Target host not listening on port 0.

a) You would receive a Syn-Ack.
b) You would receive a RST-ACK
c) You would receive a SYN-ACK

d) You would receive a PSH –ACK

Answer – The answer is b), the windows or unix box will respond with a RST-ACK. I used hping2 with the command hping2 –V –S –p 0 192.168.1.2 and hping2 –V –S –p 0 10.1.1.2 to determine the answer. Tcpdump used to capture stimulus and response. The command use was tcpdump -nnvv. Description of hping2 switches are listed below

-V verbose mode
-S SYN
-p destination port

Validation

Response from a windows 2000 box

```
02:37:34.365331 192.168.1.1.2909 > 192.168.1.2.0: S [tcp sum ok] 1960510289:1960510289(0)
win 512 (ttl 64, id 200, len 40)
02:37:34.365548 192.168.1.2.0 > 192.168.1.1.2909: R [tcp sum ok] 0:0(0) ack 1960510290
win 0 (ttl 128, id 65190, len 40)
```

Response from a Unix Box

```
02:56:15.641783 10.1.1.1.1703 > 10.1.1.2.0: S [tcp sum ok] 2091797554:2091797554(0) win
512 (ttl 64, id 516, len 40)
02:56:15.641989 10.1.1.2.0 > 10.1.1.1.1703: R [tcp sum ok] 0:0(0) ack 2091797555 win 0
(DF) (ttl 64, id 0, len 40)
```

Detect #2

```
[**] [111:13:1] (spp_stream4) STEALTH ACTIVITY (SYN FIN scan) detection
[**]
10/19-14:39:27.158103 202.54.138.61:21 -> my.net.1.2:21
TCP TTL:22 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x540D4030  Ack: 0x1C601FCE  Win: 0x404  TcpLen: 20

[**] [111:13:1] (spp_stream4) STEALTH ACTIVITY (SYN FIN scan) detection
[**]
10/19-14:39:27.178539 202.54.138.61:21 -> my.net.1.3:21
TCP TTL:22 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x540D4030  Ack: 0x1C601FCE  Win: 0x404  TcpLen: 20

[**] [111:13:1] (spp_stream4) STEALTH ACTIVITY (SYN FIN scan) detection
[**]
10/19-14:39:27.193588 202.54.138.61:21 -> my.net.1.4:21
TCP TTL:22 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x540D4030  Ack: 0x1C601FCE  Win: 0x404  TcpLen: 20
```

1) Source of Trace:

The source of the trace was from my network that includes my.net.1.2, my.net.1.3 and my.net.1.4. The network topology looks like this: cable modem → Router → → Firewall → internal network.

|
IDS

There are no services offered by my hosts that are publicly accessible. The stateful firewall is configured to allow only traffic outbound on ports 80, 443, 25, 110 and 53. The only traffic inbound should be traffic that is a response to my.net.1.2, my.net.1.3 and my.net.1.4 stimulus. My.net.1.2, my.net.1.3 and my.net.1.4 are the only hosts on the network.

2) Detect was generated by:

The detect was generated by Snort Intrusion Detection System version 2.0.0 (Build 72) with a rule set downloaded on August 31, 2003. The detect was captured in binary format.

The alert was generated by the snort Portscan Preprocessor Stream4. This a description of the Snort Portscan preprocessor from snort.org. "What the Snort Portscan Preprocessor does ….. Log the start and end of portscans from a single source IP to the standard logging facility."[12] "A portscan is also defined as a single stealth scan packet, such as NULL, FIN, SYNFIN, XMAS, etc"[13]
Listed below is a signature that can trigger similar Snort alerts.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN SYN
FIN";flags:SF; reference:arachnids,198; classtype:attempted-
recon; sid:624; rev:1;)
```

The Signature based alert would trigger based on the following:

1. Tcp traffic
2. External traffic flowing to internal
3. SYN and FIN flags set

3) Probability the source address was spoofed:

Although this detect could have been easily spoofed because it does not have to be part of an established session. I believe there is a low probability that the source address was spoofed being that the attacker in this case seemed to be performing reconnaissance, and is trying to elicit a response from the target hosts. If the attacker spoofed the source IP address he would never receive a response because the response would

[12] http://www.snort.org/docs/snort_manual/node17.html#SECTION00381100000000000000
[13] http://www.snort.org/docs/snort_manual/node17.html#SECTION00381100000000000000

travel back to the spoofed IP. The attacker would need a packet-capturing box on the wire in order to analyze the response from the target hosts.

4) Description of attack:

This detect is not an attack, but more of an information gathering attempt by the attacker. However these types of scans are often an alert to a preemptive attack. There are other situations where SYN-FIN scans are useful to an attacker such as gathering information regarding firewall ports being open. Another would be if the attacker is trying to elicit a response from the target host to determine the operating system. Or perhaps the attacker is trying to determine what ports are opened or closed. Although this would be a very "noisy" way to accomplish this. The attacker in actual trace is sending packets from his host source port 21 to destination host port 21 with the SYN and FIN flag combination set, window size of 1028, IP ID of 39426, and type of service 0x00.

5) Attack mechanism:

Listed below are the three alerts triggered by snort displayed by windump –Xnnvv –r snort.log command. Description of windump switches are listed below

-X When printing hex, print ASCII
-nn Don't convert addresses (host addresses, port numbers,) to names
-vv Displays TTL, identification, total length and options
-r Read packets from file

```
14:39:27.158103 202.54.138.61.21 > my.net.1.2.21: SF [tcp sum ok]
1410154544:1410154544(0) win 1028 (ttl 22, id 39426, len 40)
0x0000      4500 0028 9a02 0000 1606 2a80 ca36 8a3d     E..(......*..6.=
0x0010      xxxx 0102 0015 0015 540d 4030 1c60 1fce     ........T.@0.`..
0x0020      5003 0404 faf9 0000 0000 0000 0000          P.............
14:39:27.178539 202.54.138.61.21 > my.net.1.3.21: SF [tcp sum ok]
1410154544:1410154544(0) win 1028 (ttl 22, id 39426, len 40)
0x0000      4500 0028 9a02 0000 1606 2a7f ca36 8a3d     E..(......*..6.=
0x0010      xxxx 0103 0015 0015 540d 4030 1c60 1fce     ........T.@0.`..
0x0020      5003 0404 faf8 0000 0000 0000 0000          P.............
14:39:27.193588 202.54.138.61.21 > my.net.1.4.21: SF [tcp sum ok]
1410154544:1410154544(0) win 1028 (ttl 22, id 39426, len 40)
0x0000      4500 0028 9a02 0000 1606 2a7e ca36 8a3d     E..(......*~.6.=
0x0010      xxxx 0104 0015 0015 540d 4030 1c60 1fce     ........T.@0.`..
0x0020      5003 0404 faf7 0000 0000 0000 0000          P.............
```

The above scan was probably initiated by the use of the Synscan tool, whether by a person or worm (Ramen[14]). Although there are other tools and worms that use the "Synscan engine". Synscan has these attributes [15], which are consistent with the trace above.

- TCP source port 21, destination port 21 (Shown above in blue)
- Type of service 0 (Shown in red)
- IP identification number 39426 (Shown in green)
- SYN and FIN flags set (Shown in grey)

[14] http://vil.nai.com/vil/content/v_98975.htm
[15] http://www.securityfocus.com/infocus/1524

- TCP window size 1028 (Shown in purple)

Synscan is authored by psych0id[16], and Joe Stewart seemingly discovered its origins. This is a link where the tool can be downloaded http://www.psychoid.lam3rz.de/synscan.html. According to Donald Smith[17] Synscan is, "…..actually two vulnerability scanners and several support tools."[18] Joe indicates after the Synscan tool runs a scan it sends a SYN-FIN to www.microsoft.de (IP address 212.184.80.190) destination port 80 from source port 31337[19]. Interestingly enough Joe also discovered that you could stop Synscan from process from listening if you spoof a SYN-ACK from www.microsoft.de source port 80 with the target being the Synscan host destination port 31337. The actual command using hping would be this, "hping -p 31337 -s 80 -k -S -A -a 212.184.80.190 ip.address.of.scanner."[20]

Because the actual scan has a destination port of 21 we need to look further into the Ramen worm. The reasoning is due to the Ramen worm uses Synscan to attempt find, "…possible target hosts" across the internet. Then the Ramen worm could exploit a vulnerability (CVE-2000-0573[21]) found in wu-ftpd version 2.6.0 and earlier[22].

Another note is the attack upon my network is coming from reflexive ports, source and destination port of 21. I believe it uses a non-ephemeral port because it may be more likely to bypass a simple packet filter.

Listed below is a reenactment of the above scan. Since I know that the hosts that were scanned are not offering service on port 21, the first reenactment displays the stimulus as well as a response from a host with port 21 closed. Notice the response from the target host. It responds with a RST ACK, which is considered normal. The tool used to craft packets was hping2. The following was the command that was used, hping2 –V –s 21 –p 21 –SF X.X.3.17.

-V verbose mode
-SF SYN FIN
-p destination port
-s source port
X.X.3.17 is the target host

[16] http://archives.neohapsis.com/archives/incidents/2000-11/0162.html
[17] http://www.giac.org/practical/donald_smith_gcia.doc
[18] http://www.giac.org/practical/donald_smith_gcia.doc
[19] http://archives.neohapsis.com/archives/incidents/2000-11/0162.html
[20] http://archives.neohapsis.com/archives/incidents/2001-01/0146.html
[21] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0573
[22] http://vil.nai.com/vil/content/v_98975.htm

```
10:43:15.137153 X.X.24.1.22 > X.X.3.17.21: SF [tcp sum ok] 1325367630:1325367630(0)
win 512 (ttl 64, id 42514, len 40)
0x0000      4500 0028 a612 0000 4006 a5aa xxxx 1801     E..(....@.......
0x0010      xxxx 0311 0016 0015 4eff 814e 6bff 0907     ........N..Nk...
0x0020      5003 0200 394f 0000 0000 0000 0000          P...9O........
10:43:15.137197 X.X.3.17.21 > X.X.24.1.22: R [tcp sum ok] 0:0(0) ack 1325367632 win 0
(ttl 128, id 23212, len 40)
0x0000      4500 0028 5aac 0000 8006 b110 xxxx 0311     E..(Z...........
0x0010      xxxx 1801 0015 0016 0000 0000 4eff 8150     ............N..P
0x0020      5014 0000 b042 0000                         P....B..
```

Listed below is another reenactment of the above scan, with port 21 open on the target host. Notice the response, the host responded with a SYN-ACK from a SYN-FIN. Normal response should have been RST ACK. The command and switches that were used are hping2 –V –s 21 –p 21 –SF X.X.3.17. The –V switch is used for verbose mode. The –s and –p defines the source and destination port respecitively. The –SF sets the SYN and FIN bits and the X.X.3.17 specifies target host. Netcat was used to open port 21 on target host. The command used was nc –l –p 21. The –l switch specifies the host to listen, and the –p tells the host what port to listen on.

```
10:43:14.141043 X.X.24.1.21 > X.X.3.17.21: SF [tcp sum ok] 1078716459:1078716459(0)
win 512 (ttl 64, id 30434, len 40)
0x0000      4500 0028 76e2 0000 4006 d4da xxxx 1801     E..(v...@.......
0x0010      xxxx 0311 0015 0015 404b e82b 31b7 9550     ........@K.+1..P
0x0020      5003 0200 8f25 0000 0000 0000 0000          P....%........
10:43:14.141104 X.X.3.17.21 > X.X.24.1.21: S [tcp sum ok] 524964813:524964813(0) ack
1078716460 win 64512 <mss 1260> (DF) (ttl 128, id 23211, len 44)
0x0000      4500 002c 5aab 4000 8006 710d xxxx 0311     E..,Z.@...q.....
0x0010      xxxx 1801 0015 0015 1f4a 53cd 404b e82c     .........JS.@K.,
0x0020      6012 fc00 d210 0000 0204 04ec               `...........
```

6) Correlations:

The IP address 202.54.138.61 had a profile at Dshield.org. [23]

Country:	IN
Contact E-mail:	abuse@vsnl.com
AS Number:	4755
Total Records against IP:	149
Number of targets:	30
Date Range:	2003-10-30 to 2003-11-16

Top Port hit by this source:

Port	Attacks	Start	End
23	1	2003-12-15	2003-12-15

[23] http://www.dshield.org/ipinfo.php?ip=202.54.138.61&Submit=Submit

WHOIS query on source IP address: http://www.dshield.org/ipinfo.php

```
inetnum:      202.54.128.0 - 202.54.255.255
netname:      VSNL-IN
country:      IN
descr:        Videsh Sanchar Nigam Ltd - India.
              Videsh Sanchar Bhawan, M.G. Road
              Fort, Bombay 400001
admin_c:      IA15-AP
tech_c:       VT43-AP
remarks:      Internet Service Provider
mnt_by:       APNIC-HM
changed:      hostmaster@apnic.net 20020204
status:       ALLOCATED PORTABLE
source:       APNIC
notify:
mnt_lower:    MAINT-VSNL-AP
rev_srv:
start:        3392569344
end:          3392602111
diff:         32767
person:       VSNL Tech
address:      10th Floor, 2 MG Road
              Fort Mumbai - 400001
              India
country:      IN
country:      IN
phone:        +91-22-2623620
fax_no:       +91-22-2653887
e_mail:       ip-tech@giasbm01.vsnl.net.in
nic_hdl:      VT43-AP
mnt_by:       MAINT-VSNL-AP
changed:      gpsingh@giasbm01.vsnl.net.in 20010605
source:       APNIC
remarks:
```

This detect has been well documented. You can find more information about the detect as well as the buffer overflow exploit at:

Snort.org http://www.snort.org/snort-db/sid.html?sid=624

WhiteHats.com
http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids198&view=event

SecurityFocus
http://www.securityfocus.com/infocus/1524

Donald Smith[24] has a great write up on versions of the Synscan tool. His practical has a table listing various version of Synscan along with the authors of these tools.

[24] http://www.giac.org/practical/donald_smith_gcia.doc

More on the RAMEN worm
http://vil.nai.com/vil/content/v_98975.htm

7) **Evidence of targeting**
Since the detect was from my network and there are no hosts offering service on port 21 I surmised that there was no active evidence of targeting. The scan on these three hosts were probably just a portion of a larger scan.

8) **Severity**

Criticality - The machines that were scanned do not offer service on port 21. In fact these machines sit behind a firewall with no public access. **Criticality = 2**

Lethality - The probes from this detect are not very lethal. **Lethality = 2**

System Countermeasures - System countermeasures were very high. The machine does not offer service on port 21. The three targeted hosts also use software firewalls.
System Countermeasures = 5

Network Countermeasures – The machines are situated behind a firewall with no public access.
Network Countermeasures = 5

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

(2+2) – (5+5) = -6

9) **Defensive recommendation**
Monitor the suspicious host that is scanning our network. Perhaps they will continue to probe our IP address space and scan for available services. At border router and firewalls configure access-lists to deny attackers IP into our network. We should also ensure that illegal flag combinations are not allowed. Deny any reserved IP addresses from entering our network. Be sure virus definitions, IDS signatures and firewall firmware are updated. Patch all systems especially if the hosts that were probed used vulnerable version of wu-ftpd.

10) **Multiple choice test question**
If you send a SYN-FIN to an open port on a Windows or Unix box, what response would you receive ?

a) A Windows or Unix box will respond with a Syn-Ack.
b) You would receive a RST-ACK
c) You would receive a SYN-ACK
d) You would receive a PSH -ACK

Answer – The answer is a), the windows or unix box will respond with a syn-ack. If the port were closed we would expect a RST-ACK

Detect #3

```
[**] [1:1616:4] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
05/21-14:55:56.934488 203.155.239.38:2603 -> 78.37.243.118:53
UDP TTL:44 TOS:0x0 ID:52120 IpLen:20 DgmLen:58
Len: 30
[Xref => http://www.whitehats.com/info/IDS278][Xref =>
http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
05/21-15:15:13.664488 203.155.239.38:4942 -> 78.37.128.120:53
UDP TTL:44 TOS:0x0 ID:57708 IpLen:20 DgmLen:58
Len: 30
[Xref => http://www.whitehats.com/info/IDS278][Xref =>
http://cgi.nessus.org/plugins/dump.php3?id=10028]

[**] [1:1616:4] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
05/21-15:44:41.554488 203.155.239.38:3960 -> 78.37.90.40:53
UDP TTL:44 TOS:0x0 ID:35592 IpLen:20 DgmLen:58
Len: 30
[Xref => http://www.whitehats.com/info/IDS278][Xref =>
http://cgi.nessus.org/plugins/dump.php3?id=10028]
```

1) **Source of Trace:**
The source of the trace was from http://www.incidents.org/logs/Raw/2002.4.22. Listed below are Tcpdump outputs which gives a better understanding of the topology of the Ethernet network. Description of Tcpdump switches are listed below
-e Print the link-level header on each dump line.
-n Don't convert addresses (host addresses, port numbers,) to names
-r Read packets from file

Tcpdump –ne –r ./2002.4.22 outputted the MAC address shown below in red.

14:00:09.014488 0:0:c:4:b2:33 0:3:e3:d9:26:c0 0800 2974: 78.37.212.28.61469 > 64.154.80.50.80: P 0:2920(2920) ack 2921 win 8760 [tos 0x10]

After discovering both MAC addresses above, I ran the following tcpdump, grep and wc commands to determine if all of the packets were traversing between these two devices. The –L switch used with grep finds " --files-without-match" and the –l switch used with wc would output the count.

```
[root@ID tmp]# tcpdump -ne -r 2002.4.22 | grep -L "0:3:e3:d9:26:c0" | wc -l
  0
[root@ID tmp]# tcpdump -ne -r 2002.4.22 | grep -L "0:0:c:4:b2:33" | wc -l
  0
```

From this data, and the information at http://standards.ieee.org/regauth/oui/oui.txt we know that the IDS/packetsniffer is between two Cisco devices.

2) Detect was generated by:

The detect was generated by Snort Intrusion Detection System version 2.0.0 (Build 72) with a rule set downloaded on August 31, 2003. The detect was captured in binary format. The alert was generated because UDP packets were sent from an external network destined for our internal network over UDP destination port 53. Along with a content of |07| version and |04| bind with an offset of 12.

Listed below is the signature that triggered the Snort alerts.[25]

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named
version attempt"; content:"|07|version"; nocase; offset:12;
content:"|04|bind"; nocase; offset: 12; reference:nessus,10028;
reference:arachnids,278; classtype:attempted-recon; sid:1616;
rev:4;)
```

Signature triggered based on the following:

1) UDP traffic
2) External traffic flowing to internal
3) Destination port 53
4) Offset of 12 bytes
5) Hex 07 and text version
6) Hex 04 and text bind
7) Not case sensitive

3) Probability the source address was spoofed:

Even though the source IP address can be easily spoofed because there is no three-way handshake due to UDP connection. I believe there is a very low probability that the source address was spoofed. The attacker in

[25] http://www.snort.org/snort-db/sid.html?sid=1616

this case seemed to be performing reconnaissance, and is trying to determine if the victim is running Vulnerable version of Bind on their nameserver.

4) **Description of attack:**
The attacker in this case is scanning for vulnerable versions of Bind. This probe is not an attack, but more of a reconnaissance mission. In the event of finding a vulnerable machine, the attacker presumably will launch an attack.

5) **Attack mechanism:**
Listed below are the three alerts triggered by snort displayed by tcpdump using the –Xnnvv switches.

```
14:55:56.934488 203.155.239.38.2603 > 78.37.243.118.53:  [bad udp cksum b7ba!]
4660 [b2&3=0x80] TXT CHAOS)? version.bind. [|domain] (ttl 44, id 52120, len 58,
bad cksum c05!)
0x0000   4500 003a cb98 0000 2c11 0c05 cb9b ef26     E..:....,......&
0x0010   4e25 f376 0a2b 0035 0026 09d3 1234 0080     N%.v.+.5.&...4..
0x0020   0001 0000 0000 0000 0776 6572 7369 6f6e     .........version
0x0030   0462 696e 6400 0010 0003                    .bind.....
15:15:13.664488 203.155.239.38.4942 > 78.37.128.120.53:  [bad udp cksum b8b8!]
4660 [b2&3=0x80] TXT CHAOS)? version.bind. [|domain] (ttl 44, id 57708, len 58,
bad cksum 6b2e!)
0x0000   4500 003a e16c 0000 2c11 6b2e cb9b ef26     E..:.l..,.k....&
0x0010   4e25 8078 134e 0035 0026 75ad 1234 0080     N%.x.N.5.&u..4..
0x0020   0001 0000 0000 0000 0776 6572 7369 6f6e     .........version
0x0030   0462 696e 6400 0010 0003                    .bind.....
15:44:41.554488 203.155.239.38.3960 > 78.37.90.40.53:  [bad udp cksum b8b8!] 4660
[b2&3=0x80] TXT CHAOS)? version.bind. [|domain] (ttl 44, id 35592, len 58, bad
cksum e7e2!)
0x0000   4500 003a 8b08 0000 2c11 e7e2 cb9b ef26     E..:....,......&
0x0010   4e25 5a28 0f78 0035 0026 9fd3 1234 0080     N%Z(.x.5.&...4..
0x0020   0001 0000 0000 0000 0776 6572 7369 6f6e     .........version
0x0030   0462 696e 6400 0010 0003                    .bind.....
```

The attacker has sent the above payload to udp port 53 to above machines which will query the authors.bind chaos records if the machines are nameservers. If a vulnerable version of Bind is running on the nameserver, then the attacker could then exploit numerous vulnerabilities found in bind http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=version+bind (CVE-1999-0009, CVE-19990010, CAN-2002-0400, CVE-2001-0010, CVE-2001-0012, and many others) and could potentially run arbitrary code on the machine. The probe can be easily run utilizing the dig command. The following is an example, “dig txt chaos version.bind. @10.0.0.1” shown below.

```
[root@winxp root]# dig txt chaos version.bind. @10.0.0.1

; <<>> DiG 9.2.1 <<>> txt chaos version.bind. @10.0.0.1
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44051
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
```

```
;; QUESTION SECTION:
;version.bind.              CH     TXT

;; ANSWER SECTION:
version.bind.        0     CH     TXT     "9.2.1"

;; Query time: 32 msec
;; SERVER: 10.0.0.1#53(10.0.0.1)
;; WHEN: Wed Dec 10 15:48:32 2003
;; MSG SIZE  rcvd: 48
```

The command "dig txt chaos version.bind. @10.0.0.1" resulted in the DNS server responding with the version of bind (shown in red) 9.2.1. If an attacker discovers the version of bind, his next step is to find if the version is vulnerable.

6) Correlations:

WHOIS query on source IP: http://www.apnic.org/search/index.html

```
inetnum:        203.155.0.0 - 203.155.255.255
netname:        COMNET-TH
descr:          KSC Commercial Internet Co. Ltd.
descr:          2/4 Samaggi Insurance Tower 10th Fl.,
descr:          Viphavadee-Rangsit RD
descr:          Thungsonghong, Laksi
descr:          Bangkok 10210
country:        TH
admin-c:        JT183-AP
tech-c:         TOC1-AP
remarks:        service provider
remarks:        Delegate small blocks to /16 block
mnt-by:         APNIC-HM
mnt-lower:      KSC-ADMIN
changed:        admin@ns.ksc.co.th 19980825
changed:        hostmaster@apnic.net 20011016
changed:        hm-changed@apnic.net 20020830
status:         ALLOCATED PORTABLE
source:         APNIC
person:         Joost Th.A Doevelaar
nic-hdl:        JT183-AP
e-mail:         jdoevelaar@ksc.net
address:        KSC Commercial Internet Co.,Ltd.
address:        2/4 Samaggi Insurance Tower 10th Fl., Viphavadee-Rangsit
Rd.,
address:        Thungsonghong, Laksi
address:        Bangkok 10210
phone:          +66-2-9797777
fax-no:         +66-2-5885665
country:        TH
changed:        netadmin@ns.ksc.co.th 20030115
mnt-by:         KSC-ADMIN
source:         APNIC
person:         Technical Operation Center
```

```
address:        KSC Commercial Internet Co.,Ltd.
address:        Operation Department
address:        2/4 Samaggi Insurance Tower 10th Fl., Viphavadee-Rangsit
Rd.,
address:        Thungsonghong, Laksi
address:        Bangkok 10210
country:        TH
phone:          +66-2-9797777 ext. 8428
e-mail:         netadmin@ns.ksc.co.th
nic-hdl:        TOC1-AP
mnt-by:         KSC-ADMIN
changed:        admin@ns.ksc.co.th 20011012
```

Dshield had no records or profile against 203.155.239.38.

This detect has been well documented. You can find more information about the detect as well as the buffer overflow exploit at:

Whitehats: http://www.whitehats.com/info/IDS480

Security focus: http://www.securityfocus.com/bid/134/info/ (CVE-1999-0009)

7) **Evidence of targeting**
   With the limited amount of data, I surmised that there was active evidence of targeting because the attacker only selected three hosts over the network.
8) **Severity**

Criticality - Based on the probes by the attacker, we presume the attacker is looking for a DNS server. **Criticality = 5**

Lethality - The probes from this detect are not very lethal. However, they normally are good indicators of an attack exploiting known vulnerable versions of BIND. **Lethality = 5**

System Countermeasures - System countermeasures were very low. Since I am not sure if the boxes are nameservers running vulnerable versions of BIND. **System Countermeasures = 2**

Network Countermeasures – Since I am not sure if these were publicly available nameservers he gave network countermeasures a low score. **Network Countermeasures = 2**

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

(5+5) – (2+2) = 6

9) Defensive recommendation

Install the latest patches, and/or version upgrades. If this is only an internal network nameserver, block public access to it by using the proper ingress filtering on border routers and also using the proper access-list on firewalls. If the host is not a nameserver DISABLE the service named.

If the host is a nameserver, then run BIND as a non-root user. http://www.losurs.org/docs/howto/Chroot-BIND.html is a link on how to accomplish this task. As an alternative we could use the configuration options on the machine to disable the ability that allow users to query the version.bind chaos record. To perform this we would change the version option in the /etc/named.conf (Linux Red Hat 9.0) to respond with "No DNS here", shown below in red.

```
options {
        directory "/var/named";
        version "No DNS here";
        /*
         * If there is a firewall between you and nameservers you want
         * to talk to, you might need to uncomment the query-source
         * directive below.  Previous versions of BIND always asked
         * questions using port 53, but BIND 8.1 uses an unprivileged
         * port by default.
         */
        // query-source address * port 53;
```

The configuration change to the named.conf above will result in the following below when the "dig txt chaos version.bind @10.0.0.1" command is run again. Notice the response highlighted in blue, "No DNS here".

```
[root@winxp root]# dig txt chaos version.bind @10.0.0.1

; <<>> DiG 9.2.1 <<>> txt chaos version.bind @10.0.0.1
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61768
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;version.bind.                  CH      TXT

;; ANSWER SECTION:
version.bind.           0       CH      TXT     "No DNS here"

;; Query time: 26 msec
;; SERVER: 10.0.0.1#53(10.0.0.1)
;; WHEN: Thu Dec 11 10:13:50 2003
;; MSG SIZE  rcvd: 54
```

10) Multiple choice test question

Snort Signature

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named
version attempt"; content:"|07|version"; nocase; offset:12;
content:"|04|bind"; nocase; offset: 12; reference:nessus,10028;
reference:arachnids,278; classtype:attempted-recon; sid:1616;
rev:4;)
```

Based on the Snort signature (above) and the Tcpdump audit trail, found below. Would this event trigger an alert?

Tcpdump trace

```
15:44:41.554488 203.155.239.38.3960 > 78.37.90.40.53:  [bad udp cksum b8b8!] 4660
[b2&3=0x80] [|domain] (ttl 44, id 35592, len 58, bad cksum e7e2!)
0x0000   4500 003a 8b08 0000 2c11 e7e2 cb9b ef26
0x0010   4e25 5a28 0f78 0035 0026 9fd3 1234 0080
0x0020   0001 0000 0000 0700 0076 6572 7369 6f6e
0x0030   0462 696e 6400 0010 0003
```

a) Yes, this will trigger an alert based on signature provided.
b) No, this will not trigger an alert because the |07version| is not found offset 12
c) No, this will not trigger an alert because of the bad udp cksum
d) No, this will not trigger an alert based on the ttl and id values.

Answer: b, the value of |07| is not found at an offset of 12. Moving the |07| was done just for signature analysis purposes only. The actual dump is not legitimate.

Questions From Intrusion list

**based on questions received from intrusion list.
The detect was posted on November 4, 2003 and can be found at
http://cert.uni-stuttgart.de/archive/intrusions/2003/11/msg00021.html

Question1 : Please explain why would someone want to run a buffer over flow attack. Maybe even a short description of what a buffer overflow attack is…

Answer: A buffer overflow is caused when a process/program tries to store more data in a buffer than it was configured to hold. The extra data can overflow into adjacent memories or buffers. At this point the overflowed data contains code, which causes malicious activities.[26] Attackers want to run a buffer overflow attack because a successful attack will result in the compromised host executing their specific action.

26 http://www.iss.net/security_center/advice/Underground/Hacking/Methods/Technical/buffer_overflow/default.htm

Question2 : Do you think it could be just a very wide slow and stealth recon attempt that covered multiple network ranges?
Answer: Yes it could have been a "very wide slow and stealth recon attempt" but the recon attempt was still targeting these particular hosts. Again, we would need more log information to convincingly determine if the recon attempt was or was not "actively targeted".

Question3 : Please explain how this is accomplished? (Disable query to version.bind)
Answer: The answer is now included in the Defensive Recommendation, Section 9 above.

Part III Analyze This

"Analyze This" Table of contents:

*links to students practicals are found in References section

Executive Summary

The analysis was conducted to give the University an idea of compromised internal hosts, correlation of internal hosts to external hosts and internal to internal hosts. The University will be provided with the following

- Description/correlation of events
- Insight to compromised machines
- Defensive recommendations.

Throughout the analysis process, evidence of **compromised internal hosts** was discovered. The University should apply the appropriate remediation beginning with the enclosed defensive recommendations.

Files

The following alert, OOS and scan files with a date range of October 19th to October 23rd 2003 were analyzed.

Alert Files	OOS files	Scan Files
alert.031019.gz	OOS_Report_2003_10_19.gz	scan.031019.gz
alert.031020.gz	OOS_Report_2003_10_20.gz	scan.031020.gz
alert.031021.gz	OOS_Report_2003_10_21.gz	scan.031021.gz
alert.031022.gz	OOS_Report_2003_10_22.gz	scan.031022.gz
alert.031023.gz	OOS_Report_2003_10_23.gz	scan.031023.gz

Top Ten Events

I have included the top ten events by frequency. Description are taken from www.whitehats.com unless specified, please visit the site for more information.

Count	Alert
199212	SMB Name Wildcard
118308	PORTSCAN DETECTED
28546	SMB C access
15606	MY.NET.30.4 activity
11563	EXPLOIT x86 NOOP
7131	connect to 515 from inside
5730	MY.NET.30.3 activity
4518	TCP SRC and DST outside network
3266	External RPC call
3172	High port 65535 tcp - possible Red Worm - traffic
2009	Possible trojan server activity

SMB Name Wildcard

Description

This event indicates a standard netbios name table retrieval query. Windows machines often exchange these queries as a part of the filesharing protocol to determine NetBIOS names when only IP addresses are known. An attacker could use this same query to extract useful information such as workstation name, domain, and users who are currently logged in.

"…Netbios "nbtstat" frames, which will elicit a node status response from Netbios and SAMBA clients. This response contains a listing of any Netbios names known to that node."[27]

The following signature is the Snort rule that could have triggered theses events.

```
alert UDP any any -> any 137 (msg: "SMB Name Wildcard"; content:
"CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA|00 00|";)
```

There are several trojans (Chode, Qaz and Msinit) that can run on port 137 but only Msinit uses UDP. In addition, some vulnerabilities can be found on port 137 such as CVE-2000-0673, CVE-1999-0288, CVE-2000-0347, CVE-2001-1162 and CVE-1999-0810. This signature would be of great concern if these packets were generated from a source host outside, but since these packets were from the inside with a destination of outside it is most likely normal. As stated above the queries could be the nbtstat command. If this traffic is normal, the University should consider modifying their egress filtering which could minimize the amounts of false positives.

Top five destination IPs

1268	198.62.205.6
1251	151.197.115.143
1209	193.114.70.169
878	199.181.134.74
710	169.254.45.176

Top five source IPs

115637	MY.NET.80.51
72075	MY.NET.150.133
3100	MY.NET.29.2
1291	MY.NET.84.224
474	MY.NET.150.198

Sample alert trace:

```
10/19-03:06:21.289492  [**] SMB Name Wildcard [**] MY.NET.150.133:137 -> 217.98.105.230:137
10/19-03:06:30.123604  [**] SMB Name Wildcard [**] MY.NET.150.133:3117 -> 64.116.216.15:137
```

[27] http://www.sans.org/resources/idfaq/port_137.php

10/19-03:06:33.725166 [**] SMB Name Wildcard [**] MY.NET.150.133:3117 -> 202.52.206.28:137
10/19-02:57:57.073399 [**] SMB Name Wildcard [**] MY.NET.150.133:3117 -> 24.91.30.37:137
10/19-02:58:00.674987 [**] SMB Name Wildcard [**] MY.NET.150.133:3117 -> 202.88.38.182:137
10/19-02:58:00.675067 [**] SMB Name Wildcard [**] MY.NET.150.133:3117 -> 24.49.66.15:137
10/19-02:58:00.675095 [**] SMB Name Wildcard [**] MY.NET.150.133:3117 -> 80.133.197.88:137
10/19-02:58:06.077397 [**] SMB Name Wildcard [**] MY.NET.150.133:3117 -> 4.8.96.218:137

The top source address for this alert MY.NET.80.51 has had SMB C access from two external addresses shown below. The concern about the SMB C access alert is that if the alerts triggered based upon Snort alert SID 533 (Signature shown in SMB C access section below) than these alerts were part of an established session. MY.NET.80.51 should be investigated. There were no OOS or SCAN packets originating from or destined to MY.NET.80.51.

10/23-10:08:03.131388 [**] SMB C access [**] 193.252.212.236:10672 -> MY.NET.80.51:139
10/23-10:16:52.073631 [**] SMB C access [**] 218.85.43.186:1303 -> MY.NET.80.51:139

However, looking closely at other alerts generated by 193.252.212.236 from the SMB C access alert above, we find that this host has had SMB C access on 39 other MY.NET.X.X hosts (sample shown below). All of the SMB C access to the MY.NET hosts was accomplished within 4 minutes. The University should investigate these particular 39 hosts to determine for possible compromise.

10/23-10:05:56.239598 [**] SMB C access [**] 193.252.212.236:10630 -> MY.NET.84.130:139?
10/23-10:08:26.948004 [**] SMB C access [**] 193.252.212.236:13019 -> MY.NET.80.209:139
10/23-10:08:27.874578 [**] SMB C access [**] 193.252.212.236:10685 -> MY.NET.80.215:139
10/23-10:08:29.996727 [**] SMB C access [**] 193.252.212.236:10688 -> MY.NET.80.228:139
10/23-10:08:30.145746 [**] SMB C access [**] 193.252.212.236:13022 -> MY.NET.80.229:139
10/23-10:08:30.716769 [**] SMB C access [**] 193.252.212.236:10689 -> MY.NET.80.232:139
10/23-10:08:30.972579 [**] SMB C access [**] 193.252.212.236:10690 -> MY.NET.80.233:139
10/23-10:08:30.988702 [**] SMB C access [**] 193.252.212.236:13023 -> MY.NET.80.234:139
10/23-10:08:37.526664 [**] SMB C access [**] 193.252.212.236:12974 -> MY.NET.81.107:139
10/23-10:08:50.663233 [**] SMB C access [**] 193.252.212.236:10683 -> MY.NET.80.165:139
10/23-10:09:30.205588 [**] SMB C access [**] 193.252.212.236:10686 -> MY.NET.80.225:139^

? Designates first alert
^ Designates last alert

The top destination host 198.62.205.6 proved to be of some interest because this host on October 22 through the 23 rapidly scanned (TCP and UDP) the 130.85.X.X network. Sorting through the Scan files discovered this (sample shown below). There were no OOS packets or other alerts originating from or destined to 192.62.205.6. This host should be possibly blocked at the gateway.

Oct 22 18:57:55 198.62.205.6:1773 -> 130.85.17.34:445 SYN ******S*?
Oct 22 18:57:55 198.62.205.6:1768 -> 130.85.17.34:139 SYN ******S*
Oct 22 18:57:55 198.62.205.6:137 -> 130.85.17.34:137 UDP
Oct 22 18:57:54 198.62.205.6:1757 -> 130.85.18.46:445 SYN ******S*
Oct 23 06:11:56 198.62.205.6:2725 -> 130.85.163.45:445 SYN ******S*
Oct 23 06:11:56 198.62.205.6:2726 -> 130.85.163.45:139 SYN ******S*
Oct 23 06:11:57 198.62.205.6:137 -> 130.85.163.45:137 UDP

Oct 23 06:11:58 198.62.205.6:2751 -> 130.85.163.45:139 SYN ******S*
Oct 23 06:11:58 198.62.205.6:137 -> 130.85.163.45:137 UDP
Oct 23 06:11:59 198.62.205.6:2750 -> 130.85.163.45:445 SYN ******S*^

? Designates first alert
^ Designates last alert

The address 169.254.45.176 was the fifth top destination for the SMB Name Wildcard alert is interesting because addresses that fall into the 169.254.0.0 – 169.254.255.255 range are commonly found when a Windows 2000 host is unable to find a DHCP server. "A Windows 2000-based DHCP client may lose connectivity to local network resources if it is unable to reach a DHCP server at startup. Windows 2000 behaves differently than does previous versions of Windows when it is unable to find a DHCP server. If APIPA was used, the IP address is from the APIPA Class B range of 169.254.0.0 to 169.254.255.255.The Windows 2000-based DHCP client may use Automatic Private IP Addressing (APIPA) for addressing if it is unable to reach a DHCP server and is also unable to reach its default gateway".[28] Perhaps there are misconfigured Windows 2000 hosts that require static addresses instead of DHCP addresses. Therefore these hosts are using the 169.254.X.X addresses and sending these packets on the network. Another explanation could be that these hosts are unable to connect to the DHCP server at the time. Either way these hosts require attention/reconfiguration.

Correlation

Bill Young, Marilyn Morris, Martin Kinwan

PORTSCAN DETECTED

This alert was generated from the Snort Portscan Preprocessor and not by a Snort Rule. The Portscan Preprocessor logs the start and end of a Portscan from an IP address. "A portscan is defined as TCP connection attempts to more than P ports in T seconds or UDP packets sent to more than P ports in T seconds." [29] This type of alert is not necessarily an attack, but it could be a precursor to an attack. This alert is a concern because most of the alerts were generated from internal sources. This particular alert does not give destination address or port number. The sample alert traces below displays the amount of data that is given. With the top ten source IPs coming from the inside, the University should investigate MY.NET.163.107, MY.NET.84.194, MY.NET.163.249, MY.NET.111.72, MY.NET.70.154, MY.NET.73.94, MY.NET.80.149, MY.NET.1.3, MY.NET.1.5 and MY.NET.151.72 as well as the other internal MY.NET source hosts for this event for possible misuse/abuse or compromise.

[28] http://support.microsoft.com/default.aspx?scid=kb;en-us;255836&Product=win2000
[29] http://www.snort.org/docs/writing_rules/chap2.html#tth_sEc2.4.2

Top Ten Source IPs

29572 MY.NET.163.107
27221 MY.NET.84.194
15237 MY.NET.163.249
 5312 MY.NET.111.72
 4506 MY.NET.70.154
 4222 MY.NET.73.94
 1635 MY.NET.80.149
 1312 MY.NET.1.3
 1066 MY.NET.1.5
  871 MY.NET.151.72

Sample alert trace:

10/19-00:17:41.994288 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.163.107 (THRESHOLD 12 connections exceeded in 0 seconds) [**]
10/19-00:17:49.891394 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.84.194 (THRESHOLD 12 connections exceeded in 0 seconds) [**]
10/19-00:18:04.350356 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.73.94 (THRESHOLD 12 connections exceeded in 0 seconds) [**]
10/19-00:18:07.385741 [**] spp_portscan: PORTSCAN DETECTED from 213.23.46.99 (STEALTH) [**]
10/19-00:18:15.654150 [**] spp_portscan: PORTSCAN DETECTED from 195.208.238.143 (STEALTH) [**]
10/19-00:18:50.190344 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.70.154 (THRESHOLD 12 connections exceeded in 4 seconds) [**]
10/19-00:18:51.624500 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.84.194 (THRESHOLD 12 connections exceeded in 1 seconds) [**]

The top source address MY.NET.163.107 generated no other alerts however this host performed rapid port scans throughout the days included in the analysis. There were no OOS packets originating from or destined to MY.NET.163.107. This host must be investigated immediately as it may be performing reconnaissance to be used in future attacks.

The second top source address MY.NET.84.194 was also rapidly port scanning. But the interesting events about MY.NET.84.194 were that twenty-two external hosts had SMB C access to it (Shown below). There were no OOS packets originating from or destined to MY.NET.163.107. The University should investigate this host to determine for possible compromise.

10/22-21:12:43.028494 [**] SMB C access [**] 200.81.25.71:1538 -> MY.NET.84.194:139
10/22-22:57:02.945146 [**] SMB C access [**] 218.164.32.212:3987 -> MY.NET.84.194:139
10/22-23:40:15.272972 [**] SMB C access [**] 203.247.145.52:19245 -> MY.NET.84.194:139
10/23-00:23:38.142951 [**] SMB C access [**] 202.163.200.194:52805 -> MY.NET.84.194:139
10/23-00:44:21.413511 [**] SMB C access [**] 212.175.59.27:1270 -> MY.NET.84.194:139
10/23-00:44:57.041338 [**] SMB C access [**] 202.5.88.228:21291 -> MY.NET.84.194:139

10/23-01:14:28.197173 [**] SMB C access [**] 61.223.139.116:3727 -> MY.NET.84.194:139
10/23-01:33:02.528990 [**] SMB C access [**] 195.252.104.123:3298 -> MY.NET.84.194:139
10/23-02:08:36.953255 [**] SMB C access [**] 203.40.237.56:3474 -> MY.NET.84.194:139
10/23-03:20:21.714197 [**] SMB C access [**] 67.122.139.207:4545 -> MY.NET.84.194:139
10/23-03:26:50.830345 [**] SMB C access [**] 218.63.126.204:3220 -> MY.NET.84.194:139
10/23-03:44:32.656335 [**] SMB C access [**] 217.99.50.142:1305 -> MY.NET.84.194:139
10/23-04:56:29.803962 [**] SMB C access [**] 61.11.5.62:4315 -> MY.NET.84.194:139
10/23-05:14:40.853323 [**] SMB C access [**] 81.195.245.4:2797 -> MY.NET.84.194:139
10/23-05:51:55.795855 [**] SMB C access [**] 218.18.196.130:21246 -> MY.NET.84.194:139
10/23-06:09:20.970768 [**] SMB C access [**] 81.196.41.135:3139 -> MY.NET.84.194:139
10/23-07:07:46.216710 [**] SMB C access [**] 138.89.11.51:4334 -> MY.NET.84.194:139
10/23-07:21:41.919662 [**] SMB C access [**] 212.171.114.204:2035 -> MY.NET.84.194:139
10/23-07:48:54.792973 [**] SMB C access [**] 80.24.76.35:60072 -> MY.NET.84.194:139
10/23-08:21:33.923940 [**] SMB C access [**] 80.55.99.82:35868 -> MY.NET.84.194:139
10/23-08:27:24.063126 [**] SMB C access [**] 200.138.171.221:1909 -> MY.NET.84.194:139
10/23-09:15:38.230479 [**] SMB C access [**] 211.204.168.58:2494 -> MY.NET.84.194:139

SMB C access

Description
This event indicates an attempt to access the default administrative share C$. If allowed, the attacker can access the C: filesystem. This event is specific to a vulnerability, but may have been caused by any of several possible exploits. Signatures used to detect this event are specific and consider the packet payload. The packet that caused this event is normally a part of an established TCP session, indicating that the source IP address has not been spoofed.

The impact of this alert is very serious if the signature is based on Snort signature ID 533[30] (Snort Signature shown below), because this particular signature triggers based on an “established” TCP session. Meaning that the sources have completed the three-way handshake and successfully connected to the destination hosts. This could have very serious implications, as the source may have administrative access to the destination. Another fact to be wary of is that many Trojans can reside on port 139. Such as Chode, God Message worm, Msinit, Netlog, Network, Qaz, Sadmind and SMB relay. Destination hosts for this event entailing MY.NET.84.228, MY.NET.191.52, MY.NET.152.166, MY.NET.111.225 and MY.NET.110.220 need to be investigated immediately for being compromise or any other malicious activity.

The following signature is the Snort rule that could have triggered theses events.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB C access";
flow:to_server,established; content: "|5c|C$|00 41 3a 00|";reference:arachnids,339; classtype:attempted-
recon; sid:533; rev:5;)
```

[30] http://www.snort.org/snort-db/sid.html?sid=533

Correlation
Al Maslowski-Yerges

Top Five Destination IPs

5090 MY.NET.84.228
1147 MY.NET.191.52
149 MY.NET.152.166
123 MY.NET.111.225
117 MY.NET.110.220

Top Five Source IPs

663 80.50.168.42
295 138.89.11.51
236 61.147.18.195
224 61.223.139.116
217 203.197.20.41

Sample alert trace:

```
10/19-15:14:56.409042  [**] SMB C access [**] 200.171.127.32:3822 -> MY.NET.190.97:139
10/19-15:14:56.627500  [**] SMB C access [**] 200.171.127.32:3824 -> MY.NET.190.102:139
10/19-16:50:13.981791  [**] SMB C access [**] 80.246.198.53:4420 -> MY.NET.190.102:139
10/19-20:37:53.785970  [**] SMB C access [**] 219.164.12.26:1216 -> MY.NET.190.97:139
10/19-20:38:23.807843  [**] SMB C access [**] 219.164.12.26:1216 -> MY.NET.190.97:139
10/19-20:38:55.151625  [**] SMB C access [**] 219.164.12.26:1217 -> MY.NET.190.101:139
10/19-22:06:44.248758  [**] SMB C access [**] 218.110.6.64:55697 -> MY.NET.190.102:139
```

The top source address 80.50.168.42 has had SMB C access to forty-four MY.NET.X.X hosts (sample count shown below). If this host does not need access to the MY.NET.X.X hosts then the IP address of 80.50.168.42 should be blocked. Furthermore, the hosts that 80.50.168.42 had SMB C access to should be considered compromised, taken offline and investigated. There were no OOS or SCAN file packets originating from or destined to 80.50.168.42.

80.50.168.42 SMB C Access to these hosts

Count	Host
40	MY.NET.72.170
40	MY.NET.72.142
35	MY.NET.72.212
35	MY.NET.72.178
32	MY.NET.69.223
29	MY.NET.70.144
28	MY.NET.69.222

MY.NET.30.4 activity

The top three destination ports are 51433, 80 and 524 with a destination address of MY.NET.30.4 could be considered normal if MY.NET.30.4 is a Novell webserver. As indicated by Novell technical documentation and James Filberto the "NetWare Enterprise Server installed, by default the Apache Web Server will get port 51080 for HTTP and 51443 for HTTPS."[31] TCP Port 524 handles the NCP Requests, and UDP port 524 handles NCP for time synchronization.[32] Traffic appears to be normal.

Correlation
James Filberto

Top three Destination Ports

10379 51443
3901 80
1211 524

Top Ten Source IPs

2934 68.55.85.180
2743 68.54.91.147
1124 172.142.110.232
997 151.196.19.202
474 68.33.10.149
441 68.55.62.79
440 68.55.205.180
397 68.84.131.246
365 151.196.34.226
351 151.196.42.116

Sample alert trace:

```
10/19-00:18:17.912445  [**] MY.NET.30.3 activity [**] 68.55.53.222:1032 -> MY.NET.30.3:524
10/19-00:18:54.077182  [**] MY.NET.30.3 activity [**] 165.247.89.143:2727 -> MY.NET.30.3:524
10/19-00:12:06.764221  [**] MY.NET.30.3 activity [**] 165.247.89.143:2727 -> MY.NET.30.3:524
10/19-00:12:46.049627  [**] MY.NET.30.3 activity [**] 165.247.89.143:2727 -> MY.NET.30.3:524
10/19-00:12:53.186755  [**] MY.NET.30.3 activity [**] 165.247.89.143:2727 -> MY.NET.30.3:524
10/19-00:33:11.687946  [**] MY.NET.30.3 activity [**] 68.55.233.51:63785 -> MY.NET.30.3:524
10/19-00:33:11.701717  [**] MY.NET.30.3 activity [**] 68.55.233.51:63785 -> MY.NET.30.3:524
```

[31] http://support.novell.com/servlet/tidfinder/2963227

[32] http://www.novell.com/coolsolutions/netware/features/a_ports_nw5_nw.html

EXPLOIT x86 NOOP

Description
This signature detects a string of the character 0x90. Depending on the context, this usually indicates the NOP operation in x86 machine code. Many remote buffer overflow exploits send a series of NOP (no-operation) bytes to pad their chances of successful exploitation. This event is specific to a vulnerability, but may have been caused by any of several possible exploits. Signatures used to detect this event are specific and consider the packet payload.

Top Five Destination IPs

375 MY.NET.15.198
366 MY.NET.27.103
235 MY.NET.5.95
200 MY.NET.80.16
190 MY.NET.81.18

Top destination ports
8400 135
2069 445
785 80

Top Five Source IPs

764 209.6.97.168
418 24.87.153.94
412 216.232.208.22
314 195.110.140.66
281 63.229.211.22

Sample alert trace:

```
10/19-04:29:02.364383  [**] EXPLOIT x86 NOOP [**] 211.239.106.153:3924 -> MY.NET.190.101:135
10/19-04:52:13.947285 [**] EXPLOIT x86 NOOP [**] 208.172.13.254:80 -> MY.NET.150.121:1253
10/19-04:52:35.992586 [**] EXPLOIT x86 NOOP [**] 208.172.13.190:80 -> MY.NET.150.121:1260
10/19-05:17:31.332630  [**] EXPLOIT x86 NOOP [**] 62.219.147.22:1501 -> MY.NET.190.101:135
10/19-07:15:35.384809  [**] EXPLOIT x86 NOOP [**] 211.92.136.145:64185 -> MY.NET.190.101:135
10/19-07:28:17.069309  [**] EXPLOIT x86 NOOP [**] 206.24.192.222:80 -> MY.NET.150.203:1648
10/19-07:49:10.887166  [**] EXPLOIT x86 NOOP [**] 80.184.135.158:1052 -> MY.NET.189.62:80
10/19-07:49:11.160567  [**] EXPLOIT x86 NOOP [**] 80.184.135.158:1052 -> MY.NET.189.62:80
10/19-07:49:11.216787  [**] EXPLOIT x86 NOOP [**] 80.184.135.158:1052 -> MY.NET.189.62:80
10/19-07:49:11.451998  [**] EXPLOIT x86 NOOP [**] 80.184.135.158:1052 -> MY.NET.189.62:80
10/19-07:49:11.507989  [**] EXPLOIT x86 NOOP [**] 80.184.135.158:1052 -> MY.NET.189.62:80
10/19-07:49:11.562796  [**] EXPLOIT x86 NOOP [**] 80.184.135.158:1052 -> MY.NET.189.62:80
10/19-07:49:11.675507  [**] EXPLOIT x86 NOOP [**] 80.184.135.158:1052 -> MY.NET.189.62:80
```

The following signature is the Snort rule that could have triggered theses events.

```
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 90|"; depth: 128; reference:arachnids,181;
classtype:shellcode-detect; sid:648; rev:6;)
```

This alert does have a high probability of false positives where legitimate traffic could trigger an event. "The x86 NOP can frequently be found in day-to-day traffic, particularly when transferring large files".[33] However to confirm the University's event, sifting through the packet payload is necessary.

The top destination ports of 135 and 445 causes concern as this may indicate this incident is related to the DCOM vulnerability found in Windows hosts "Buffer Overrun In RPCSS Service" Microsoft document MS03-039.[34] As noted from Internet Security Systems, "The RPC DCOM service may be accessible via several different ports over TCP or UDP. The most logical attack vector to exploit is TCP port 135. However, Microsoft has reported that the vulnerabilities may be exploited via UDP ports 135, 137, 138, 445 and TCP ports 135, 139, 445, 593. Microsoft has also reported that COM Internet Service (CIS) may be vulnerable over port 80 and port 443 if CIS is enabled[35]. More information about CIS is available in the corresponding Microsoft Security Bulletin MS03-039. The University should ensure these machines are patched for the DCOM vulnerability.

More details about this can be found at the following:

Buffer Overrun: CAN-2003-0715

Buffer Overrun: CAN-2003-0528

Denial of Service: CAN-2003-0605

The top source address 209.6.97.168 has generated another alert, SMB Name Wildcard. In addition, this host had been scanning the 130.85.X.X network on ports associated with the DCOM attack. (Sample shown below taken from SCAN files). There were no OOS file packets originating from or destined to 209.6.97.168.

```
Oct 23 03:01:13 209.6.97.168:1635 -> 130.85.10.87:445 SYN ******S*
Oct 23 03:01:13 209.6.97.168:1636 -> 130.85.10.87:139 SYN ******S*
Oct 23 03:01:13 209.6.97.168:1637 -> 130.85.10.85:445 SYN ******S*
Oct 23 03:01:14 209.6.97.168:1667 -> 130.85.10.85:139 SYN ******S*
Oct 23 09:48:20 209.6.97.168:4161 -> 130.85.80.148:139 SYN ******S*
Oct 23 09:48:20 209.6.97.168:137 -> 130.85.80.148:137 UDP
Oct 23 09:48:20 209.6.97.168:4162 -> 130.85.11.11:445 SYN ******S*
Oct 23 09:48:21 209.6.97.168:4180 -> 130.85.80.149:445 SYN ******S*
Oct 23 09:48:21 209.6.97.168:4181 -> 130.85.80.149:139 SYN ******S*
Oct 23 09:48:22 209.6.97.168:4190 -> 130.85.11.6:139 SYN ******S*
```

[33] http://www.snort.org/snort-db/sid.html?sid=648
[34] http://www.microsoft.com/technet/treeview/?url=/technet/security/bulletin/MS03-039.asp
[35] http://www.microsoft.com/technet/treeview/?url=/technet/security/bulletin/MS03-039.asp

The top destination address MY.NET.15.198 is being used to contact an IRC server. The MY.NET.15.198 user is contacting the IRC server through an XDCC client shown below in blue. Then the IRC server 64.157.246.22 is responding, shown in red. This host must be taken offline immediately and investigated for misuse.

10/23-16:24:19.142674 [**] [UMBC NIDS IRC Alert] XDCC client detected attempting to IRC [**] MY.NET.15.198:1039 -> 64.157.246.22:6667
10/23-16:24:57.743253 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan. [**] 64.157.246.22:6667 -> MY.NET.15.198:1039
10/23-16:45:57.896478 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan. [**] 64.157.246.22:6667 -> MY.NET.15.198:1063
10/23-16:29:09.013283 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan. [**] 64.157.246.22:6667 -> MY.NET.15.198:1043
10/23-16:40:42.936589 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan. [**] 64.157.246.22:6667 -> MY.NET.15.198:1058
10/23-17:00:39.812372 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan. [**] 64.157.246.22:6667 -> MY.NET.15.198:1078
10/23-17:01:42.802236 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan. [**] 64.157.246.22:6667 -> MY.NET.15.198:1079
10/23-17:02:07.904784 [**] [UMBC NIDS IRC Alert] XDCC client detected attempting to IRC [**] MY.NET.15.198:1080 -> 64.157.246.22:6667
10/23-16:43:14.155473 [**] [UMBC NIDS IRC Alert] XDCC client detected attempting to IRC [**] MY.NET.15.198:1061 -> 64.157.246.22:6667
10/23-16:44:17.119817 [**] [UMBC NIDS IRC Alert] XDCC client detected attempting to IRC [**] MY.NET.15.198:1062 -> 64.157.246.22:6667
10/23-17:04:34.804026 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan. [**] 64.157.246.22:6667 -> MY.NET.15.198:1082
10/23-16:55:25.715516 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan. [**] 64.157.246.22:6667 -> MY.NET.15.198:1073
10/23-16:56:27.855622 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan. [**] 64.157.246.22:6667 -> MY.NET.15.198:1074

The IRC server 64.427.246.22 generated no other alerts and there were no OOS or SCAN packets originating from or destined to 64.157.246.22. 64.157.246.22 had records at Dshield.[36]

IP Address: 64.157.246.22

HostName: 64.157.246.22

DShield Profile:

Country:	US
Contact E-mail:	abuse@level3.com
AS Number:	3356
Total Records against IP:	22

36 http://www.dshield.org/ipinfo.php?SANSDSHIELD=821e4330d3929580d3a23fd4831fb704&ip=64.157.246.22

Number of targets:	1
Date Range:	2003-12-02 to 2003-12-04

Update Summary

Top Port hit by this source:

Port	Attacks	Start	End
113	16	2004-01-05	2004-01-11

Correlation
Jeff Zahr Don Murdoch

connect to 515 from inside

"LPRng is an implementation of the Berkeley lpr print spooling utility. LPRng contains a function, use_syslog(), that returns user input to a string in LPRng that is passed to syslog() as the format string. As a result, it is possible to corrupt the program's flow of execution by entering malicious format specifiers. In testing this has been exploited to remotely elevate privileges." [37]

There are many exploits that can be used against the LPR service on Unix flavored operating systems. The Ramen trojan[38] exploits the LPRng vulnerability, and if the hosts becomes infected the trojan installs an http server on port 27374. However, the source port for all connections to 128.183.110.242 from host MY.NET.162.41 was 721. Which in Microsoft windows LPR defaults to using TCP ports 721-731.[39] This would rule out the connections of this particular alert being related to the Ramen Trojan. The lpdw0rm Trojan exploits a vulnerability in the LPRng (CVE-2000-0917[40]). The lpdw0rm trojan installs a backdoor on TCP port 666 and adds two logins (kork and kork2) with no passwords while one of the logins has root permissions.[41] Because the backdoor is installed on port 666, we can rule out the connections of this alert being related to the lpdw0rm Trojan. Traffic appears to be normal.

Correlation
Mike Poor

Top Destination IPs Destination Port

[37] http://www.securityfocus.com/bid/1712/discussion/
[38] http://xforce.iss.net/xforce/alerts/id/advise71
[39] http://support.microsoft.com/default.aspx?scid=kb;en-us;179156
[40] http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0917
[41] http://cert.uni-stuttgart.de/archive/incidents/2001/04/msg00308.html

7126 128.183.110.242 515

Top Source IP Source Port

7129 MY.NET.162.41 721

Sample alert trace:

```
10/21-22:38:11.817932  [**] connect to 515 from inside [**] MY.NET.162.41:721 ->
128.183.110.242:515
10/21-22:38:23.835850  [**] connect to 515 from inside [**] MY.NET.162.41:721 ->
128.183.110.242:515
10/21-22:45:45.494332  [**] connect to 515 from inside [**] MY.NET.162.41:721 ->
128.183.110.242:515
10/21-22:38:54.882174  [**] connect to 515 from inside [**] MY.NET.162.41:721 ->
128.183.110.242:515
10/21-22:39:00.891127  [**] connect to 515 from inside [**] MY.NET.162.41:721 ->
128.183.110.242:515
10/21-22:46:16.540625  [**] connect to 515 from inside [**] MY.NET.162.41:721 ->
128.183.110.242:515
10/21-22:39:40.951846  [**] connect to 515 from inside [**] MY.NET.162.41:721 ->
128.183.110.242:515
10/21-22:48:43.760113  [**] connect to 515 from inside [**] MY.NET.162.41:721 ->
128.183.110.242:515
10/21-22:42:08.171239  [**] connect to 515 from inside [**] MY.NET.162.41:721 ->
128.183.110.242:515
```

MY.NET.30.3 activity

The total number of connections to MY.NET.30.3 was 5730, with 97.8% (5607) of those connecting to port 524. I believe the destination host MY.NET.30.3 is a Novell server and the connections to the server are either Novell NCP requests (TCP 524) or Novell NCP time synchronization (UDP 524). Novell technical documentation states that TCP Port 524 handles the NCP Requests, and UDP port 524 handles NCP for time synchronization.[42] Traffic appears to be normal.

Correlation
Steven Gamble

Top Ten Source IPs

1224 68.57.90.146
735 68.55.27.157
639 68.55.233.51

[42] http://www.novell.com/coolsolutions/netware/features/a_ports_nw5_nw.html

605 68.55.62.79
572 141.157.6.106
463 68.55.105.5
209 68.55.53.222
200 68.55.250.229
107 68.48.217.68
101 165.247.97.243

Sample alert trace:

```
10/19-00:18:17.912445  [**] MY.NET.30.3 activity [**] 68.55.53.222:1032 -> MY.NET.30.3:524
10/19-00:18:54.077182  [**] MY.NET.30.3 activity [**] 165.247.89.143:2727 -> MY.NET.30.3:524
10/19-00:12:06.764221  [**] MY.NET.30.3 activity [**] 165.247.89.143:2727 -> MY.NET.30.3:524
10/19-00:12:46.049627  [**] MY.NET.30.3 activity [**] 165.247.89.143:2727 -> MY.NET.30.3:524
10/19-00:12:53.186755  [**] MY.NET.30.3 activity [**] 165.247.89.143:2727 -> MY.NET.30.3:524
10/19-00:33:11.687946  [**] MY.NET.30.3 activity [**] 68.55.233.51:63785 -> MY.NET.30.3:524
10/19-00:33:11.701717  [**] MY.NET.30.3 activity [**] 68.55.233.51:63785 -> MY.NET.30.3:524
```

TCP SRC and DST outside network

The alerts were triggered because the source and destination addresses were both outside the network. With the proper ingress and egress filtering, this type of traffic would not be typical unless crafted. The alerts were generated by the Snort Stream4 preprocessor. "The stream4 module provides TCP stream reassembly and stateful analysis capabilities to Snort. Robust stream reassembly capabilities allow Snort to ignore "stateless" attacks such as stick and snot produce."[43] This is a concern because earlier versions of Snort (pre 2.0 rc1) are vulnerable to heap overflow (CAN-2003-0209). If an attacker sends crafted packets the Snort sensor could crash.[44] [45] [46]

This could be a very serious threat, if the University has Snort sensors prior to 2.0 rc1 and the traffic contains the exploit code. Alternatively if this traffic is generated from the inside and destined for the outside, the University could have some compromised hosts that are participating in DOS attacks. Further investigation is necessary.

Top Five Destination IPs

2854 218.16.124.131
1421 211.91.144.72
42 68.55.61.253

[43] http://www.snort.org/docs/writing_rules/chap2.html#tth_sEc2.4.5
[44] http://www.snort.org/advisories/snort-2003-04-16-1.txt
[45] http://www.kb.cert.org/vuls/id/139129
[46] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0209

14 63.211.66.115
11 66.93.118.125

Top Five Destination Ports
2860 21
1420 996
68 80
54 143
19 5190

Top Five Source IPs

4280 169.254.244.56
78 68.55.0.64
42 10.0.1.12
28 192.168.1.101
23 192.168.0.5

Sample alert trace:

10/19-10:24:19.369080 [**] TCP SRC and DST outside network [**] 169.254.244.56:2320 -> 218.16.124.131:21
10/19-10:24:23.501993 [**] TCP SRC and DST outside network [**] 169.254.244.56:2319 -> 211.91.144.72:996
10/19-10:24:25.612836 [**] TCP SRC and DST outside network [**] 169.254.244.56:2321 -> 218.16.124.131:21
10/19-10:24:29.535948 [**] TCP SRC and DST outside network [**] 169.254.244.56:2319 -> 211.91.144.72:996
10/19-10:24:37.501417 [**] TCP SRC and DST outside network [**] 169.254.244.56:2320 -> 218.16.124.131:21
10/19-10:24:40.510008 [**] TCP SRC and DST outside network [**] 169.254.244.56:2320 -> 218.16.124.131:21
10/19-10:24:41.617381 [**] TCP SRC and DST outside network [**] 169.254.244.56:2319 -> 211.91.144.72:996
10/19-10:24:50.667892 [**] TCP SRC and DST outside network [**] 169.254.244.56:2319 -> 211.91.144.72:996
10/19-10:24:58.631676 [**] TCP SRC and DST outside network [**] 169.254.244.56:2320 -> 218.16.124.131:21
10/19-10:25:05.061422 [**] TCP SRC and DST outside network [**] 169.254.244.56:2322 -> 211.91.144.72:996

The top source address is using the 169.254.X.X IP address (Shown above). The address 169.254.244.56 falls into the 169.254.0.0–169.254.255.255 range and as previously stated are commonly found when a Windows 2000 host is unable to find a DHCP server. Perhaps there are misconfigured Windows 2000 hosts that require static addresses instead of DHCP addresses. Therefore these hosts are using the 169.254.X.X addresses and sending these packets on the network. Another explanation could be that these hosts are unable to connect to

the DHCP server at the time. Either way these hosts require attention/reconfiguration.

The top two destination address 218.16.124.131 and 211.91.144.72 displayed no evidence of alerts other than "TCP SRC and DST outside network." In addition there were no OOS or SCAN packets originating from or destined to 218.16.124.131 or 211.91.144.72.

Correlations
Michael Wilkinson

External RPC call

Description
"Remote procedure calls (RPCs) allow programs on one computer to execute procedures on a second computer by passing data and retrieving the results. RPC is therefore widely used for many distributed network services such as remote administration, NFS file sharing, and NIS. However there are numerous flaws in RPC which are being actively exploited. Many RPC services execute with elevated privileges that can provide an attacker unauthorized remote root access to vulnerable systems."[47]

The signatures for RPC are abundant, more information about RPC signature scan be downloaded from www.snort.org. All of the alerts had a destination port of 111. Listed below is a short summary of the RPC alert listings at whitehats.com with a destination port of 111[48].

- IDS23/portmap-request-rexd [UDP any -> 111]
- IDS18/portmap-request-admind [UDP any -> 111]
- IDS10/portmap-request-rstatd [UDP any -> 111]
- IDS14/portmap-request-yppasswd [UDP any -> 111]
- IDS407/portmap-request-nlockmgr [UDP any -> 111]
- IDS22/portmap-request-pcnfsd [UDP any -> 111]
- IDS17/portmap-request-cmsd [UDP any -> 111]
- IDS13/portmap-request-mountd [UDP any -> 111]
- IDS428/portmap-listing-111 [TCP any -> 111]
- IDS25/portmap-request-selection_svc [UDP any -> 111]
- IDS12/portmap-request-ypserv [UDP any -> 111]
- IDS21/portmap-request-nisd [UDP any -> 111]
- IDS16/portmap-request-bootparam [UDP any -> 111]
- IDS125/portmap-request-ypupdated [UDP any -> 111]
- IDS542/portmap-request-espd [UDP any -> 111]

[47] http://www.sans.org/top20/?printer=Y#u2
[48] http://www.whitehats.com/cgi/arachNIDS/Search?f%3A1=DestinationPort&e%3A1=%3D%3D+x&v%3A1=111&search=&sort=TIME&format=DEFAULT&max=950&grouping=or

• IDS20/portmap-request-sadmind [UDP any -> 111]
• IDS15/portmap-request-status [UDP any -> 111]
• IDS24/portmap-request-ttdbserv [UDP any -> 111]
• IDS19/portmap-request-autofsd [UDP any -> 111]
• IDS133/portmap-request-rusers [UDP any -> 111]

Source host 193.114.70.169 appears to be scanning the network for Unix flavored boxes with the Portmapper port open. Notice in the sample alert trace (located below) 193.114.70.169 is scanning the Universities MY.NET network on October 23, 2003 for 6 hours from 430-1030 (2841 count). The University should investigate the probes to be sure the attacker was not successful in executing commands on the destination hosts. In addition, the University should edit ingress filtering to limit access to only hosts that require outside-to-inside connection to port 111.

Correlation

Al Williams

Top Five Destination IPs

18	MY.NET.24.65
8	MY.NET.6.15
6	MY.NET.28.9
5	MY.NET.75.140
5	MY.NET.60.172

Top Four Source IPs

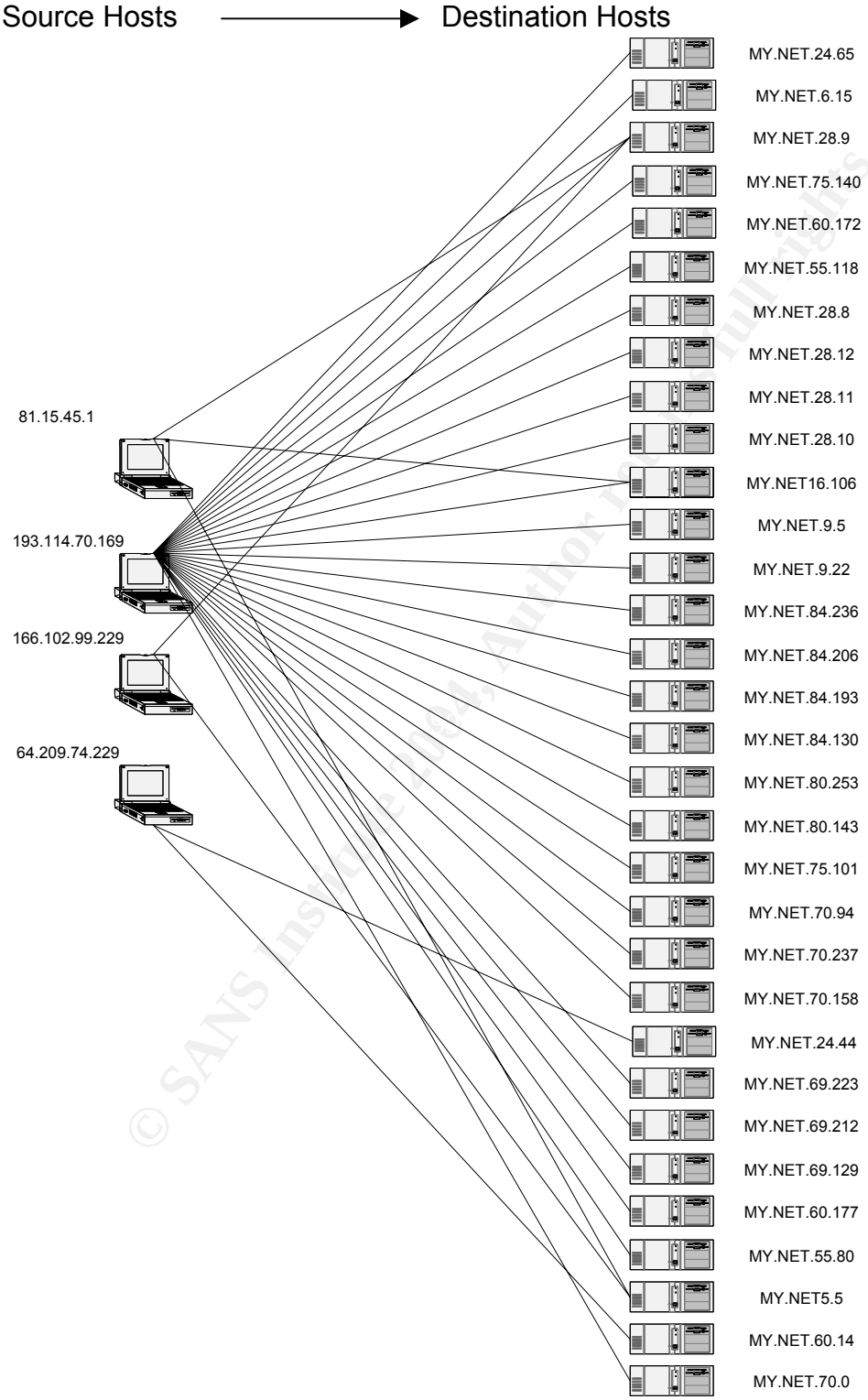
2838	193.114.70.169
420	81.15.45.1
7	166.102.99.229
2	64.209.74.229

Sample alert trace:

```
10/23-08:05:42.351109  [**] External RPC call [**] 193.114.70.169:2755 -> MY.NET.97.44:111
10/23-08:05:43.679797  [**] External RPC call [**] 193.114.70.169:2795 -> MY.NET.97.45:111
10/23-08:05:46.641380  [**] External RPC call [**] 193.114.70.169:2577 -> MY.NET.97.0:111
10/23-08:05:47.046911  [**] External RPC call [**] 193.114.70.169:2846 -> MY.NET.97.74:111
10/23-08:05:47.144993  [**] External RPC call [**] 193.114.70.169:2844 -> MY.NET.97.73:111
10/23-08:05:47.753882  [**] External RPC call [**] 193.114.70.169:2846 -> MY.NET.97.74:111
10/23-08:05:48.552338  [**] External RPC call [**] 193.114.70.169:3957 -> MY.NET.60.40:111
10/23-08:05:52.177899  [**] External RPC call [**] 193.114.70.169:2855 -> MY.NET.97.80:111
10/23-08:05:52.941365  [**] External RPC call [**] 193.114.70.169:2855 -> MY.NET.97.80:111
10/23-08:05:54.166482  [**] External RPC call [**] 193.114.70.169:2859 -> MY.NET.97.85:111
10/23-08:05:54.887360  [**] External RPC call [**] 193.114.70.169:2859 -> MY.NET.97.85:111
```

Link Graph

Top 4 External IPs connecting to RPC (111) port to Top 32 Internal hosts



High port 65535 tcp - possible Red Worm – traffic

Port 65535 traffic could possibly be a worm (Adore, or sometimes known as Red) or Trojans (Sins or RC1[49]), however port 65535 can be a normally used ephemeral port. Most of this traffic appears to be p2p file sharing as now you can customize the ports that are being used. In addition to customizing ports, p2p traffic can also be tunneled through normally allowed ports such as 80 (http) and 25 (smtp) with programs such as HTTP-tunnel and KaazaHttp[50]. Until we have access to packet payload, we will be unable to determine without a doubt if these events are false positives. Traffic appears to be normal.

Top Five Destination IPs

1112	200.96.13.157
1022	MY.NET.80.105
320	66.66.71.92
268	MY.NET.153.141
23	202.156.254.68

Top Destination Ports

1630	65535
1022	3951
268	2071
89	25
61	80

Top Five Source IPs

1114	MY.NET.80.105
1023	200.96.13.157
310	MY.NET.153.141
284	66.66.71.92
24	MY.NET.24.20

Sample alert trace:

10/23-12:12:39.833841 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.80.105:3951 -> 200.96.13.157:65535
10/23-12:12:46.725832 [**] High port 65535 tcp - possible Red Worm - traffic [**] 200.96.13.157:65535 -> MY.NET.80.105:3951

[49] http://www.simovits.com/nyheter9902.html
[50] http://www.ezebusiness2000.com/College.htm

10/23-12:13:05.726468 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.80.105:3951 -> 200.96.13.157:65535
10/23-12:13:13.455257 [**] High port 65535 tcp - possible Red Worm - traffic [**] 200.96.13.157:65535 -> MY.NET.80.105:3951
10/23-12:13:15.502024 [**] High port 65535 tcp - possible Red Worm - traffic [**] 200.96.13.157:65535 -> MY.NET.80.105:3951
10/23-12:13:17.782272 [**] High port 65535 tcp - possible Red Worm - traffic [**] 200.96.13.157:65535 -> MY.NET.80.105:3951
10/23-12:13:17.782654 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.80.105:3951 -> 200.96.13.157:65535
10/23-12:13:23.789437 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.80.105:3951 -> 200.96.13.157:65535
10/23-12:13:28.242019 [**] High port 65535 tcp - possible Red Worm - traffic [**] 200.96.13.157:65535 -> MY.NET.80.105:3951
10/23-12:13:28.290316 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.80.105:3951 -> 200.96.13.157:65535

The top destination address 200.96.13.157 Red Worm alerts were either originated from or destined to the top source address MY.NET.80.105. MY.NET.80.105 uses source port 3951 to connect to 200.96.13.157 on port 65535. We need more information regarding the connections between these two hosts to determine if MY.NET.80.105 has been compromised. If the signature that triggered these alerts is only based on a port of 65535 there is a high possibility that this is a false positive. Perhaps an application is using this particular port, however until we have access to packet payload, we will be unable to determine without a doubt if these events are false positives. There were no other alerts from 200.96.13.157. There were also no OOS or SCAN packets originating from or destined to 200.96.13.157.

The top source address MY.NET.80.105 had these other alerts (Trace 1). There are three external hosts that had SMB C access on MY.NET.80.105. As stated before in the "SMB C access" section, this alert triggers on an established session. If these hosts are not authorized for access, MY.NET.80.105 should be checked for possible compromise. In Trace 2, this is the output of OOS packets destined for MY.NET.80.105 with a destination port of 2740. Port 2740 is commonly associated with the Alarm service. There were also no SCAN packets originating from or destined to MY.NET.80.105.

Trace 1

10/22-20:40:03.866948 [**] Incomplete Packet Fragments Discarded [**] 162.84.167.130:0 -> MY.NET.80.105:0
10/23-08:49:13.667435 [**] SMB C access [**] 218.14.66.28:1026 -> MY.NET.80.105:139
10/23-09:02:47.543980 [**] SMB C access [**] 213.77.167.125:21085 -> MY.NET.80.105:139
10/23-09:02:50.242028 [**] SMB C access [**] 213.77.167.125:21085 -> MY.NET.80.105:139
10/23-09:32:51.308468 [**] SMB C access [**] 66.222.145.74:3541 -> MY.NET.80.105:139
10/23-09:53:00.291182 [**] NMAP TCP ping! [**] 194.78.59.253:80 -> MY.NET.80.105:2740
10/23-10:08:11.284772 [**] SMB C access [**] 193.252.212.236:13008 -> MY.NET.80.105:139
10/23-10:12:55.074409 [**] SMB Name Wildcard [**] MY.NET.80.105:137 -> 148.233.104.107:137

Trace 2

10/20-19:13:52.824515 217.5.90.20:49357 -> MY.NET.80.105:2740

10/23-16:15:04.633102 80.143.11.192:53335 -> MY.NET.80.105:2740
10/23-16:15:07.626017 80.143.11.192:53335 -> MY.NET.80.105:2740
10/23-16:15:13.623871 80.143.11.192:53335 -> MY.NET.80.105:2740
10/23-16:18:58.961766 212.21.255.78:55191 -> MY.NET.80.105:2740
10/23-16:19:07.949486 212.21.255.78:55191 -> MY.NET.80.105:2740
10/23-16:20:40.761366 217.230.14.202:3704 -> MY.NET.80.105:2740
10/23-18:33:27.420929 81.56.214.240:36474 -> MY.NET.80.105:2740
10/23-18:33:30.409590 81.56.214.240:36474 -> MY.NET.80.105:2740
10/23-18:33:36.407165 81.56.214.240:36474 -> MY.NET.80.105:2740
10/23-19:58:26.385159 80.143.11.192:48393 -> MY.NET.80.105:2740
10/23-20:41:58.338297 24.43.50.231:36521 -> MY.NET.80.105:2740
10/23-20:42:04.297890 24.43.50.231:36521 -> MY.NET.80.105:2740
10/23-21:33:07.432159 203.218.221.69:33456 -> MY.NET.80.105:2740
10/23-21:33:10.493641 203.218.221.69:33456 -> MY.NET.80.105:2740

Correlation
Al Maslowski-Yerges, James Fillberto

Possible trojan server activity

"27374 is one of the default ports of the BackDoor-G2.svr.gen trojan, more commonly known as SubSeven." "… trojan of choice for most DDoS attacks and clone attacks on specific services, such as IRC. Scans of this port are often accompanied by scans of port 1243, another default SubSeven port of older versions."[51]
All of these Trojans run on port 27374 Bad Blood, Fake SubSeven, li0n, Ramen, Seeker, SubSeven , SubSeven 2.1 Gold, Subseven 2.1.4 DefCon 8, SubSeven 2.2, SubSeven Muie, The Saint[52]

According to incidents.org during the days being analyzed in this report, port 27374 had an increase of activity.[53] The chart located below displays the increase in port 27374 activity from October 19 2003 through October 23 2003 MY.NET.24.34, MY.NET.12.6, and MY.NET.163.249 were the top three MY.NET destinations. Although port 27374 can be used by applications, these hosts should be considered suspicious and investigated further for compromise. If the connections by these hosts are legitimate, perhaps the University can encourage their users to use different ports for their applications. Thereby reducing the number of false positives.

2003-10-23	1846	50093	205162
2003-10-22	3205	76786	166015

[51] http://dshield.org/ports/port27374.php
[52] http://www.simovits.com/nyheter9902.html
[53] http://isc.incidents.org/port_details.html?port=27374

2003-10-21	2963	77250	265643
2003-10-20	2443	76664	265984
2003-10-19	2120	74470	318235
2003-10-18	2415	55080	151368
2003-10-17	2103	16702	64186
2003-10-16	1607	14151	68704

[54]

Top Five Destination IPs

562 MY.NET.163.249
403 200.163.61.175
29 MY.NET.12.6
21 MY.NET.24.34
18 64.41.183.130

Top Five Source IPs

553 200.163.61.175
410 MY.NET.163.249
304 66.169.146.100
71 212.95.105.31
63 67.64.149.135

Sample alert trace:

10/19-05:33:19.184884 [**] Possible trojan server activity [**] 67.64.149.135:4348 -> MY.NET.190.66:27374
10/19-05:33:21.787472 [**] Possible trojan server activity [**] 67.64.149.135:4439 -> MY.NET.190.97:27374
10/19-05:33:21.790175 [**] Possible trojan server activity [**] MY.NET.190.97:27374 -> 67.64.149.135:4439
10/19-05:33:21.793867 [**] Possible trojan server activity [**] 67.64.149.135:4443 -> MY.NET.190.101:27374
10/19-05:33:21.795385 [**] Possible trojan server activity [**] MY.NET.190.101:27374 -> 67.64.149.135:4443
10/19-05:33:21.797607 [**] Possible trojan server activity [**] 67.64.149.135:4444 -> MY.NET.190.102:27374
10/19-05:33:29.172362 [**] Possible trojan server activity [**] 67.64.149.135:4748 -> MY.NET.190.164:27374
10/19-05:33:29.183224 [**] Possible trojan server activity [**] 67.64.149.135:4750 -> MY.NET.190.166:27374

---

[54] http://isc.incidents.org/port_details.html?port=27374&repax=1&tarax=2&srcax=2&percent=N&days=70&Redraw=Submit+Query

Correlation
Doug Kite, Ronnie Clark

The top destination address MY.NET.163.249 is the third highest port scanner in the PORTSCAN DETECTED section with 15237 alerts. By investigating the other alerts generated by MY.NET.163.249 (shown below) we discover that this host appears to be infected with the Msblast worm which exploits a vulnerability found in Microsoft hosts. (MS03-026[55] CAN-2003-0352[56]) According to Network Associates[57], "When run, it scans a random IP range to look for vulnerable systems on TCP port 135. The worm attempts to exploit the DCOM RPC vulnerability on the found systems to create a remote shell on TCP port 4444. It then instructs the system to download the worm to the %WinDir%\system32 directory and execute it. (The target system is issued a TFTP command to downloads the worm from the infected host system [TFTP UDP port 69]." "The worm contains a payload to initiate a Denial of Service attack against **windowsupdate.com** after August 16."[58] "This payload involves sending 40 byte SYN packets to windowsupdate.com on TCP port 80 for the purpose of preventing users from patching their systems via Windows Update. The source IP address is spoofed on each packet, using a random local CLASS B IP."[59]

As we can see below there is a connection from 212.81.218.50 to MY.NET.163.249 port 135 (Shown in blue), indicative of an infected MSBlast host scanning for a vulnerable Windows host on port 135. Then the worm appears to have successfully created a remote shell on port 4444 (shown in red) seen by the alert "Internal MSBlast Infection request". Finally, host MY.NET.163.249 is caught establishing a connection to a TFTP server (217.232.24.100) to download the worm (shown in green). MY.NET.163.247 needs to be immediately taken offline, apparently there are some hosts within the Universities network that are not patched for this vulnerability. There was no evidence of SCAN packets. Unfortunately, with the limited amount of data we are not able to verify if MY.NET.163.249 is sending SYN packets to Windows Update because the source address will be spoofed.

10/22-19:03:38.010514 [**] EXPLOIT x86 NOOP [**] 212.81.218.50:1753 -> MY.NET.163.249:135
10/22-19:03:38.714721 [**] [UMBC NIDS] Internal MSBlast Infection Request [**] MY.NET.163.249:4444 -> 212.81.218.50:2066
10/22-19:10:55.778374 [**] EXPLOIT x86 NOOP [**] 200.96.9.59:3191 -> MY.NET.163.249:135
10/22-19:54:48.831648 [**] EXPLOIT x86 NOOP [**] 172.200.112.17:3004 -> MY.NET.163.249:135
10/22-21:19:30.893002 [**] EXPLOIT x86 NOOP [**] 213.189.168.208:2364 -> MY.NET.163.249:135
10/22-21:32:07.975083 [**] SMB C access [**] 203.131.85.145:21146 -> MY.NET.163.249:139

[55] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp
[56] http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352
[57] http://vil.nai.com/vil/content/v_100547.htm
[58] http://vil.nai.com/vil/content/v_100547.htm
[59] http://vil.nai.com/vil/content/v_100547.htm

10/22-22:37:32.850196 [**] [UMBC NIDS] Internal MSBlast Infection Request [**]
MY.NET.163.249:4444 -> 130.67.101.88:4865
10/23-00:01:18.689313 [**] EXPLOIT x86 NOOP [**] 12.81.161.61:3198 -> MY.NET.163.249:135
10/23-00:02:06.925557 [**] SMB C access [**] 202.107.222.214:2841 -> MY.NET.163.249:139
10/23-00:29:20.695761 [**] EXPLOIT x86 NOOP [**] 200.29.18.227:3393 -> MY.NET.163.249:135
10/23-02:13:13.592809 [**] EXPLOIT x86 NOOP [**] 218.167.190.194:1246 -> MY.NET.163.249:445
10/23-03:37:11.775372 [**] SMB C access [**] 203.54.177.201:1475 -> MY.NET.163.249:139
10/23-03:41:11.205240 [**] EXPLOIT x86 NOOP [**] 4.3.6.237:3390 -> MY.NET.163.249:135
10/23-04:28:11.752683 [**] SMB C access [**] 61.217.131.168:4123 -> MY.NET.163.249:139
10/23-05:09:13.322190 [**] SMB C access [**] 203.238.39.147:4143 -> MY.NET.163.249:139
10/23-06:10:44.357874 [**] SMB C access [**] 195.175.122.32:1231 -> MY.NET.163.249:139
10/23-07:35:09.198944 [**] SMB C access [**] 201.4.133.245:1291 -> MY.NET.163.249:139
10/23-08:19:42.706316 [**] TFTP - Internal UDP connection to external tftp server [**]
MY.NET.163.249:4711 -> 217.232.24.100:69
10/23-08:19:43.796262 [**] TFTP - Internal UDP connection to external tftp server [**]
217.232.24.100:69 -> MY.NET.163.249:4711
10/23-08:19:43.796606 [**] TFTP - Internal UDP connection to external tftp server [**]
MY.NET.163.249:4711 -> 217.232.24.100:69
10/23-08:19:44.697604 [**] TFTP - Internal UDP connection to external tftp server [**]
MY.NET.163.249:4711 -> 217.232.24.100:69
10/23-08:19:53.687602 [**] TFTP - Internal UDP connection to external tftp server [**]
217.232.24.100:69 -> MY.NET.163.249:4711

In addition these OOS packets were discovered when searching for MY.NET.163.249. Port 6667 is commonly known as an IRC server port or Trojan port. With all of the information gathered, it is clear that MY.NET.163.249 must be taken offline immediately.

10/23-08:19:55.950681 200.96.192.58:1961 -> MY.NET.163.249:6667
10/23-08:43:13.960177 200.96.192.58:4009 -> MY.NET.163.249:6667
10/23-10:32:31.055054 200.96.192.58:2737 -> MY.NET.163.249:6667
10/23-10:41:50.507347 200.96.192.58:3014 -> MY.NET.163.249:6667

Summary of Alerts (Occurrence > 1)

199212 SMB Name Wildcard
118308 PORTSCAN DETECTED
28546 SMB C access
15606 MY.NET.30.4 activity
11563 EXPLOIT x86 NOOP
7131 connect to 515 from inside
5726 MY.NET.30.3 activity
4518 TCP SRC and DST outside network
3266 External RPC call
3172 High port 65535 tcp - possible Red Worm - traffic
2009 Possible trojan server activity
1825 ICMP SRC and DST outside network
752 NMAP TCP ping!
633 UMBC NIDS IRC Alert

494 SUNRPC highport access!
455 Null scan!
438 High port 65535 udp - possible Red Worm - traffic
106 UMBC NIDS IRC Alert
105 FTP passwd attempt
84 Back Orifice
83 TFTP - Internal UDP connection to external tftp server
74 Incomplete Packet Fragments Discarded
62 Tiny Fragments - Possible Hostile Activity
53 EXPLOIT x86 stealth noop
51 NETBIOS NT NULL session
38 DDOS shaft client to handler
27 EXPLOIT x86 setuid 0
26 EXPLOIT x86 setgid 0
25 EXPLOIT NTPDX buffer overflow
14 FTP DoS ftpd globbing
14 DDOS mstream client to handler
13 TFTP - Internal TCP connection to external tftp server
11 TFTP - External UDP connection to internal tftp server
10 RFB - Possible WinVNC - 010708-1
10 Attempted Sun RPC high port access
5 HelpDesk MY.NET.70.49 to External FTP
4 NIMDA - Attempt to execute cmd from campus host
2 Traffic from port 53 to port 123
2 TFTP - External TCP connection to internal tftp server
2 Probable NMAP fingerprint attempt
2 External FTP to HelpDesk MY.NET.70.50
2 External FTP to HelpDesk MY.NET.70.49
2 External FTP to HelpDesk MY.NET.53.29
2 connect to 515 from outside

Other Interesting information

The top OOS source address was 217.174.98.145. This host has been particularly busy scanning the Universities network with 1062 “PORTSCAN DETECTED” alerts. In addition, 217.174.98.145 was sending SYN packets with the reserved bits set to 130.85.111.52 destination port 25. This was discovered when searching through the SCAN files.

```
Oct 20 02:53:31 217.174.98.145:47370 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 03:12:09 217.174.98.145:48027 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 04:25:12 217.174.98.145:49477 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 04:53:39 217.174.98.145:50049 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 04:53:45 217.174.98.145:50049 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 06:19:37 217.174.98.145:51624 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 06:19:40 217.174.98.145:51624 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
```

Oct 20 06:19:46 217.174.98.145:51624 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 06:19:58 217.174.98.145:51624 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 06:20:22 217.174.98.145:51624 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 06:28:04 217.174.98.145:51785 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 06:58:29 217.174.98.145:52444 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 07:11:40 217.174.98.145:52719 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 07:45:16 217.174.98.145:53397 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS
Oct 20 07:45:25 217.174.98.145:53397 -> 130.85.111.52:25 SYN 12****S* RESERVEDBITS

The second top OOS source address 195.111.1.93 was found to be port scanning with 936 occurrences of "PORTSCAN DETECTED" alerts. When searching through the SCAN files 195.111.1.93 was also sending SYN packets with the reserve bits set to 130.85.X.X hosts.

Oct 19 07:49:45 195.111.1.93:41442 -> 130.85.100.165:80 SYN 12****S* RESERVEDBITS
Oct 19 07:42:45 195.111.1.93:48608 -> 130.85.100.165:80 SYN 12****S* RESERVEDBITS
Oct 19 07:50:15 195.111.1.93:44761 -> 130.85.100.165:80 SYN 12****S* RESERVEDBITS
Oct 19 07:43:15 195.111.1.93:51475 -> 130.85.100.165:80 SYN 12****S* RESERVEDBITS
Oct 19 07:43:45 195.111.1.93:54379 -> 130.85.100.165:80 SYN 12****S* RESERVEDBITS
Oct 19 07:44:15 195.111.1.93:57996 -> 130.85.100.165:80 SYN 12****S* RESERVEDBITS
Oct 19 07:44:45 195.111.1.93:33842 -> 130.85.100.165:80 SYN 12****S* RESERVEDBITS
Oct 19 17:03:11 195.111.1.93:32930 -> 130.85.60.14:80 SYN 12****S* RESERVEDBITS
Oct 19 17:04:11 195.111.1.93:42480 -> 130.85.60.14:80 SYN 12****S* RESERVEDBITS
Oct 19 17:04:40 195.111.1.93:46835 -> 130.85.60.14:80 SYN 12****S* RESERVEDBITS

The top OOS destinations MY.NET.111.52 and MY.NET.12.6 appear to be mail servers when searching through the alert files. The "Possible Trojan server activity" from MY.NET.12.6 are most likely attributed to 64.41.183.130 randomly using an ephemeral port 27374 to communicate with MY.NET.12.6 (shown in blue). The "External MiMail alert" from MY.NET.12.6 indicates that it is probably a mail server (shown in green).

10/19-17:03:18.782290 [**] NMAP TCP ping! [**] 213.30.181.1:80 -> MY.NET.111.52:25
10/20-01:04:59.645357 [**] NMAP TCP ping! [**] 195.6.62.30:80 -> MY.NET.111.52:25
10/22-15:50:55.956792 [**] Possible trojan server activity [**] 64.41.183.130:27374 -> MY.NET.12.6:25
10/22-15:51:02.611643 [**] Possible trojan server activity [**] 64.41.183.130:27374 -> MY.NET.12.6:25
10/22-15:51:02.611793 [**] Possible trojan server activity [**] 64.41.183.130:27374 -> MY.NET.12.6:25
10/22-15:51:23.601697 [**] Possible trojan server activity [**] 64.41.183.130:27374 -> MY.NET.12.6:25
10/22-15:51:23.678433 [**] Possible trojan server activity [**] MY.NET.12.6:25 -> 64.41.183.130:27374
10/22-15:51:23.929117 [**] Possible trojan server activity [**] 64.41.183.130:27374 -> MY.NET.12.6:25
10/22-15:51:30.675746 [**] Possible trojan server activity [**] MY.NET.12.6:25 -> 64.41.183.130:27374
10/22-19:20:55.181951 [**] [UMBC NIDS] External MiMail alert [**] 68.100.94.160:3139 -> MY.NET.12.6:25
10/22-19:14:18.484863 [**] [UMBC NIDS] External MiMail alert [**] 66.160.91.140:3857 -> MY.NET.12.6:25
10/22-19:38:19.765249 [**] [UMBC NIDS] External MiMail alert [**] 68.100.94.160:4548 -> MY.NET.12.6:25

The top SCAN file sources 130.85.1.3 and 130.85.70.154 had no occurrences of alerts and OOS packets. The Top SCAN file destinations 192.26.92.30 and 192.55.83.30 also had no occurrences of alerts and OOS packets.

Top Ten Talkers in MY.NET

115624 MY.NET.80.51
72067 MY.NET.150.133
7132 MY.NET.162.41
3100 MY.NET.29.2
1290 MY.NET.84.224
1118 MY.NET.80.105
474 MY.NET.150.198
447 MY.NET.163.249
311 MY.NET.153.141
195 MY.NET.42.9

Top Alert Source IPs

115757 MY.NET.80.51
72074 MY.NET.150.133
29572 MY.NET.163.107
27223 MY.NET.84.194
15684 MY.NET.163.249
7132 MY.NET.162.41
5312 MY.NET.111.72
4508 MY.NET.70.154
4279 169.254.244.56
4222 MY.NET.73.94

Top Alert Destination IPs

15606 MY.NET.30.4
7126 128.183.110.242
5728 MY.NET.30.3
5092 MY.NET.84.228
2854 218.16.124.131
1421 211.91.144.72
1268 198.62.205.6
1251 151.197.115.143
1209 193.114.70.169
1147 MY.NET.191.52

Top Alert Destination Ports

199121 137
28609 139
10379 51443
8459 135

7128 515
6818 524
5019 80
3265 111
3003 21
2100 445

Top OOS Source IPs

1142 217.174.98.145
1130 195.111.1.93
1038 212.16.0.33
973 158.196.149.61
792 194.67.62.194
685 82.82.64.209
682 213.23.46.99
472 195.208.238.143
454 195.14.47.202
437 200.77.250.50

Top OOS Destination IPs

7867 MY.NET.111.52
4115 MY.NET.12.6
1672 MY.NET.100.165
1504 MY.NET.69.181
1407 MY.NET.24.44
839 MY.NET.75.240
734 MY.NET.84.143
471 MY.NET.24.34
327 MY.NET.100.230
282 MY.NET.6.7

Top OOS Destination Ports

13447 25
4194 80
1489 8887
1255 4662
406 113
246 110
90 1214
56 6881
41 6883
26 443

Top Scan Source IPs

2166933 130.85.1.3
1294187 130.85.70.154
966595 130.85.163.107
888185 130.85.84.194
669973 130.85.163.249
273705 130.85.42.1
213577 130.85.70.129
211571 130.85.1.5
175961 130.85.80.149
171526 130.85.111.72

Top Scan Destination IPs

57085 192.26.92.30
43945 192.55.83.30
32455 203.20.52.5
32276 130.94.6.10
30276 130.85.15.27
26947 204.152.186.189
26036 131.118.254.33
25707 216.109.116.17
24599 131.118.254.34
23570 131.118.254.35

Top Scan Destination Ports

5808397 135
2370289 53
2048637 80
211497 445
132539 22321
109499 6257
93326 137
73033 4000
63769 554
50813 7674

Five Selected External Sources

1. 193.114.70.169 was selected because of the attempts to connect to port 111 on 2838 destination hosts located on the University's network. Registration information by http://www.ripe.net/whois. Dshield had no records against this IP address.

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html
inetnum:        81.15.0.0 - 81.15.127.255
netname:        IS-LINANET-20020805
descr:          Lina.Net
descr:          PROVIDER
country:        IS
admin-c:        SSS92-RIPE
tech-c:         LNOC2-RIPE
tech-c:         GITH1-RIPE
status:         ALLOCATED PA
source:         RIPE
mnt-by:         RIPE-NCC-HM-MNT
mnt-lower:      LINANET-MNT
mnt-routes:     LINANET-MNT
notify:         noc@lina.net
notify:         gunni@sensa.is
changed:        hostmaster@ripe.net 20020805
changed:        hostmaster@ripe.net 20030813 # is.linanet.markom via
https://lirportal.ripe.net
changed:        hostmaster@ripe.net 20030813 # is.linanet.markom via
https://lirportal.ripe.net
route:          81.15.0.0/17
origin:         AS15605
descr:          Lina.net network
mnt-by:         LINANET-MNT
notify:         noc@lina.net
changed:        markom@lina.net 20030813
source:         RIPE
role:           Lina.Net Network Operations Centre
address:        Lina.Net hf.
address:        Skaftahlid 24
address:        105 Reykjavik
address:        Iceland
phone:          +354 5596000
fax-no:         +354 5596099
e-mail:         noc@lina.net
admin-c:        SSS92-RIPE
tech-c:         MARK2-RIPE
nic-hdl:        LNOC2-RIPE
mnt-by:         LINANET-MNT
notify:         noc@lina.net
changed:        markom@lina.net 20030813
changed:        markom@lina.net 20030904
source:         RIPE
person:         Stefan Snorri Stefansson
address:        Lina.Net hf.
address:        Skaftahlid 24
address:        105 Reykjavik
address:        Iceland
phone:          +354 5596000
e-mail:         sss@lina.net
```

```
e-mail:       stefan.snorri.stefansson@lina.net
nic-hdl:      SSS92-RIPE
mnt-by:       LINANET-MNT
changed:      markom@lina.net 20030813
source:       RIPE
person:       Gunnar Ingvi Thorisson
address:      Sensa ehf.
address:      www.sensa.is
address:      Lynghalsi 4
address:      110 Reykjavik
address:      Iceland
phone:        +354-577-1340
e-mail:       gunni@sensa.is
e-mail:       gunni@gunni.is
nic-hdl:      GITH1-RIPE
notify:       ripe@sensa.is
changed:      ripe@sensa.is 20030226
source:       RIPE
```

2. 130.85.1.3 was selected for the simple reason of being the top source IP address for Scans. Registration information by http://ws.arin.net/cgi-bin/whois.pl. Dshield[60] had records against this IP address noted below registration information

```
OrgName:    University of Maryland Baltimore County
OrgID:      UMBC
Address:    UMBC University Computing
City:       Baltimore
StateProv:  MD
PostalCode: 21250
Country:    US

NetRange:   130.85.0.0 - 130.85.255.255
CIDR:       130.85.0.0/16
NetName:    UMBCNET
NetHandle:  NET-130-85-0-0-1
Parent:     NET-130-0-0-0-0
NetType:    Direct Assignment
NameServer: UMBC5.UMBC.EDU
NameServer: UMBC4.UMBC.EDU
NameServer: UMBC3.UMBC.EDU
Comment:
RegDate:    1988-07-05
Updated:    2000-03-17

TechHandle: JJS41-ARIN
TechName:   Suess, John J.
TechPhone:  +1-410-455-2582
TechEmail:  jack@umbc.edu
```

[60] http://www.dshield.org/ipinfo.php?ip=130.85.1.3&Submit=Submit

```
# ARIN WHOIS database, last updated 2003-12-09 19:15
```

IP Address: 130.85.1.3

HostName: UMBC3.UMBC.EDU

DShield Profile:

Country:	US
Contact E-mail:	jack@UMBC.EDU
AS Number:	0
Total Records against IP:	60
Number of targets:	10
Date Range:	2003-11-16 to 2003-12-09

Top 10 Ports hit by this source:

Port	Attacks	Start	End
53	61	2003-11-16	2003-12-09
1024	2	2003-12-04	2003-12-04

3. 200.163.61.175 was selected because it was the top source IP address for the “Possible trojan server activity” alert. Registration information by http://whois.lacnic.net/. Dshield had no records against this IP address.

```
% Copyright LACNIC lacnic.net
%  The data below is provided for information purposes
%  and to assist persons in obtaining information about or
%  related to AS and IP numbers registrations
%  By submitting a whois query, you agree to use this data
%  only for lawful purposes.
%  2003-12-08 19:35:44 (BRST -02:00)

inetnum:     200.128/9
status:      allocated
owner:       Comite Gestor da Internet no Brasil
ownerid:     BR-CGIN-LACNIC
responsible: Frederico A C Neves
address:     Av. das Nações Unidas, 11541, 7° andar
address:     04578-000 - São Paulo - SP
country:     BR
phone:       +55 11 9119-0304 []
owner-c:     CGB
tech-c:      CGB
inetrev:     200.128/9
nserver:     NS.DNS.BR
nsstat:      20031201 AA
nslastaa:    20031201
nserver:     NS1.DNS.BR
nsstat:      20031201 AA
```

```
nslastaa:    20031201
nserver:     NS2.DNS.BR
nsstat:      20031201 AA
nslastaa:    20031201
remarks:     These addresses have been further assigned to Brazilian
users.
remarks:     Contact information can be found at the WHOIS server
located
remarks:     at whois.registro.br and at http://whois.nic.br
created:     19950104
changed:     20020902

nic-hdl:     CGB
person:      Comite Gestor da Internet no Brasil
e-mail:      blkadm@NIC.BR
address:     Av. das Nações Unidas, 11541, 7° andar
address:     04578-000 - São Paulo - SP
country:     BR
phone:       +55 19 9119-0304 []
created:     20020902
changed:     20020902

% whois.lacnic.net accepts only direct match queries.
% Types of queries are: POCs, ownerid, CIDR blocks, IP
% and AS numbers.
```

4. 68.55.85.180 was selected because it was the one of the top external source Alert IP address. Registration information by http://ws.arin.net/cgi-bin/whois.pl. Dshield had no records against this IP address.

```
CustName:   Comcast Cable Communications, Inc.
Address:    3 Executive Campus
Address:    5th Floor
City:       Cherry Hill
StateProv:  NJ
PostalCode: 08002
Country:    US
RegDate:    2003-03-19
Updated:    2003-03-19

NetRange:   68.55.0.0 - 68.55.255.255
CIDR:       68.55.0.0/16
NetName:    BALTIMORE-A-6
NetHandle:  NET-68-55-0-0-1
Parent:     NET-68-32-0-0-1
NetType:    Reassigned
Comment:    NONE
RegDate:    2003-03-19
Updated:    2003-03-19

TechHandle: IC161-ARIN
TechName:   Comcast Cable Communications, Inc.
TechPhone:  +1-856-317-7300
TechEmail:  cips-ip-registration@cable.comcast.com

OrgAbuseHandle: NAPO-ARIN
```

```
OrgAbuseName:   Network Abuse and Policy Observance
OrgAbusePhone:  +1-856-317-7272
OrgAbuseEmail:  abuse@comcast.net

OrgTechHandle: IC161-ARIN
OrgTechName:   Comcast Cable Communications, Inc.
OrgTechPhone:  +1-856-317-7300
OrgTechEmail:  cips-ip-registration@cable.comcast.com

# ARIN WHOIS database, last updated 2003-12-07 19:15
```

5. 217.174.98.145 was selected because it was the top OOS source IP. Registration information by http://www.ripe.net/whois. Dshield had no records against this IP address.

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html
inetnum:        217.174.96.0 - 217.174.98.255
netname:        SUNET2000
descr:          Sunet 2000 Ltd
descr:          120 8 Prishvina Moscow
descr:          Russia
country:        RU
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
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
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
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
```

Defensive Recommendations

The University should evaluate their existing security policies, configurations and procedures to ensure they meet their expectations. The following list is not comprehensive, just a starting point for the University.

1. Security policy
    a. Development
    b. Frequent evaluation/maintenance
2. Proper ingress and egress filtering on network.
    a. Placing the proper ingress filtering will limit the ability of packets not deemed "normal" to traverse the University's network devices. For example, the University should not allow packets into the network if both destination and source address is from the "outside" or "inside". Proper ingress filtering limits an attackers ability to "spoof" traffic to the University's internal network.
    b. Using the appropriate egress filtering will limit the University's internal users ability to spoof their source address. This should reduce the "undesirable" traffic leaving the network.
    c. Proper filtering should also reduce the quantity of the "Top Ten" Alerts (SMB Name Wildcard, SMB C access External RPC call, TCP SRC and DST outside network, etc.…) that were very prevalent for the University. In addition, filtering will help limit p2p file sharing and other miscellaneous activities.
3. Packet capturing boxes strategically placed
    a. Install packet-capturing tool such as Tcpdump to provide University with an audit trail. The audit trail will assist the IDS analyst in determining if alerts are false positives.
    b. Installing these boxes will definitely consume hard disk space. The University should consider filtering out certain traffic to minimize disk consumption. For example, the University could choose to not log outgoing port 80 traffic. Port 80 is used for example purposes only, the University should determine what traffic should be logged.
4. Securing the Servers
    a. Configure Servers securely
        i. Turn off or disable services that are not required.
        ii. Turn on and configure logging

   b. Perform regularly scheduled maintenance
      i. Operating system patches (i.e. Microsoft Servers, Unix Servers, etc)
      ii. Application patches (sendmail, MS SQL, Oracle, etc.)
      iii. Frequent backups
   c. Consider Host-based IDS/Server Firewall
5. Securing the workstations
   a. Configure workstations securely
      i. Turn off or disable services that are not required.
      ii. Give internal users limited permissions. (Users that only need web access should not be administrators)
      iii. Turn on and configure logging
   b. Perform regularly scheduled maintenance
      i. Operating system patches
      ii. Application patches
   c. Consider Desktop Firewall
6. Network Segmentation
   a. Placing servers/critical nodes on a separate network segment
7. Additional IDS sensors
   a. After network segmentation is implemented, add IDS to server segment(s).

Description of Analysis

The Analysis was done on a dual processor Pentium III machine with 1 gigabyte of memory running Red Hat Linux 9.0. The files (alerts,oos and scans) were downloaded from http://www.incidents.org/logs/. The files were unzipped using gunzip and concatenated into three files alert, scan and oos. The files were analyzed with a series scripts written by Chris Baker http://www.giac.org/practical/Chris_Baker_GCIA.zip and Chris Calabrese http://www.giac.org/practical/Chris_Calabrese_GCIA.html. Some of the scripts from Chris Baker and Chris Calabrese were slightly modified. Using these scripts and other tools like grep, wc and sort I was able to analyze the data and then correlate some of the data to other students practical assignments.

Chris baker's scripts

```
grep "\[\*\*\]" alerts.txt | grep -v spp_portscan | cut -d \> -f 2 | cut -d : -f 1 | sed s/\
//g | sort | uniq -c | sort -nr > alerts.dstips.log
```

```
grep "\[\*\*\]" alerts.txt | grep -v spp_portscan | grep -v Tiny\ Fragments | grep -v
ICMP\ SRC | cut -d \> -f 2 | cut -d : -f 2 | sed s/\ //g | sort | uniq -c | sort -nr >
alerts.dstports.log

grep "\[\*\*\]" alerts.txt | grep -v spp_portscan | cut -d \] -f 3 | cut -d \- -f 1 | cut -d : -
f 1 | sed s/\ //g >> alerts.srcips.log.unsorted
grep PORTSCAN alerts.txt | cut -d \] -f 2 | cut -d \  -f 6 | sed s/\ //g >>
alerts.srcips.log.unsorted
cat alerts.srcips.log.unsorted | sort | uniq -c | sort -nr > alerts.srcips.log
rm alerts.srcips.log.unsorted

Tally of dst ips:
grep "..\/..\-..\:..\:"  oos.txt | cut -d \> -f 2 | cut -d \: -f 1 | sed s/\ //g | sort | uniq -c |
sort -nr > oos.dstips.log

Tally of dst ports:
grep "..\/..\-..\:..\:"  oos.txt | cut -d \> -f 2 | cut -d \: -f 2 | sed s/\ //g | sort | uniq -c |
sort -nr > oos.dstports.log

Tally of src ips:
grep "..\/..\-..\:..\:"  oos.txt | cut -d \> -f 1 | cut -d \  -f 2 | cut -d \: -f 1 | sed s/\ //g |
sort | uniq -c | sort -nr > oos.srcips.log

The top 10 MY.NET talkers
#!/bin/sh
grep "..\/..\-..\:..\:" oos.txt | cut -d \> -f 1 | cut -d \ -f 2 | cut -d \: -f 1 | sed s/\ //g >>
top10talkers.log.unsorted
grep "\[\*\*\]" alerts.txt | grep -v spp_portscan | cut -d \] -f 3 | cut -d \- -f 1 | cut -d : -
f 1 | sed s/\ //g >> top10talkers.log.unsorted
grep PORTSCAN alerts.txt | cut -d \f -f 1 | cut -d \: -f 4 >>
top10talkers.log.unsorted
cat top10talkers.log.unsorted | sort | uniq -c | sort -nr > top10talkers.log
rm top10talkers.log.unsorted

Chris Calabrese's scripts
Top Scan Destination Ports
cat scan.txt | awk '$5 == "->" { print $6 }' | cut -d : -f 2 | sort | uniq -c | sort -rn |
head

cat scan.txt | awk '$5 == "->" { print $4 ":" $6 }' | cut -d : -f 1,4 > ports.out

grep ":$port" ports.out | cut -d : -f 1 | sort | uniq -c | sort -rn | head

Top Scan Source Hosts by Traffic
```

cat scan.txt | awk '$5 == "->" { print $4 }' | cut -d : -f 1 | sort | uniq -c | sort -rn | head

Top Scan Destination Hosts by Traffic
gzcat scan*.gz | awk '$5 == "->" { print $6 }' | cut -d : -f 1 | sort | uniq -c | sort -rn | head

Modified scripts from Chris baker and Chris Calabrese
Sample scripts
grep "MY.NET.30.4 activity" alert.txt > mynet304.txt
cat mynet304.txt | cut -d \> -f 2 | cut -d : -f 1 | sed s/\ //g | sort | uniq -c | sort -nr > mynet304.dstips
grep "SMB Name Wildcard" alert.txt | cut -d \> -f 2 | cut -d : -f 1 | sed s/\ //g | sort | uniq -c | sort -nr > smbname.dstips

References:

I. Incidents.org
URL:
http://www.incidents.org
Internet Storm Center. 08 Nov 2003
<http://isc.incidents.org/port_details.html?port=27374>

II. SecurityFocus

http://www.securityfocus.com
Multiple Vendor LPRng User-Supplied Format String Vulnerability.
Security Focus. 31 Nov 2003
<http://www.securityfocus.com/bid/1712/discussion/>

III. Snort.org

Writing Snort Rules. Snort.org 05 Nov 2003.
<http://www.snort.org/docs/writing_rules/chap2.html#tth_sEc2.4.2>
Snort.org. 24 Nov 2003
<http://www.snort.org/snort-db/sid.html?sid=533>
Snort.org. 24 Nov 2003
<http://www.snort.org/advisories/snort-2003-04-16-1.txt>

IV. Novell
URL:
http://support.novell.com/servlet/tidfinder/2963227
http://www.novell.com/coolsolutions/netware/features/a_ports_nw5_nw.html

V. Microsoft
URL:

http://support.microsoft.com/default.aspx?scid=kb;en-us;179156

VI. Cert.org
URL:
http://www.kb.cert.org/vuls/id/139129

VII. Simovits Consulting
Ports used by trojans. Simovits Consulting. 24 Nov 2003
<http://www.simovits.com/nyheter9902.html>

VIII. Dshield.org
Port 27374 – SubSeven. Dshield.org. 01 Dec 2003
<http://dshield.org/ports/port27374.php>
IP info. Dshield.org. 01 Dec 2003
<http://www.dshield.org/ipinfo.php?ip=130.85.1.3&Submit=Submit>

IX. Latin American and Caribbean Internet Address Registry
URL:
http://whois.lacnic.net/

X. American Registry for Internet Numbers
URL:
http://www.arin.net/index.html

XI. Asia Pacific Network Information Center
URL:
http://www.apnic.net

XII. SANS
Port 137 Scan. SANS. 01 Dec 2003
http://www.sans.org/resources/idfaq/port_137.php
U2 Remote Procedure Calls (RPC). SANS 01 Dec 2003
http://www.sans.org/top20/?printer=Y - u2

XIII. Whitehats.com
URL:
http://www.whitehats.com/
Whitehats.com. 02 Dec 2003
<http://www.whitehats.com/cgi/arachNIDS/Search?f%3A1=DestinationPort&e%3A1=%3D%3D+x&v%3A1=111&search=&sort=TIME&format=DEFAULT&max=950&grouping=or>

XIV. CVE
URL:
http://www.cve.mitre.org/
CAN-2003-0209. Common Vulnerabilities and Exposures. 10 Nov 2003
<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0209>

XV. GIAC student practicals
URL:
http://www.giac.org/practical/GCIA/Doug_Kite_GCIA.pdf
http://www.giac.org/practical/GCIA/Bill_Young_GCIA.pdf
http://www.giac.org/practical/GCIA/James_Filiberto_GCIA.pdf
http://www.giac.org/practical/GCIA/Steven_Gamble_GCIA.pdf
http://www.giac.org/practical/GCIA/Don_Murdoch_GCIA.pdf

http://www.giac.org/practical/GCIA/Al_Maslowski-Yerges_GCIA.pdf
http://www.giac.org/practical/GCIA/Susan_Kovacevich_GCIA.pdf
http://www.giac.org/practical/Mike_Poor_GCIA.doc
http://www.giac.org/practical/GCIA/Ronnie_Clark_GCIA.doc
http://www.giac.org/practical/michael_wilkinson_gcia.doc
http://www.giac.org/practical/Jeff_Zahr_GCIA.doc
http://www.giac.org/practical/Martin_Kirwan.doc

Last Updated: July 27th, 2017

Upcoming Training

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