



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at <http://www.giac.org/registration/gcia>

SANS GIAC Level Two – Intrusion Detection In Depth
GCIA Practical Assignment – SANS EAST Washington, D.C.
Practical Assignment Version 3.0 – August 13, 2001

Kenneth James Harwood

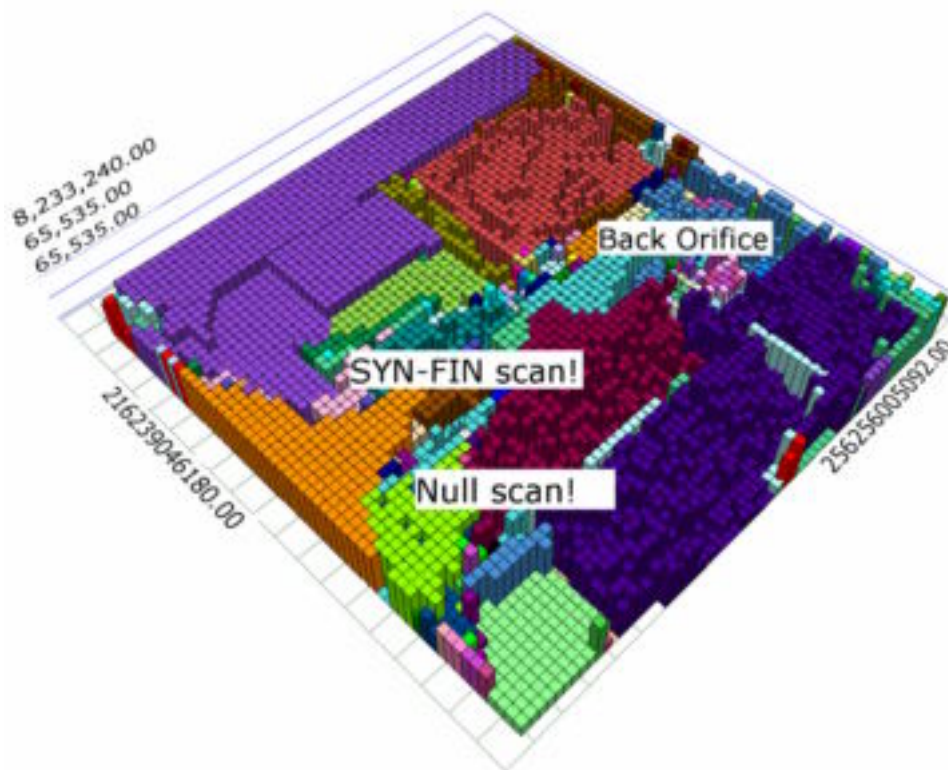


Table of contents

ASSIGNMENT 1 - DESCRIBE THE STATE OF INTRUSION DETECTION	5
Snort's Flexible Response Capabilities Examined as a Security Policy Enforcement	
Tool	5
Introduction	5
Method	6
TEST NETWORK	6
PROCEDURES	7
Results	7
ICMP	7
TELNET	9
FTP	11
HTTP	14
SMB	22
SUMMARY AND DISCUSSION	27
REFERENCES	28
APPENDIX	29
1-A SNORT Configuration File	29
ASSIGNMENT 2- NETWORK DETECTS	33
Detect-1 Comcast@Home Cable Modem IGMP Broadcast	33
1. Source of Trace	33
2. Detect was Generated by	33
3. Probability the Source Address was Spoofed	33
4. Description of Attack	34
5. Attack Mechanism	34
6. Correlations	35
7. Evidence of Active Targeting	38
8. Severity	38
9. Defensive Recommendations	38
10. Multiple Choice Test Question	39
Detect-2 Scan Proxy Attempt	39
1. Source of Trace	39
2. Detect was Generated by	39
3. Probability the Source Address was Spoofed	40
4. Description of Attack	41
5. Attack Mechanism	41
6. Correlations	41
7. Evidence of Active targeting	42
8. Severity	42
9. Defensive Recommendations	42
10. Multiple Choice Test Question	42
Detect-3 MISC Large ICMP Packet Broadcast Scan	43
1. Source of Trace	43
2. Detect was Generated by	43
3. Probability the Source Address was Spoofed	45

4. Description of Attack.....	46
5. Attack Mechanism	46
6. Correlations	47
7. Evidence of Active targeting.....	47
8. Severity	47
9. Defensive Recommendations.....	48
10. Multiple Choice Test Question	48
Detect-4 Port 6112 SYN Scan	48
1. Source of Trace	48
2. Detect was Generated by	48
3. Probability the Source Address was Spoofed	50
4. Description of Attack.....	50
5. Attack Mechanism	51
6. Correlations	52
7. Evidence of Active targeting.....	53
8. Severity	53
9. Defensive Recommendations.....	53
10. Multiple Choice Test Question	53
Detect-5 Scripted FTP Scan & Storage Test	54
1. Source of Trace	54
2. Detect was Generated by	54
3. Probability the Source Address was Spoofed	60
4. Description of Attack.....	60
5. Attack Mechanism	61
6. Correlations	62
7. Evidence of Active targeting.....	63
8. Severity	63
9. Defensive Recommendations.....	63
10. Multiple Choice Test Question	63
References.....	64
ASSIGNMENT 3- ANALYZE THIS	67
GIAC University Intrusion Detection Preliminary Audit.....	67
Executive Summary	67
List of files used.....	68
Top 10's	68
Alerts	68
Scans	70
Top 10 Talkers in MY.NET	73
Top10 Alerts Signatures	73
ICMP traceroute	73
connect to 515 from inside	74
spp_http_decode: IIS Unicode attack detected	76
MISC Large UDP Packet	78
SNMP public access	79
INFO MSN IM Chat data	82
INFO - ICQ Access.....	84

High port 65535 udp – possible Red Worm – traffic	86
ICMP Router Selection.....	89
SMB Name Wildcard.....	90
Top 10 Scans	93
UDP scan	93
TCP *****S* scan.....	94
TCP ***** scan.....	95
TCP ****p*** scan.....	96
TCP *2UA**** scan.....	96
TCP ***A*R*F scan.....	97
TCP *2*A**S* scan.....	97
TCP *2**PR*F scan.....	97
TCP *2UAP**** scan.....	98
TCP *2***R** scan.....	98
TCP *2****SF scan.....	99
Out of Specification Packets.....	99
External sources Requiring Further Investigation.....	100
Description of the analysis process.....	107
Systems and Tools Used.....	107
The Procedure	107
References.....	110

ASSIGNMENT 1 - DESCRIBE THE STATE OF INTRUSION DETECTION

Snort's Flexible Response Capabilities Examined as a Security Policy Enforcement Tool

Introduction

The lightweight Network Intrusion Detection (NID) tool Snort, version 1.8.3, written by Martin Roesch has been used extensively as a NID, packet sniffer, and packet logger. Documentation and How-to's for these uses are plentiful on: www.snort.org/documentation.html, <http://archives.neohapsis.com/archives/snort/>, and at SANS own Reading Room: http://rr.sans.org/intrusion/intrusion_list.php just as a starter. But very little information has been posted regarding Snort's sparsely documented "other feature" flexible response. Many postings on the Internet cite the existence of the "flex-resp" feature, but very few provide any insight into how to use it. One notable exception is a posting by Nathan Labadie dated 09/30/2001, who claims flexible response is quite effective at slowing the spread of the CodeRed Internet worm.

When the flexible response rule option is used and an alert is generated, snort will take action by "reacting" to the stimulating alert and attempt to "snipe" a session in progress. It does this by crafting a spoofed packet from either, or both, of the source and destination hosts' internet addresses, and essentially tries to hijack the session by sending a RESET in the case of a TCP based protocol, or an ICMP error message if ICMP or UDP is involved.

This technology has awesome potential for security related purposes. Imagine a network appliance intelligent enough (in theory) to react to any hostile traffic by actively closing any attempted session, automatically, and without operator intervention in near real time. Unfortunately this ultimate security appliance isn't available since no one has invented the perfect Intrusion Detection System. One that knows what to expect from the hackers out there and can identify "hostile traffic" 100% of the time with no false positives. So all perspective Intrusion Detection Analysts can rest assured there will be a future job market. However there may be other uses for such a "snort reactor" system in today's networks.

The purpose of this paper is to examine the flexible response rule option and apply it with a simplified rule set designed to test Snort's performance in "sniping" or hijacking current network sessions of some common protocols, and then to suggest that snort can be deployed as an effective security policy enforcement tool using flexible response. Rules were constructed to snipe: ICMP ping, Telnet, FTP, HTTP, and Windows SMB file sharing traffic between two hosts to observe the resulting traffic, and to demonstrate effective flex-resp rules. These protocols were chosen as they are simple enough for testing flex-resp rules, and they are often used to violate or ignore a given Network Security Policy.

The scope of this paper will be testing snort rules on a local network segment with the intent of demonstrating snort's flex-resp capability. Secondly, that snort can provide an additional layer of network security between the firewall and local hosts. These tests are meant as a proof of

concept and should not be considered conclusive, nor even applicable to all network configurations. The latter part of this paper will discuss the limitations of these results and suggest alternative methods for further and more extensive testing of flex-resp. Christian Lademann, author of the **readme.flexresp** file included the windows version of snort 1.8.2 warns, “Consider this code as ALPHA. Heavy testing is needed.” Yet the promise of flex-resp may be realized now to some degree. Mr. Roesch assures us that release 1.8.3 now has working “flexresp code” as stated on his download page at www.snort.org/downloads/html.

Method

TEST NETWORK

The test network was composed of three nodes as shown in Figure 1, connected by a 10BaseT hub. The reserved class C address space of 192.168.0.0 was used on this network. Machine names and their associated MAC layer addresses are shown below each machine.

A RedHat Linux 7.1 kernel 2.4.2 Laptop served as the Snort sensor/reactor (referred to hereafter as SnortBox). It was installed with Snort 1.8.3 (build 88), using the configure `--enable-flexresp` switch while compiling, along with libnet 1.02-a, and libpcap 0.6-2-9.

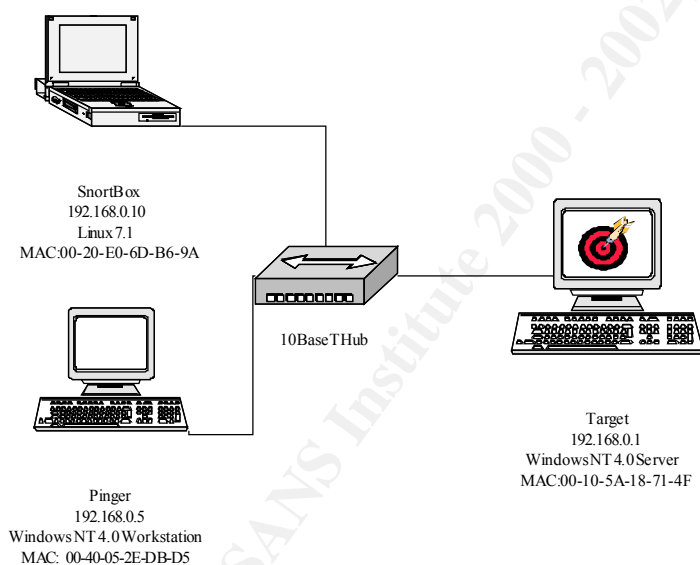


Figure 1

The target machine (referred to hereafter as Target) was installed with Windows NT 4.0 server patched with service pack 6a, and Internet Explorer 5.00.2314.1003. It was installed with Microsoft's Internet Information Services 4.0, and served up: two versions of the default web page on ports 80 and 81, and the default ftp site 192.168.0.1. The GoodTech telnet server for Windows, www.goodtechsys.com/downloads2.asp was installed and allowed to run with default settings. Target also shared two folders named NTShare and OKShare, which were created in the

root folder of C:\ with default NTFS file and share permissions granting the EVERYONE group full access.

The attacking machine (referred to hereafter as Pinger) was installed with Windows NT 4.0 Workstation, and patched with service pack 6a, Internet Explorer 5.50.4807.2300 and all Microsoft recommended critical updates as of 12/27/01.

Both Windows machines used only the TCP/IP protocol as NetBEUI was removed after installation. Identical security credentials were used to logon to the Windows machines with Administrator rights. It should also be noted that only Pinger used a 10BaseT Ethernet card, the other machines had 100BaseT Ethernet cards installed. Pinger and Target were each connected to the hub with less than five feet of CAT5 UTP wire. The SnortBox was connected by CAT5 UTP wire over an undetermined distance in excess of forty feet.

PROCEDURES

The actual procedures used in each test are described in the appropriate sections. However all tests were repeated a number of times to ensure the sample packets presented here are truly representative of the result. The same snort configuration file was used in all tests. It is based on the default **snort.conf** file in the 1.8.3 distribution release with all default pre and post processors in place. However unessential comments and all rules not tested in this paper were removed. The configuration file appears in Appendix 1-A. A commented copy of each rule tested which proved effective, to any degree, appears in the **snort.config**. The most effective rules for each protocol are reported in the results section.

Results

ICMP

Internet Control Message Protocol, or ICMP, traffic is used to report errors conditions in the network between hosts using TCP/IP. An administrator might want to restrict ICMP traffic from leaving the network due to the valuable information which it can report to a malicious user. A ping scan of a remote subnet will quickly reveal any live hosts if they are not silenced from responding an intervening router, firewall, or configured not to respond to this traffic. A malicious user could do a ping scan from an internal host to gather reconnaissance information, which could then be transferred out of the network by other means, such as a compromised host. But ICMP traffic may be desirable within the local network for this very same information to the Administrator.

From a command prompt the command "ping 192.168.0.1" was issued on Pinger. The following snort alert rule was constructed to identify this traffic, and attempt to block it.

```
alert icmp any any -> $Target any (resp:icmp_all; \
  msg: "Pinger is calling, Hang Up!");
```

The following is the snort alerts & partial capture of the resulting traffic:

(note: This hex data automatically displayed by snort 1.8-win32 has been removed for clarity)

```
[**] [1:0:0] Pinger is calling, Hang Up! [**]  
01/04-19:17:52.663924 192.168.0.5 -> 192.168.0.1  
ICMP TTL:32 TOS:0x0 ID:47393 IpLen:20 DgmLen:60  
Type:8 Code:0 ID:1 Seq:21 ECHO
```

```
[**] [1:0:0] Pinger is calling, Hang Up! [**]  
01/04-19:17:53.683924 192.168.0.5 -> 192.168.0.1  
ICMP TTL:32 TOS:0x0 ID:47649 IpLen:20 DgmLen:60  
Type:8 Code:0 ID:1 Seq:22 ECHO
```

```
[**] [1:0:0] Pinger is calling, Hang Up! [**]  
01/04-19:17:54.683924 192.168.0.5 -> 192.168.0.1  
ICMP TTL:32 TOS:0x0 ID:47905 IpLen:20 DgmLen:60  
Type:8 Code:0 ID:1 Seq:23 ECHO
```

```
[**] [1:0:0] Pinger is calling, Hang Up! [**]  
01/04-19:17:55.683924 192.168.0.5 -> 192.168.0.1  
ICMP TTL:32 TOS:0x0 ID:48161 IpLen:20 DgmLen:60  
Type:8 Code:0 ID:1 Seq:24 ECHO
```

```
01/04-19:17:52.663924 0:40:5:2E:DB:D5 -> 0:10:5A:18:71:4F type:0x800 len:0x4A  
192.168.0.5 -> 192.168.0.1 ICMP TTL:32 TOS:0x0 ID:47393 IpLen:20 DgmLen:60  
Type:8 Code:0 ID:1 Seq:21 ECHO  
+++++
```

```
01/04-19:17:52.663924 0:10:5A:18:71:4F -> 0:40:5:2E:DB:D5 type:0x800 len:0x4A  
192.168.0.1 -> 192.168.0.5 ICMP TTL:128 TOS:0x0 ID:18717 IpLen:20 DgmLen:60  
Type:0 Code:0 ID:1 Seq:21 ECHO REPLY  
+++++
```

```
01/04-19:17:52.663924 0:20:E0:6D:B6:9A -> 0:40:5:2E:DB:D5 type:0x800 len:0x6E  
192.168.0.1 -> 192.168.0.5 ICMP TTL:64 TOS:0xF4 ID:20738 IpLen:20 DgmLen:96  
Type:3 Code:0 DESTINATION UNREACHABLE: NET UNREACHABLE  
** ORIGINAL DATAGRAM DUMP:  
192.168.0.5 -> 192.168.0.1 ICMP TTL:32 TOS:0x0 ID:8633 IpLen:20 DgmLen:20  
** END OF DUMP  
+++++
```

```
01/04-19:17:52.663924 0:20:E0:6D:B6:9A -> 0:40:5:2E:DB:D5 type:0x800 len:0x6E  
192.168.0.1 -> 192.168.0.5 ICMP TTL:64 TOS:0xF4 ID:30280 IpLen:20 DgmLen:96  
Type:3 Code:1 DESTINATION UNREACHABLE: HOST UNREACHABLE  
** ORIGINAL DATAGRAM DUMP:  
192.168.0.5 -> 192.168.0.1 ICMP TTL:32 TOS:0x0 ID:8633 IpLen:20 DgmLen:20  
** END OF DUMP  
+++++
```

```
01/04-19:17:52.663924 0:20:E0:6D:B6:9A -> 0:40:5:2E:DB:D5 type:0x800 len:0x6E  
192.168.0.1 -> 192.168.0.5 ICMP TTL:64 TOS:0xF4 ID:48946 IpLen:20 DgmLen:96  
Type:3 Code:3 DESTINATION UNREACHABLE: PORT UNREACHABLE  
** ORIGINAL DATAGRAM DUMP:  
192.168.0.5 -> 192.168.0.1 ICMP TTL:32 TOS:0x0 ID:8633 IpLen:20 DgmLen:20  
** END OF DUMP  
+++++
```

What is interesting to note from the capture above is three things:

- 1) SnortBox generated the final three packets in response to the first packet, a ping command issued on Pinger. This is shown above by the MAC layer address in bold type, **0:20:E0:6D:B6:9A**. Three packets are generated by the rule option “resp:icmp_all” which crafts and sends ICMP packets to the source host indicating that the HOST, PORT, and NET are all unreachable.
- 2) The spoofed snort packets have a TTL of, 64, which is different from the alerting packet’s TTL of 32.
- 3) Although alerts were generated, the snort rule failed to stop the Target from responding. The second packet recorded is Target’s response to the ECHO request. Notice that all packets are logged with the same time stamp, 19:17:52.663924. This pattern was repeated four times. The default Windows ping command sends four ECHO requests

TELNET

Telnet is a TCP based protocol, which is logging on to a remote system and executing commands. Typically it uses TCP port 23 on the server, and all text is sent in the clear thus all user names and passwords are transferred in clear text making this a very unsafe protocol for remote administration. It is a favorite scanning port for malicious users seeking a vulnerable system hoping to take advantage of a careless administrator. Although a well-defended network might block telnet at the router or firewall, it could still be used between local systems. Once again, a malicious user might take advantage of this, and attack a poorly protected local host.

From a command prompt the command “telnet 192.168.0.1” was issued on Pinger three times. The following snort alert rule was constructed to identify this traffic, and attempt to block it based on packets for Target at port 23 with the ACK flag set.

```
alert tcp any any -> $Target 23 (resp: rst_all; \
  msg:"Forbidden Telnet Attempt=Flex Resp!"; flags: A;)
```

The following is the snort alerts & partial capture of the resulting traffic:

```
[**] [1:0:0] Forbidden Telnet Attempt=Flex Resp! [**]
01/08-15:56:49.664294 0:40:5:2E:DB:D5 -> 0:10:5A:18:71:4F type:0x800 len:0x3C
192.168.0.5:2093 -> 192.168.0.1:23 TCP TTL:128 TOS:0x0 ID:33371 IpLen:20
DgmLen:40 DF
***A**** Seq: 0x91E11AF5 Ack: 0x10622 Win: 0x2238 TcpLen: 20

[**] [1:0:0] Forbidden Telnet Attempt=Flex Resp! [**]
01/08-15:56:59.644294 0:40:5:2E:DB:D5 -> 0:10:5A:18:71:4F type:0x800 len:0x3C
192.168.0.5:2094 -> 192.168.0.1:23 TCP TTL:128 TOS:0x0 ID:33883 IpLen:20
DgmLen:40 DF
***A**** Seq: 0x92082812 Ack: 0x10632 Win: 0x2238 TcpLen: 20

[**] [1:0:0] Forbidden Telnet Attempt=Flex Resp! [**]
```


3) Also the TTL suddenly changes from 128 to 255 in the spoofed packets, and they each use the same ID number 44710. The window size of the spoofed packets is set to 0x0.

4) Pinger sends the third packet of the three-way handshake in the session, and causes the first alert that activates the snort rule. SnortBox appears to succeed in closing the session cleanly if not suddenly with a single crafted packet sent to each host. The end result on Pinger is a Telnet error window reporting, "Connection to Host lost," before the telnet session ever got started. This is achieved by the crafted packet directed at Pinger which arrives with the correct sequence number and the RESET flag set.

FTP

File Transfer Protocol is part of the TCP/IP application protocol suite used for transferring files between two hosts. What is unusual about it, is that it uses two ports rather than one. Separate ports are used for transmission: port 21 for a command channel, and port 20 for data. There are two types of FTP, passive and active FTP. An excellent discussion of the differences is posted at the following URL: <http://www.slacksite.com/other/ftp.html>. Both forms typically use a static port 21 for initial connections and for a command channel on the server to an ephemeral client port. Active FTP uses local port 20 to actively connect to a second ephemeral port on the client for data transfers. This second ephemeral client port is normally one number higher than the first. If a passive FTP server were started on port 21, then the server waits passively for the client to open a second ephemeral port locally for data transfers. In either case server port 21 is the common element.

Hackers have spent long hours tirelessly searching for port 21 on the Internet, and there's a good chance a malicious user wants to find an FTP server on the local network too. A compromised host may have an FTP server running to allow file transfers between local hosts and maybe in and out of the network. A properly configured firewall and/or IDS should alert the administrator to any unauthorized FTP traffic, but what about the unauthorized traffic between hosts? A common policy violation is the abuse of the local network by transferring and storing unauthorized or illegal files. Napster and Gnutella are examples of other applications used to transfer files, but for our test we will use FTP.

From a command prompt the command "ftp 192.168.0.1" was issued on Pinger three times. The following snort alert rule was constructed to identify this traffic, and attempt to block it based on packets accessing ports 20 or 21 with the ACK bit set plus ALL others.

```
alert tcp any any -> any 20:21 (resp:rst_snd; \
  msg: "Illegal FTP Access Flexed!"; flags: A+;)
```

The following is a portion of the snort alerts & partial capture of the resulting traffic:

```
[**] [1:0:0] Illegal FTP Access Flexed! [**]
01/09-10:02:08.768016 0:40:5:2E:DB:D5 -> 0:10:5A:18:71:4F type:0x800 len:0x3C
192.168.0.5:2107 -> 192.168.0.1:21 TCP TTL:128 TOS:0x0 ID:64613 IpLen:20
DgmLen:40 DF
***A**** Seq: 0x5BB3016B Ack: 0x10AAF Win: 0x2238 TcpLen: 20
```

```
01/09-10:02:08.768016 0:40:5:2E:DB:D5 -> 0:10:5A:18:71:4F type:0x800 L
192.168.0.5:2107 -> 192.168.0.1:21 TCP TTL:128 TOS:0x0 ID:64357 IpLen:
DgmLen
*****S* Seq: 0x5BB3016A Ack: 0x0 Win: 0x2000 TcpLen: 24
TCP Options (1) => MSS: 1460
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

01/09-10:02:08.768016 0:10:5A:18:71:4F -> 0:40:5:2E:DB:D5 type:0x800 L
192.168.0.1:21 -> 192.168.0.5:2107 TCP TTL:128 TOS:0x0 ID:51729 IpLen:
DgmLen
***A**S* Seq: 0x10AAE Ack: 0x5BB3016B Win: 0x2238 TcpLen: 24
TCP Options (1) => MSS: 1460
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

01/09-10:02:08.768016 0:40:5:2E:DB:D5 -> 0:10:5A:18:71:4F type:0x800 L
192.168.0.5:2107 -> 192.168.0.1:21 TCP TTL:128 TOS:0x0 ID:64613 IpLen:
DgmLen
***A**** Seq: 0x5BB3016B Ack: 0x10AAF Win: 0x2238 TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

01/09-10:02:08.768016 0:10:5A:18:71:4F -> 0:40:5:2E:DB:D5 type:0x800 L
192.168.0.1:21 -> 192.168.0.5:2107 TCP TTL:128 TOS:0x0 ID:51985 IpLen:
DgmLen
***AP*** Seq: 0x10AAF Ack: 0x5BB3016B Win: 0x2238 TcpLen: 20
32 32 30 20 74 61 72 67 65 74 20 4D 69 63 72 6F 220 target Micro
73 6F 66 74 20 46 54 50 20 53 65 72 76 69 63 65 soft FTP Service
20 28 56 65 72 73 69 6F 6E 20 34 2E 30 29 2E 0D (Version 4.0)..
0A
```

The three-way handshake is completed, and the FTP welcome message is passed from the Target to Pinger. The first RESET packet arrives just after this, and the session is closed. Each time an attempt was made to enter a user name at the prompt the result was a an error message, "Connection closed by remote host." Pinger then opens a new session on a higher port.

1) SnortBox generates the fifth and seventh packets in response to the third packet, which completed the handshake. This is shown above by the MAC layer address in bold type, **0:20:E0:6D:B6:9A**. One packet was generated by each alert and sent to the source host as prescribed by the rule option “resp:rst_snd” which crafts and sends a TCP packet to the sending host with the ACK and RESET flags set.

3) The TTL is 255 in the spoofed packets with a static ID number of 44710. The window size of the spoofed packets is set to 0x0.

5) Pinger sends the sixth packet in the capture with only an ACK flag set causing the second alert to fire. Pinger then processes the resulting RESET packet closing the session.

7) The rule option Flags: was set to “A+”, yet it appears to ignore SYN/ACK packets.

```
alert tcp any any -> any 20:21 (resp:rst_snd; \
    msg: "Illegal FTP AccessFlexed!"; flags: A;)
```

HTTP

Hyper Text Transport Protocol is the bread & butter of traffic on the Internet and most likely the intranet as well. TCP port 80 is the standard server port for all World Wide Web traffic. Studies have found that Internet abuse in the work place is high. Vaunt.com did a survey asking, “during an average workday, how much time do you spend surfing non-work related sites?” They found over 90% of employees spend up to ten minutes or more a day surfing the Internet. Almost 50% reported spending over 30 minutes surfing, and 24.5% reporting spending more than one hour surfing. Every Corporation and University should have a network usage policy in place which restricts certain types of web surfing to non-work related sites, pornography, gambling sites, etc... If the administrator wishes to restrict web surfing at all, clumsy “net nanny” programs must be installed to filter traffic causing more overhead and possibly reducing network performance. Unless all web traffic is blocked via port 80 at the firewall, somebody always manages to circumvent these efforts. But all that goes beyond the discussion of this paper.

Snort gives us a rule option just for this purpose, React. The Snort manual by Martin Roesch states, “the basic reaction is blocking interesting sites users want to access: New York Times, slashdot, or something really important – napster and porn sites.” Although not a perfect solution yet, ... it is still another option.

REACT Rule Option

The following snort alert rule was constructed to identify this traffic, and attempt to block it based on a word content list called “forbidden_words” and port 80 on the Target server.

```
alert tcp any any <> $Target 80 (react: block; \
  msg:"Forbidden Web List Access Attempt=Flex Resp!"; \
  content-list: "forbidden_words");
```

The contents of the file “forbidden_words” is reproduced here:

```
#Forbidden Web Words
"porn"
"testwordAdult"
"sex"
```

The default web page served by IIS4.0 resides at 192.168.0.1, port 80. A duplicate copy of the default web page was setup on Target, but at port 81 for a controlled measure to rule out any extraneous network conditions which might cause a web page to fail to appear or delay its appearance. The http header information in the http code was altered to read “Welcome to IIS 4.0 testwordAdult!” on port 80.

Internet Explorer on Pinger was pointed to <http://192.168.0.1/default.asp>. The refresh button was clicked once, after the first attempt to display the page.

The following is a portion of the snort alerts & partial capture of the resulting traffic. Due to the large number of packets generated during this experiment, only the most significant will be reproduced here chronologically in the order in which they appeared:

```
[**] [1:0:0] Forbidden Web List Access Attempt=Flex Resp! [**]
```

```
01/09-14:39:24.466189 0:10:5A:18:71:4F -> 0:40:5:2E:DB:D5 type:0x800  
len:0x5EA  
192.168.0.1:80 -> 192.168.0.5:2157 TCP TTL:128 TOS:0x0 ID:15638 IpLen:20  
DamLen:1500 DF
```

...abbreviated packet....

```
01/09-14:39:24.466189 0:40:5:2E:DB:D5 -> 0:10:5A:18:71:4F type:0x800 len:0x3C  
192.168.0.5:2157 -> 192.168.0.1:80 TCP TTL:128 TOS:0x0 ID:63849 IpLen:20  
DgmLen:40 DF  
***A**** Seq: 0x539FE944 Ack: 0x30805 Win: 0x2238 TcpLen: 20
```

```
01/09-14:39:24.466189 0:10:5A:18:71:4F -> 0:40:5:2E:DB:D5 type:0x800  
len:0x5EA  
192.168.0.1:80 -> 192.168.0.5:2157 TCP TTL:128 TOS:0x0 ID:15894 IpLen:20  
DgmLen:1500 DF
```

...abbreviated packet....

```
01/09-14:39:24.476189 0:10:5A:18:71:4F -> 0:40:5:2E:DB:D5 type:0x800  
len:0x5EA  
192.168.0.1:80 -> 192.168.0.5:2157 TCP TTL:128 TOS:0x0 ID:16150 IpLen:20  
DgmLen:1500 DF
```

15

...abbreviated packet....

```

...abbreviated packet....
=====

```

```
01/09-14:39:24.486189 0:20:E0:6D:B6:9A -> 0:40:52:E:DB:D5 type:0x800
```

---capture abbreviated---

16

What is happening here? The first attempt to open the web page succeeded yet it still generated an alert as noted above. The bolded text trigger word “testwordAdult”, is plainly seen in the first packet. Pinger responds with one packet and data is transferred from Target to Pinger until the first spoofed packet from SnortBox is seen. SnortBox spoofed the last five packets are directed them at Target. They were found a little further down in the capture.

The alternative web page at <http://192.168.0.1:81/default.asp> appeared normally.

1) SnortBox generated the last six packets in response to the first packet. This is shown above by the MAC layer address in bold type, **0:20:E0:6D:B6:9A**. It is unknown how many packets or of what type are generated by the REACT rule option.

17

3) The spoofed packets have a static ID number of 44710 as before, and the TTL's of all packets are 64, not the expected 255. The window size is now set to 0x400, not 0x0.

4) Two alerts are logged during the capture, but only one stimulating packet can be found. Notice the TTL =255, the type of service is 10 and the packet id of the second alert is 0. This appears to be an anomaly suggesting packet corruption.

5) The first SnortBox packet arrives with a FIN flag and a payload of data, which appears to be some sort of web page with a warning shown in bold. The packet was repeated five times in a row, and again later on in a second block of five.

6) It is strange for data to arrive with a FIN flag and appears to have no effect on the session. When the data is cut out and placed in its own document Figure 2 appears:

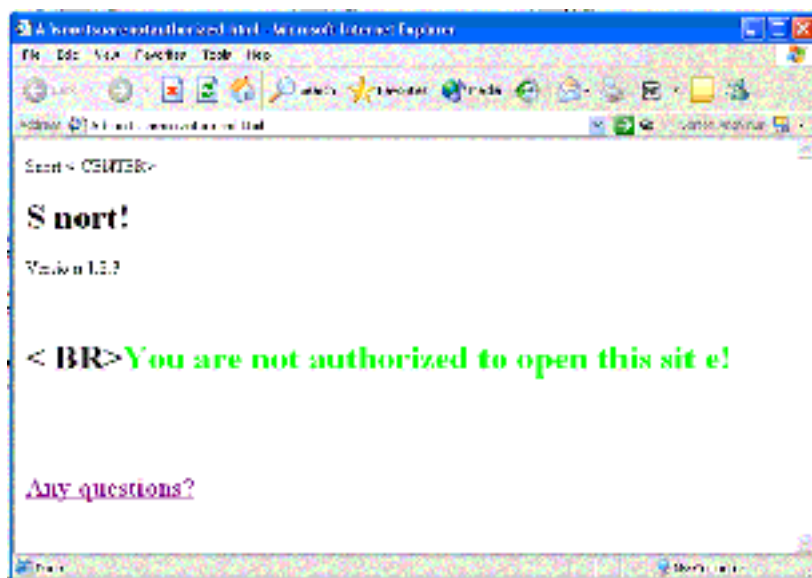


Figure 2

If you click on the link to "Any questions?" it will send email to mszarpak@elka.pw.edu.pl.

7) The next block of spoofed packets shown above with the RESET flag, are attempting to reset the session with Pinger. Notice that each one has a unique acknowledgement number based on the stimulating packet's sequence number. Each packet after the first has the prefix **0x10, 0x20, 0x30, 0x40** added to the base number. This block of five packets appeared in the capture immediately following the Snort FIN packet blocks.

They might be generated by a rule using the ACK+ option since the number "5" seems to correspond to "five" packets from Target with the ACK for ACK+ flags set which preceded these in the capture. Some of Target's packets in question are presented above, but this is not including the original stimulating packet. The sequence number could come from the acknowledgement number of Targets last true packet shown above. They could be attempting a

They are more likely a direct result of the original stimulating packet. Their acknowledgement numbers are based on multiples of the sequence number of the stimulating packet, and they all have the proper sequence number taken for the stimulating packet. With the RESET flag set they appear to be the true attempt to snipe the session.

An alternative to the REACT rule option is using the RESP rule option as shown in the other protocols above. The following snort alert rule was constructed to duplicate the REACT rule using the content rule option.

Internet Explorer on Pinger was pointed to <http://192.168.0.1/default.asp>. The refresh button was clicked twice, after the first attempt to display the page.

```

[**] [1:0:0] Forbidden Webpage Access=Flex Resp! [**]
01/09-14:43:45.136189 0:10:5A:18:71:4F -> 0:40:5:2E:DB:D5 type:0x800
len:0x5EA
192.168.0.1:80 -> 192.168.0.5:2165 TCP TTL:128 TOS:0x0 ID:42006 IpLen:20
DgmLen:1500 DF
***AP*** Seq: 0x302B4 Ack: 0x5787E446 Win: 0x2162 TcpLen: 20

```

...abbreviated packet....

```
01/09-14:43:45.136189 0:40:5:2E:DB:D5 -> 0:10:5A:18:71:4F type:0x800 len:0x3C  
192.168.0.5:2165 -> 192.168.0.1:80 TCP TTL:128 TOS:0x0 ID:24170 IpLen:20  
DgmLen:40 DF
```

```
***A**** Seq: 0x5787E446 Ack: 0x30868 Win: 0x2238 TcpLen: 20
```

...abbreviated packet....

```
01/09-14:43:45.136189 0:10:5A:18:71:4F -> 0:40:5:2E:DB:D5 type:0x800
len:0x5EA
192.168.0.1:80 -> 192.168.0.5:2165 TCP TTL:128 TOS:0x0 ID:42518 IpLen:20
DgmLen:1500 DF
***A*** Seq: 0x30E1C Ack: 0x5787E446 Win: 0x2162 TcpLen: 20
20 20 20 20 3C 2F 63 65 6E 74 65 72 3E 3C 2F 64          </center></d
69 76 3E 0D 0A 3C 64 69 76 20 61 6C 69 67 6E 3D  iv>..
```

...abbreviated packet....

```
01/09-14:43:45.136189 0:10:5A:18:71:4F -> 0:40:5:2E:DB:D5 type:0x800
len:0x1AA
192.168.0.1:80 -> 192.168.0.5:2165 TCP TTL:128 TOS:0x0 ID:42774 IpLen:20
DgmLen:412 DF
***AP*** Seq: 0x313D0 Ack: 0x5787E446 Win: 0x2162 TcpLen: 20
28 63 29 31 39 39 37 20 4D 69 63 72 6F 73 6F 66 (c)1997 Microsof
74 20 43 6F 72 70 6F 72 61 74 69 6F 6E 2E 20 41 t Corporation. A
6C 6C 20 72 69 67 68 74 73 20 72 65 73 65 72 76 ll rights reserv
65 64 2E 0D 0A 09 09 09
```

...abbreviated packet....

```
01/09-14:43:45.136189 0:40:5:2E:DB:D5 -> 0:10:5A:18:71:4F type:0x800 len:0x3C
192.168.0.5:2165 -> 192.168.0.1:80 TCP TTL:128 TOS:0x0 ID:24426 IpLen:20
DgmLen:40 DF
***A*** Seq: 0x5787E446 Ack: 0x31544 Win: 0x2238 TcpLen: 20
+++++
```

```
01/09-14:43:45.146189 0:20:E0:6D:B6:9A -> 0:10:5A:18:71:4F type:0x800
len:0x36
192.168.0.5:2165 -> 192.168.0.1:80 TCP TTL:255 TOS:0x0 ID:44710 IpLen:20
DgmLen:40
***A*R** Seq: 0x5787E446 Ack: 0x30868 Win: 0x0 TcpLen: 20
+++++
```

```
01/09-14:43:45.146189 0:20:E0:6D:B6:9A -> 0:40:5:2E:DB:D5 type:0x800 len:0x36  
192.168.0.1:80 -> 192.168.0.5:2165 TCP TTL:255 TOS:0x0 ID:44710 IpLen:20  
DgmLen:40  
***A*R** Seq: 0x302B4 Ack: 0x5787E9FA Win: 0x0 TcpLen: 20
```

The first packet shown here generated the alert, with the content “testwordAdult”, and the ACK and PUSH flags. SnortBox crafts four spoofed packets, sending two to each host. In testing, this rule seemed to be the most successful at sniping HTTP traffic although the reason is not clear. Approximately 80% of the trials resulted in a 404.1 error page being displayed after a long timeout or when the browser was halted by user intervention. In the remaining trials some portion of the page was transmitted and displayed in the browser, but not the entire page before being timed out or cancelled. In all instances the download time appeared to exceed 10 seconds, which is “the maximum response time before users lose interest,” says Jakob Nielsen, writing for useit.com’s Alertbox in May of 1996, according to “traditional human factor guidelines. Nielson goes on to say that “users have been trained to endure so much suffering” that the limit may be increased to 15 seconds. Snort appears to meet and beat this threshold. Maybe the Internet isn’t all that much faster today?

What is interesting to note from the capture above is 5 things:

- 1) SnortBox generated four packets in response to the first stimulating packet, but not before a good deal of data is transferred to Pinger. This is shown above by the MAC layer address in bold type, **0:20:E0:6D:B6:9A**. Two sets of spoofed packets are generated by the REACT rule option and sent to each host. The second pair is generated with different sequence and acknowledgement numbers.
- 2) The sequence and acknowledgement numbers of the spoofed packets, highlighted in red and blue, are not the same numbers found in the alerting packet. This appears to have been stimulated from the first packet shown above with sequence and acknowledgement numbers shown in green and red.
- 3) The second set of RESET packets appear to have no effect since their sequence and acknowledgement numbers do not relate to any packets in this capture.
- 4) Target appears to have an error in the second to last packet above with TCP port 2165, where the sequence and acknowledgement numbers are the same. The window size is set to 0x0, and a natural RESET flag appears to stimulate a new session with a SYN flag for Pinger on a new port, 2166 in the next packet
- 5) A static ID number of 44710 is present and the TTL's of all spoofed packets are 255, not the expected 128. The window size is once again set to 0x0.

SMB

The final protocol examined is SMB, which stands for Server Messaging Block. SMB is the basis for Microsoft's Windows Networking protocol NetBIOS Extended User Interface (NetBEUI). NetBEUI is the transport layer protocol written by IBM in 1985 to carry NetBIOS API information between computers in a workgroup. NetBIOS uses names up to fifteen characters in length to identify computers on a subnet such as "Pinger" and "Target". Windows95/98, and NT all use SMB to communicate. But Snort can only read IP, ARP, TCP, AND UDP protocols. Fortunately there is NBT or NetBIOS over TCP/IP. This is the standard defined between RFC's 1001 and 1002 which makes it possible for NetBIOS to work over a TCP/UDP network.

"Using Samba," by Robert Eckstein, David Collier-Brown, and Peter Kelly, is a good online beginner's prep to SAMBA, which is the Linux term for SMB. The link is: http://www.oreilly.com/catalog/samba/chapter/book/ch01_03.html. Microsoft's version of SMB is proprietary and not yet published. A good deal of effort by the samba development community has gone into reverse engineering it, but its behavior may not conform to the RFC's.

SMB packets have their own header and there are many "dialects" of the SMB protocol. Fortunately they are all backwards compatible which allows us to extract common signatures from certain SMB packets. SMB communicates via "fire and forget" datagrams and persistent sessions. This test will focus on SMB sessions, particularly Windows File Sharing.

A properly secured windows server will be locked down with the most restrictive NTFS and file sharing permissions based on need for user names and groups. In Windows 2000 it is even possible to restrict file sharing across a network based on the requesting computer's domain account. So why would a conscientious administrator need to worry about a malicious user on the network attempting to reach files without authorization? Well what happens if a green administrator (or a not so green one) makes a change in the permissions and mistakenly gives EVERYONE full access? Or how about those lazy users who open up sharing to everyone out of convenience? Unprotected Windows network sharing is listed in the SANS Top20 Most Critical Internet Security Vulnerabilities right here: <http://sans.org/top20.htm>. Here come those malicious users again...

From Pinger's desktop the Network Neighborhood icon was opened to reveal the entries of the local workgroup "Snort" which consisted of Pinger and Target. The icon for Target was double clicked to open it. A few seconds later, the run command `\\192.168.0.1\NTShare` was executed.

The following snort alert rule was constructed to identify this traffic, and attempt to block it based on any IP address contacting 192.168.0.1 on port 139 with the ACK and PUSH flags set.

```
alert tcp any any -> 192.168.0.1 139 (resp: rst_snd; flags: AP; \
  msg: "Bash 'em down! Illegal access, AP flag.");)
```

The following is a portion of the snort alerts & partial capture of the resulting traffic. Due to the large number of packets generated during this experiment, only a few of the most significant will be reproduced here. Packets may appear out of sequence as noted:

```
[**] [1:0:0] Bash 'em down! Illegal access, AP flag. [**]
01/04-18:53:54.513924 192.168.0.5:1576 -> 192.168.0.1:139
TCP TTL:128 TOS:0x0 ID:3361 IpLen:20 DgmLen:144 DF
***AP*** Seq: 0x10D39276 Ack: 0x16B489 Win: 0x2149 TcpLen: 20
```

```
[**] [1:0:0] Bash 'em down! Illegal access, AP flag. [**]
01/04-18:53:54.523924 192.168.0.5:1601 -> 192.168.0.1:139
TCP TTL:128 TOS:0x0 ID:4385 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x147B9763 Ack: 0x16B4C8 Win: 0x2238 TcpLen: 20
```

```
[**] [1:0:0] Bash 'em down! Illegal access, AP flag. [**]
01/04-18:53:54.523924 192.168.0.5:1601 -> 192.168.0.1:139
TCP TTL:128 TOS:0x0 ID:4641 IpLen:20 DgmLen:214 DF
***AP*** Seq: 0x147B97AB Ack: 0x16B4CC Win: 0x2234 TcpLen: 20
```

```
[**] [1:0:0] Bash 'em down! Illegal access, AP flag. [**]
01/04-18:53:54.523924 192.168.0.5:1601 -> 192.168.0.1:139
TCP TTL:128 TOS:0x0 ID:4897 IpLen:20 DgmLen:296 DF
***AP*** Seq: 0x147B9859 Ack: 0x16B529 Win: 0x21D7 TcpLen: 20
```

```
01/04-18:53:54.513924 0:40:5:2E:DB:D5 -> 0:10:5A:18:71:4F type:0x800 len:0x9E
192.168.0.5:1576 -> 192.168.0.1:139 TCP TTL:128 TOS:0x0 ID:3361 IpLen:20
DgmLen:144 DF
***AP*** Seq: 0x10D39276 Ack: 0x16B489 Win: 0x2149 TcpLen: 20
00 00 00 64 FF 53 4D 42 A2 00 00 00 00 18 03 80 ...d.SMB.....
6A 80 00 00 00 00 00 00 00 00 00 00 00 88 00 8E j.....
01 08 40 00 18 FF 00 00 00 00 0E 00 06 00 00 00 ..@.....
```


---Three Way Handshake Completes Here---

24

What is interesting to note from the capture above is six things:

- 1) SnortBox generated the second and last packets. This is shown above by the MAC layer address in bold type, **0:20:E0:6D:B6:9A**. The first one appears to be in response to the first packet shown above.
- 2) The sequence number of the first spoofed packet is highlighted in red as is the acknowledgement number of the stimulating packet. The spoofed acknowledgement number, colored in blue, appears out of sequence but is repeated in the next packet in the capture.
- 3) Clearly the first crafted packet arrives at Pinger before Target's packet with the same sequence and acknowledgement numbers.
- 4) Pinger appears to reply to the crafted packet with a natural RESET of its own, closing the session on port 1576. This packet also has identical sequence and acknowledgement numbers. Pinger then it starts a new session in the next packet on port 1601 with a SYN flag. Pinger repeats this pattern of starting new session on incrementally increasing port numbers as shown above out of context.
- 5) The second crafted packet did not appear to be associated with any alerts. It does share common values with two preceding packets however. It is unclear what stimulated this packet.
- 6) The TTL remains 255 in the spoofed packets with a static ID number of 44710. The window size of the spoofed packets is set to 0x0.

The snort rule tested here for SMB packets:

```
alert tcp any any -> 192.168.0.1 139 (resp: rst_snd; flags: AP; \
  msg: "Bash 'em down! Illegal access, AP flag.");)
```

was 100% effective in preventing Pinger from opening any shared resources on Target. For fun, while testing this rule Target attempted to open shares on Pinger, and Pinger "pinged" Target. Both of these commands completed normally and network performance did not appear adversely affected by the snort reactor.

Another interesting rule is designed to block a single shared folder on Target while allowing unrestricted access to other shares:

```
alert tcp any any <> any 137:139 (resp: rst_snd; \
  msg: "Illegal Shared Folder Access=Flexed!"; flags: AP; \
  content: "|540053006800610072006500|";)
```

This rule generates alerts based on SMB traffic between anyone on the common NetBIOS ports, flags: AP, and with a hex equivalent payload of **N.T.S.h.a.r.e**. It attempts to close the TCP session, but so far has only managed to delay the share window from opening by 5 to 20 seconds on average, and almost two minutes in one extreme example.

SUMMARY AND DISCUSSION

Snort flexible response rules were tested for five common protocols; ICMP, TELNET, FTP, HTTP, and SMB on a simple network between Windows NT hosts. Specific rules were tailored to snipe or close a network session carried over each of these protocols. With the lone exception of ICMP all protocols were demonstrated to be susceptible to a denial of service attack by snort's crafted packets spoofing the hosts involved.

Windows' ICMP ping packets may not be susceptible to this type of attack in this test scenario due to the close timing of packets sent, and the operating system's apparent lack of response to the ICMP error packets. Further testing is required to rule out these possibilities in a larger network where more network latency may become a variable. Also this rule should be tested between more host operating systems.

Telnet and FTP traffic appeared to be highly vulnerable to snort's tactics perhaps due to their sequential packet delivery architecture. This suggests that any symmetric protocol based on TCP packets flowing in both directions may also be susceptible to similar attacks. Future experiments could focus on rule sets design to alert on ASCII characters in the username or passwords. Also rule headers which deny service based on internal networks, external networks, or allowing service based on specific IP's only are valid test variables.

Web traffic based on HTTP, which is asymmetrical, appears to be a little less vulnerable to hijacking. It appears the author has made an ineffectual attempt insert data into the crafted packets. It was extremely easy to pick on a web page with completely predictable content. But trying to shoot down a web session based on random word lists could prove much more unwieldy and resource intensive beyond the point of futility. However as a punitive measure for a known offending user, a remote administrator could craft rules to block all port 80 traffic from a chronic offender's IP address, or subnet based on port or web address IP. This avoids having to reconfigure routers or encumbering firewalls with more rules. This sanction would have to be imposed by a reactor on the local subnet, however, since a properly configured firewall or router with egress filtering would block remotely spoofed packets. REACT rules with expanded wordlists are worth examining in a noisier environment to test snort's ability to handle this traffic with a more realistic load. RESP rules may provide the most reliable option of sniping sessions for a specific packet signature.

Windows File Sharing traffic was the most challenging protocol this paper attempted to snipe until a rule was crafted to deny service based on IP addresses. Considerable effort was spent searching for an effective rule to deny traffic based on a share name. Many early attempts managed to flood the network and did not block access to the share. At best the share was delayed from opening. Further research and insight into the SMB protocol may yield better results. More network latency might also improve snort's effectiveness to this end.

Further testing should be performed on other types of traffic considered undesirable to networks based on policy. Effective rule sets for UDP traffic, for instance, might provide a useful tool in combating excessive streaming media content or instant messaging at the office. Online games are another form of unnecessary and unproductive traffic on the network. All of these things can

decrease worker productivity, and encumber the network, decreasing the network's ability to transport legitimate business traffic.

This paper should be viewed from the perspective that it is only a preliminary experiment into "reactor" intrusion detection systems. The results presented here may not be typical. The results are inherently limited by the unrealistic size, and simplicity of the test network. A larger network with more background traffic, populated by more varied hosts, and multiple network segments may yield more accurate results applicable to a production network.

Until the perfect IDS appliance is invented, vigilant analysts will have to continue to guard the electronic frontier. But snort can help keep watch inside the fence using the techniques demonstrated here. An automated security policy enforcement tool might come in handy. Especially when the offender and his favorite "hideouts" are known. Given the demanding job of running a network, reading endless logs, tracing down known security incidents, potential intrusions, and handling internal security incidents, this option becomes more and more appealing.

"It is estimated that 80% of attacks are from within the organization," writes Terry Boston, author of the article "The Insider Threat." The malicious user is forever on the look out for new opportunities. The bottom line is security in layers. Ideally the network is defended by: a comprehensive security policy, firewalls, intrusion detection systems, proxy servers, and routers. Internal hosts should be protected by above all safe and sound user practices according to policy, and also: anti-virus programs, file permissions, limited services, security policy accounts managed by products such as Windows 2000 Active Directory, and perhaps personal firewalls and port sentries such as tcpwrappers. But when these things fail, there's still the IDS out there watching...

The answer is simple: security in layers. Why not have a layer which is "on the wire"? Intrusion detection system's have been watching the wire for some time, but couldn't do anything about what was on it until now.

REFERENCES

Boston, Terry. "The Insider Threat."

URL: http://rr.sans.org/securitybasics/insider_threat2.php (Oct. 2000).

Connors, Matthew. "System Policies and the System Policy Editor (SPE) for Microsoft Windows."

URL: <http://rr.sans.org/win/SPE.php> (Feb. 2001).

Eckstein, Robert. Collier-Brown, David. Kelly, Peter. "Using Samba."

URL: http://www.oreilly.com/catalog/samba/chapter/book/ch01_03.html (Nov. 1999).

JANET-CERT. "Protecting People."

URL: <http://www.ja.net/CERT/JANET-CERT/prevention/people.html> (Jan. 2002).

Kipp, Steve. "Espionage and the Insider."

URL: <http://rr.sans.org/securitybasics/espionage.php> (July. 2001).

Labadie, Nathan W., Email: "Slowing down the spread of worms."

URL: <http://archives.neohapsis.com/archives/incidents/2001-09/0497.html> (Sept. 2001).

Lademann, Christian. "Readme.flexresp" snort for windows distribution 1.8.3.

URL: <http://www.snort.org/packages.html> (Jan. 2002).

Nielsen, Jakob. "Top Ten Mistakes in Web Design."

URL: <http://www.useit.com/alertbox/9605.html> (May, 1996).

Nielsen, Jakob. "Response Times: The Three Important Limits," an excerpt from Chapter 5 of *Usability Engineering*, Published by Morgan Kaufmann, San Francisco 1994

URL: <http://useit.com/papers/responsetime.html> (Jan. 2002).

RFC 1001 – "Protocol Standard For A NetBIOS Service on a TCP/UDP Transport: Concepts and Methods."

URL: <ftp://ftp.isi.edu/in-notes/rfc1001.txt> (Mar. 1987).

Roesch, Martin. "Snort Users Manual Snort Release: 1.8.3."

URL: http://www.snort.org/docs/writing_rules/ (Nov. 2001).

SANS.com Reading Room:

URL: http://rr.sans.org/intrusion/intrusion_list.php (Jan. 2002).

SANS.com Resources. "The Twenty Most Critical Internet Security Vulnerabilities (Updated) The Experts' Consensus Version 2.501."

URL: <http://www.sans.org/top20.htm> (Nov. 2001).

Slacksite.com "Active FTP vs. Passive FTP, a Definitive Explanation."

URL: <http://www.slacksite.com/other/ftp.html> (Jan. 2002).

Snort Forum at Neohapsis:

URL: <http://archives.neohapsis.com/archives/snort/> (Jan. 2002).

Vault.com. "Internet Use Survey of 451 Employees."

URL: <http://www.vault.com/surveys/internetuse2000/results2000.jsp?results=2&image=employee> (Fall. 2000).

APPENDIX

1-A SNORT Configuration File

```
#-----  
# http://www.snort.org      Snort 1.8.0 Ruleset  
# Contact: snort-sigs@lists.sourceforge.net
```

```

#-----
# NOTE:This ruleset only works for 1.8.0 and later
#-----
# $Id: snort.conf,v 1.30 2001/04/20 03:43:51 cazz Exp $
#
#####
# Step #1: Set the network variables:
# var HOME_NET $eth0_ADDRESS
# Set up your web servers, or simply configure them
# to HOME_NET
var HTTP_SERVERS 192.168.0.1
var Target $(Target:-192.168.0.1)
var Pinger $(Pinger:-192.168.0.5)
#####
# Step #2: Configure preprocessors
preprocessor defrag
# stream/stream2: TCP stream reassembly
preprocessor stream2: timeout 10, ports 21 23 80 110 143, maxbytes 16384
# http_decode: normalize HTTP requests
preprocessor http_decode: 80 -unicode -cginull
# portscan: detect a variety of portscans
preprocessor portscan: $HOME_NET 4 3 portscan.log
#-----
# FTP RULES last modified 01/09/02
#-----
#alert tcp any any <> any 20:21 (resp:rst_all; msg: "Illegal FTP Access
Flexed!"; flags: A+; content: "testwordAdult");
#alert tcp any any -> any 20:21 (resp:rst_snd; msg: "Illegal FTP Access
Flexed!"; flags: A+;)
alert tcp any any -> any 20:21 (resp:rst_snd; msg: "Illegal FTP Access
Flexed!"; flags: A;)
#-----
# ICMP RULES
#-----
alert icmp any any -> $Target any (resp:icmp_all; msg: "Pinger is calling,
Hang Up!");
#-----
# NETBIOS RULES
#-----
#WINDOWS FILE SHARING RULES
#-----
#Proven GCIA RULES
#Modified Last 12/28/01
#alert tcp any any <> any 139 (msg: "Illegal Shared Folder Access"; flags:
AP; content: "|540053006800610072006500|");
#alert tcp any any <> any 139 (resp: rst_all,icmp_port; msg: "Illegal Shared
Folder Access=Flexed!"; flags: AP; content: "|540053006800610072006500|");
#alert tcp any any <> any 139 (resp: rst_all,icmp_host; msg: "Illegal Shared
Folder Access=Flexed!"; flags: AP; content: "|540053006800610072006500|");
#alert tcp any any <> any 137:139 (resp: rst_all,icmp_host; msg: "Illegal
Shared Folder Access=Flexed!"; flags: AP; content:
"|540053006800610072006500|");
#alert tcp any any <> any 137:139 (react: block; msg: "Illegal Shared Folder
Access=Flexed!"; flags: AP; content: "|540053006800610072006500|");
#alert tcp any any <> any 137:139 (resp: rst_rcv,icmp_port; msg: "Illegal
Shared Folder Access=Flexed!"; flags: AP; content: "|00413a00|");

```

```

#alert tcp any any <> any 137:139 (resp: rst_snd; msg: "Illegal Shared Folder
Access=Flexed!"; flags: AP; content: "|540053006800610072006500|");
#alert tcp 192.168.0.5 any -> 192.168.0.1 139 (resp: rst_all; msg: "Bash 'em
down! Illegal access.");)
#alert tcp 192.168.0.5 any -> 192.168.0.1 139 (resp: rst_all; flags: A+;
msg: "Bash 'em down! Illegal access.");)
#alert tcp 192.168.0.5 any -> 192.168.0.1 139 (resp: rst_all; flags: AP;
msg: "Bash 'em down! Illegal access, AP flag.");)
#alert tcp any any -> 192.168.0.1 139 (resp: rst_all; flags: AP; msg: "Bash
'em down! Illegal access, AP flag.");)
#Best rule to date-----
-----
#alert tcp any any -> 192.168.0.1 139 (resp: rst_snd; flags: AP; msg: "Bash
'em down! Illegal access, AP flag.");)
#-----
-----
#-----
#activate tcp any any <> any 137:139 (resp: rst_snd; msg: "Illegal Shared
Folder Access=Flexed!"; flags: AP; content: "|540053006800610072006500|";
activates: 1;)
#dynamic tcp any any <> any 137:139 (activated_by: 1; count: 50; resp:
rst_snd; msg: "Illegal Shared Folder Access=Flexed!"; flags: AP; content:
"|540053006800610072006500|");)
#-----
#----- see snort0104@1717.log
#activate tcp any any <> any 137:139 (resp: rst_snd; msg: "Illegal Shared
Folder Access=Flexed!"; flags: AP; content: "|540053006800610072006500|";
activates: 1;)
#dynamic tcp any 137:139 <> any any (activated_by: 1; count: 50; resp:
rst_rcv; msg: "Illegal Shared Folder Access=Flexed Again!");)
#-----
#-----
#activate tcp any any -> any 137:139 (resp: rst_snd; msg: "Illegal Shared
Folder Access=Flexed!"; flags: AP; content: "|540053006800610072006500|";
activates: 1;)
#dynamic tcp any any <- any 137:139 (activated_by: 1; count: 5; resp:
rst_rcv; msg: "Illegal Shared Folder Access=Flexed Again!");)
alert tcp any any -> 192.168.0.1 139 (resp: rst_snd; flags: AP; msg: "Bash
'em down! Illegal access, AP flag.");)
#-----
# TELNET RULES
#-----
#Proven GCIA RULES
#Modified Last: 01/08/02
#TELNET Port 23
#alert tcp any any <> $Target 23 (resp: rst_all; msg:"Forbidden Telnet
Attempt=Flex Resp!"; flags: A+;)
alert tcp any any -> $Target 23 (resp: rst_all; msg:"Forbidden Telnet
Attempt=Flex Resp!"; flags: A;)
#Proven GCIA RULES
#Modified Last: 01/09/02
#-----
#HTTP
#-----
#alert tcp $Pinger any <> $Target 80 (msg:"Forbidden Webpage Acess"; flags:
A+; content: "welcome"; nocase;)

```



```

#alert tcp any any <> $Target 80 (msg:"Forbidden Webpage Acss";flags: A+;
content: "welcome"; nocase;)
#alert tcp any any <> $Target any (msg:"Forbidden Webpage Acss";flags: A+;
content: "welcome"; nocase;)
#alert tcp any any <> $Target any (msg:"Forbidden Webpage Acss";flags: A+;
content: "testwordAdult"; nocase;)
#alert tcp any any <> $Target any (resp: rst_all; msg:"Forbidden Webpage
Acss=Flex Resp!";flags: A+; content: "testwordAdult"; nocase;)
#alert tcp any any <> any any (resp: rst_all; msg:"Forbidden Webpage
Acss=Flex Resp!";flags: A+; content: "testwordAdult"; nocase;)
#end
alert tcp any any <> any any (resp: rst_all; msg:"Forbidden Webpage
Acss=Flex Resp!";flags: A+; content: "testwordAdult"; nocase;)
#alert tcp any any <> $Target 80 (react: block; msg:"Forbiden Web List Access
Attempt=Flex Resp!"; content-list: "forbidden_words");)
log ip any any -> any any (msg:"Packet Log");)

```

ASSIGNMENT 2- NETWORK DETECTS

Detect-1 Comcast@Home Cable Modem IGMP Broadcast

1. Source of Trace

This trace was picked up on a home network using [Comcast@Home](#) as a high-speed cable modem service provider which is assigned a DHCP address by Comcast.

2. Detect was Generated by

This trace was generated by a Tiny Software WinRoutePro Firewall security log on the “Comcast” interface of a multi-homed Windows 2000 Server which is connected directly to the Comcast@Home rented cable modem.

This particular log format uses: the date and time, the packet filtering rule number which activated in the order it was applied, the name of the interface, what action occurred, the identified protocol, and finally the source and destination IPs. If ports were involved they would have been listed following the IP address.

```
[28/Dec/2001 08:55:11] Packet filter: ACL 4:6 Comcast: drop packet in: Protocol 2, 192.168.100.1 -> 224.0.0.1
```

```
[28/Dec/2001 08:58:11] Packet filter: ACL 4:6 Comcast: drop packet in: Protocol 2, 192.168.100.1 -> 224.0.0.1
```

```
[28/Dec/2001 09:01:11] Packet filter: ACL 4:6 Comcast: drop packet in: Protocol 2, 192.168.100.1 -> 224.0.0.1
```

```
[28/Dec/2001 09:04:11] Packet filter: ACL 4:6 Comcast: drop packet in: Protocol 2, 192.168.100.1 -> 224.0.0.1
```

3. Probability the Source Address was Spoofed

192.168.100.1 is an invalid IP on the Internet, which should be blocked by routers on the Internet and by Comcast, since they have a reserved IP address range for customers on the Internet. Therefore the likelihood of spoofing is low. However it is possible a single local host, the cable modem itself, is sending the packets. The destination IP is the broadcast address for all multicast hosts 224.0.0.1, which “is assigned to the permanent group of all IP hosts (including gateways). This is used to address all multicast hosts on the directly connected network.” (RFC 1112, p.3)

A “Whois” command returned the following:

```
IANA (NET-MCAST-NET)
  Internet Assigned Numbers Authority
  4676 Admiralty Way, Suite 330
  Marina del Rey, CA 90292-6695
  US

  Netname: MCAST-NET
  Netblock: 224.0.0.0 - 239.255.255.255
```

Coordinator:
Internet Corporation for Assigned Names and Numbers (IANA-ARIN) res-
ip@iana.org
(310) 823-9358

4. Description of Attack

Packets from the 192.168.0.0 reserved class C range appear to be coming from 192.168.0.1 and are directed at the class D multicast broadcast address 224.0.0.1 approximately every three minutes. The log states the packets consisted of protocol 2 and were dropped. This network does not use the 192.168.0.0. class C address range, and the packets are being received by the external routing interface of the firewall. Thus it is unlikely that an internal host is the source. These packets have been reported in the network logs since November of 2001, when tracking began, and have no known internal stimulus.

RFC 2236 states, "IGMP messages are encapsulated in IP datagrams, with an IP protocol number of 2. All IGMP messages described in this document are sent with IP TTL 1..." (RFC 2236, p.2) Multicast routers are expected to query the attached network (the local network), periodically with a general query to solicit membership information. "A General Query is addressed to the all-systems multicast group (224.0.0.1)." (RFC 2236, p.4) The pattern is timed on a three-minute interval, which supports the notion of a periodic query. IGMP is a Network layer protocol, which does not use ports that is also consistent with the WinRoutePro logs.

Information from the correlation section below will support the theory that the cable modem, which is believed to be a routing device, is broadcasting IGMP general queries looking for all multicast group members on the local network. Information in the correlation section will also suggest that detect could be a crafted packet carrying the RC1 Trojan, however the preponderance of evidence does not support this.

A Common Vulnerability & Exposure (CVE) search turned up no directly related CVE numbers, however there were a couple relevant entries concerning IGMP traffic. CVE-1999-0918 reports a malformed, fragmented IGMP packet can cause a remote Denial of Service in some Windows platforms. CAN-2001-0796 reports there is a possible remote Denial of service via malformed IGMP multicast packets which only affects FreeBSD 3.0 and some versions of SGI IRIX.

5. Attack Mechanism

Unknown. This is unlikely an attack and more likely a function of IGMP attempting to reach all local hosts on the network. Since the firewall is configured with a "deny all" incoming filter policy, the 192.168.0.0 subnet is denied entry into the network. The local hosts have not been exposed to these packets, so it is unknown how they would respond.

This detect appears to be originating from the cable modem itself since the 192.168.0.1 address is an illegal Internet address, and the outward facing firewall interface is encountering the packets. The cable modem is also a routing device, which may be RFC 2236 compliant. It could then be periodically checking the local network for multicast hosts with the 224.0.0.1 broadcast. [Comcast@Home](#) provides rental modems to its subscribers. It is conceivable that only a small

portion of the modems which Comcast uses generate this traffic, since there are so few reports and the similarity of two MAC address's identified below is very striking.

Assuming this were an attack and not the modem, then it may be a Smurf style “ping” for multicast IP's. If a multicast group member were located, they might respond to the local 192.168.0.1 address. If enough packets were received on the local network segment, stimulating enough responses directed at the local address, a localized denial of service for the host 192.168.0.1, and a possible slow down in the network performance would result.

It could also be a masqueraded attempt to use an untargeted exploit over IGMP as indicated by the last detect in the correlation section, however this is unlikely, given there is no evidence of a malicious payload in the log file.

6. Correlations

There were four reported instances of a similar trace found at www.google.com with this search string “Protocol 2, 192.168.100.1 -> 224.0.0.1”. Two of the reports confirm the use of cable modems and one mentions Comcast specifically, which suggests the trace is connected to the ISP. Two of the traces also report the use of port 65535, which associated with the RC1 Trojan.

Jim Nanney reports that he uses a cable modem and has seen a very similar log entry. His log confirms the protocol number with PROTO=2, and a TTL=1, and adds new information about the possible ports in use. The RFC does not refer to the use of ports. Assuming this traffic was in fact IGMP general queries, then, this log may be incorrectly identifying ports by reporting 65535. Nanney's trace is shown below, and the complete email thread can be seen here: <http://archives.neohapsis.com/archives/incidents/2001-02/0287.html>

```
I use a cable modem service so I do not believe the attack is aimed at
my
box, but it has some strange things in it and I was looking for
verification
of what it is.
Here is the logs:
Feb 21 09:54:32 nanlinux kernel: Packet log: input REJECT eth0 PROTO=2
192.168.100.1:65535 224.0.0.1:
L=28 S=0xC0 I=0 F=0x0000 T=1 (#5)
Feb 21 09:57:32 nanlinux kernel: Packet log: input REJECT eth0 PROTO=2
192.168.100.1:65535 224.0.0.1:65535 L=28 S=0xC0 I=0 F=0x0000 T=1 (#5)
Feb 21 10:00:32 nanlinux kernel: Packet log: input REJECT eth0 PROTO=2
192.168.100.1:65535 224.0.0.1:65535 L=28 S=0xC0 I=0 F=0x0000 T=1 (#5)
Feb 21 10:03:32 nanlinux kernel: Packet log: input REJECT eth0 PROTO=2
192.168.100.1:65535 224.0.0.1:65535 L=28 S=0xC0 I=0 F=0x0000 T=1 (#5)
Feb 21 10:06:32 nanlinux kernel: Packet log: input REJECT eth0 PROTO=2
192.168.100.1:65535 224.0.0.1:65535 L=28 S=0xC0 I=0 F=0x0000 T=1 (#5)
```

A second trace posted in a foreign language reveals that this detect is related to it, and its causing concern for the author, Peter Gade Jensen. The complete email thread appears here and the trace is reproduced below: http://www.sslug.dk/emailarkiv/teknik/2000_10/msg01278.html

Torben Fjordingstad's response below suggests that PROTO=2 is equated with Internet Group Management Protocol (IGMP) which correlates with RFC 2236 and this log shows 224.0.0.1 is the address in question with a TTL=1 which also supports Fjordingstad.

```
>IN=eth0 OUT= MAC=01:00:5e:00:00:01:00:20:40:62:7a:79:08:00  
>SRC=192.168.100.1 > DST=224.0.0.1 LEN=28 TOS=0x00 PREC=0xC0 >TTL=1  
ID=0 PROTO=2
```

Proto 2 er igmp (Internet Group Management Protocol).
Der er en maskine på nettet der sender multicast. Det bruges f.eks. i
mbone. 224.0.0.1 er en multicast adresse.

A post by Wieland Gmeiner reports that he has logged similar detects, and he also has Comcast cable modem service. Part of Gmeiner's log is reproduced here, and his email thread is located at: <http://webmail.cotse.com/nix/susesecurity/lists/susesec/current/0124.html>

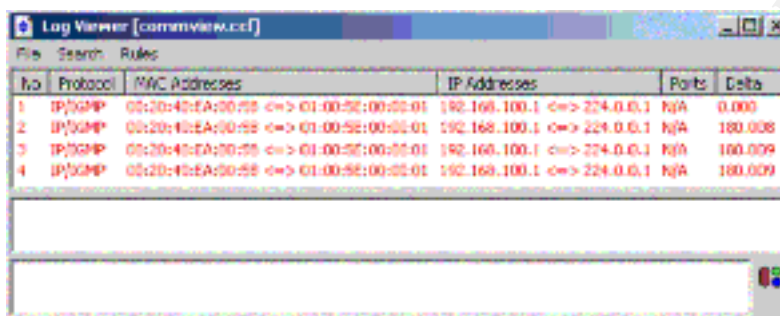
```
I can't figure out the meaning of entries in /var/log/firewall : [...]  
Dec 5 00:20:07 w1 kernel: SuSE-FW-UNALLOWED-TARGETIN=eth0 OUT=  
MAC=01:00:5e:00:00:01:00:20:40:e5:b4:35:08:00 SRC=192.168.100.1  
DST=224.0.0.1 LEN=28 TOS=0x00 PREC=0xC0 TTL=1 ID=0 PROTO=2 Dec 5  
00:23:07 w1 kernel: SuSE-FW-UNALLOWED-TARGETIN=eth0 OUT=  
MAC=01:00:5e:00:00:01:00:20:40:e5:b4:35:08:00 SRC=192.168.100.1  
DST=224.0.0.1 LEN=28 TOS=0x00 PREC=0xC0 TTL=1 ID=0 PROTO=2 Dec 5  
00:26:07 w1 kernel: SuSE-FW-UNALLOWED-TARGETIN=eth0 OUT=  
MAC=01:00:5e:00:00:01:00:20:40:e5:b4:35:08:00 SRC=192.168.100.1  
DST=224.0.0.1 LEN=28 TOS=0x00 PREC=0xC0 TTL=1 ID=0 PROTO=2 Dec 5  
00:29:07 w1 kernel: SuSE-FW-UNALLOWED-TARGETIN=eth0 OUT=  
MAC=01:00:5e:00:00:01:00:20:40:e5:b4:35:08:00 SRC=192.168.100.1  
DST=224.0.0.1 LEN=28 TOS=0x00 PREC=0xC0 TTL=1 ID=0 PROTO=2 Dec 5
```

This suggests that more Comcast@Home subscribers may be receiving this traffic. Given the low number of reports, and the fact that two of the four reports are known to be Comcast subscribers, suggests the possibility that the source host, 192.168.0.1 is operated by Comcast@Home. Which is of course assuming that 192.168.0.0 traffic is permitted inside the Comcast network.

Notice the strange MAC address notations in Jensen and Gmeiner's logs, highlighted in red and blue. If the first portion represents a six byte MAC broadcast address, and the 00 byte is a delimiter between the destination and source MACs, then the first two bytes of Jensen and Gmeiner's addresses, 20:40, strongly suggests the modems are related products made by the same manufacturer, probably for the same purpose. Johannes Geiger responds to Gmeiner that the cable modem itself is the source of this traffic. "Comcast," Geiger writes, "came out about 2 weeks ago and switched my cable modem and this garbage started with me. But I can't seem to get iptables to not log that entry, since it just is filling up the log file."

The MAC address of the cable modem in the network which recorded this detect just happens to be **00:20:40:EA:00:5B**. This fact leads to a conclusion that all of these modems are of the same make, and probably performing the same routine function.

Commview 3.2 is a network sniffer by Tamosoft, which is capable of decoding the IGMP protocol. It was eventually installed on the firewall host some weeks after this detect was originally recorded to capture the following packet log file and discover the true nature of these packets. This figure shows that the IGMP protocol is clearly in use sending broadcasts from the IP 192.168.100.1, with a MAC address of 00:20:40:EA:00:5B to the Multicast broadcast address 224.0.0.1 with a broadcast MAC address of **01:00:5E:00:00:01**. This evidence would seem to resolve the matter completely.



No	Protocol	MAC Addresses	IP Addresses	Ports	Delta
1	IP/IGMP	00:20:40:EA:00:5B <=> 01:00:5E:00:00:01	192.168.100.1 <=> 224.0.0.1	N/A	0.000
2	IP/IGMP	00:20:40:EA:00:5B <=> 01:00:5E:00:00:01	192.168.100.1 <=> 224.0.0.1	N/A	180.008
3	IP/IGMP	00:20:40:EA:00:5B <=> 01:00:5E:00:00:01	192.168.100.1 <=> 224.0.0.1	N/A	180.009
4	IP/IGMP	00:20:40:EA:00:5B <=> 01:00:5E:00:00:01	192.168.100.1 <=> 224.0.0.1	N/A	180.009

Figure 3

Some Comcast@Home modems with a 00:20:40:xx:xx:xx MAC are multicast routing devices which are also RFC 2236 compliant and scan the local network every three minutes with a multicast discovery packet. However, the last link still casts some doubt into the seemingly benign nature of this detect.

This final link suggests the most disturbing interpretation of a similar detect by Scott Sawyer. Sawyer's detect is reproduced below with an abbreviated payload, which has the typical 192.168.0.1 source IP, 224.0.0.1 destination IP, is identified as protocol 2, and has a TTL=1, is attempting a buffer overflow exploit! Note the sbn/rpc.statd as pointed out by Curtis Zinzilieta in a follow-up post. The link to the complete message thread is found here: <http://www.infomagic.net/pipermail/luna/2001-May/000285.html>

It is also using the port 65535, which in this case, fits the profile of the RC1 Trojan as posted here: <http://www.blackcode.com/trojans/details.php?id=1073> and it is carrying a payload. However no further information regarding the nature of the RC1 Trojan could be gleamed from a web search.

This packet does not conform to pattern of the detects above since it carries a payload, and that payload has dubious intentions. It is likely that this packet was crafted to spoof IGMP traffic since it otherwise conforms to the RFCs, but appears to have malicious content. Its similarity to the other detects, suggests the author of the packet had knowledge of the IGMP general query and may have tried to imitate it. If so, then the attacker must be located on the local subnet or he got very lucky with a carefully crafted TTL of 1 upon arriving at Sawyer's host.

```
May 26 20:08:01 fw /USR/SBIN/CRON[3634]: (mail) CMD ( if [ -x
/usr/sbin/exim -a
-f /etc/exim.conf ]; then /usr/sbin/exim -q >/dev/null 2>&1; fi)
```

--Contents Abbreviated--


```

01/17-03:39:05.765254 A.B.164.119:8080 -> 202.99.41.15:64695
TCP TTL:128 TOS:0x0 ID:33838 IpLen:20 DgmLen:40
***A**R** Seq: 0x0 Ack: 0xBBD9A31 Win: 0x0 TcpLen: 20
=====
01/17-03:39:05.765710 202.99.41.15:64696 -> A.B.164.119:80
TCP TTL:109 TOS:0x0 ID:36510 IpLen:20 DgmLen:48 DF
*****S* Seq: 0xBBD9A31 Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=====
01/17-03:39:05.765844 A.B.164.119:80 -> 202.99.41.15:64696
TCP TTL:128 TOS:0x0 ID:34094 IpLen:20 DgmLen:44 DF
***A**S* Seq: 0x16EC79 Ack: 0xBBD9A32 Win: 0x2238 TcpLen: 24
TCP Options (1) => MSS: 1460
=====
01/17-03:39:08.683493 A.B.164.119:80 -> 202.99.41.15:64696
TCP TTL:128 TOS:0x0 ID:34350 IpLen:20 DgmLen:44 DF
***A**S* Seq: 0x16EC79 Ack: 0xBBD9A32 Win: 0x2238 TcpLen: 24
TCP Options (1) => MSS: 1460
=====
01/17-03:39:08.718642 202.99.41.15:64696 -> A.B.164.119:80
TCP TTL:109 TOS:0x0 ID:49054 IpLen:20 DgmLen:48 DF
*****S* Seq: 0xBBD9A31 Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=====
01/17-03:39:08.718804 A.B.164.119:80 -> 202.99.41.15:64696
TCP TTL:128 TOS:0x0 ID:34606 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x16EC7A Ack: 0xBBD9A32 Win: 0x2238 TcpLen: 20
=====
01/17-03:39:08.719942 202.99.41.15:64695 -> A.B.164.119:8080
TCP TTL:109 TOS:0x0 ID:51870 IpLen:20 DgmLen:48 DF
*****S* Seq: 0xBBD9A30 Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=====
01/17-03:39:08.720034 A.B.164.119:8080 -> 202.99.41.15:64695
TCP TTL:128 TOS:0x0 ID:34862 IpLen:20 DgmLen:40
***A**R** Seq: 0x0 Ack: 0xBBD9A31 Win: 0x0 TcpLen: 20
=====
01/17-03:39:14.692020 A.B.164.119:80 -> 202.99.41.15:64696
TCP TTL:128 TOS:0x0 ID:35118 IpLen:20 DgmLen:44 DF
***A**S* Seq: 0x16EC79 Ack: 0xBBD9A32 Win: 0x2238 TcpLen: 24
TCP Options (1) => MSS: 1460
=====
01/17-03:39:14.953549 202.99.41.15:64696 -> A.B.164.119:80
TCP TTL:109 TOS:0x0 ID:18335 IpLen:20 DgmLen:40
*****R** Seq: 0xBBD9A32 Ack: 0xBBD9A32 Win: 0x0 TcpLen: 20
=====

```

3. Probability the Source Address was Spoofed

A scanner typically wants to receive the feedback from their scan so the source address 202.99.41.15 is probably. Also a connection is half opened by the local host, but 202.99.41.15 abruptly RESETS it implying that 202.99.41.15 is a live host, although not definitively the scanning host. However, it is still possible that a local host is spoofing this traffic and sniffing the response as well.

4. Description of Attack

This detect appears to be an automated recognizance scan looking for available web and proxy servers on ports 80 and 8080. This noisy SYN scan attempted to solicit SYN/ACK's from any open services it finds with no stealth at all. This scan occurred only once over a period of two weeks, but in that scan two attempts to connect to port 80 were made three seconds apart. This is indicative of a windows operating system re-try between the first two out of four packets attempts, but it might also be a deliberate attempt to "look like" a windows host. The packet captures above show that the scanner did not complete a handshake with the web server at A.B.164.119. Rather it ignored or RESET each half opened connection.

The sequence number of the first packet to port 80, highlighted in red, is 1 higher than the packet to port 8080. Yet the IP ID number, highlighted in blue, has changed from 254 to 510 respectively in the same packets. Perhaps the scanner is running multiple threads scanning several subnets at the same moment. The IP ID numbers seem to roll over during this brief but intense scan suggesting the scanner is quite busy sending a lot of packets.

There are numerous CVE entries for proxy vulnerabilities and for many different products. A search for "web proxy" at the CVE site returned the results found here: <http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=web%20proxy> One notable entry in this list is CVE-2001-0239. Microsoft's Internet Security and Acceleration Server 2000 Web Proxy has a vulnerability, which "allows remote attackers to cause a denial of service via a long web request." Webmin, a web based Linux administration program uses which port 8080, turned up CVE-2001-0222 regarding a symlink attack by local users, and two more CVE candidates. These are found at: <http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=webmin>.

5. Attack Mechanism

The scanner is performing reconnaissance work by sending SYN packets at ports 8080 and 80 on an entire subnet. It may be listening for a RESET or SYN/ACK's to indicate live hosts or a responding service, and recording this information for future use. Most likely, it is a prelude to some other form of attack. Perhaps they're looking for web defacement targets. Or maybe they want to surf anonymously via a web proxy.

A SYN scan at ports 8080 and 80 are consistent with the Ring Zero Trojan, or Webmin 8080 scan attacks. However there was no reported activity involving port 3128, which is also associated with the Ring Zero Trojan, which reduces the chances this is a Ring Zero detect.

6. Correlations

David R. Steiner reported a very similar scan in July 2000. At the time it was inconclusive whether the scan he reported was Ring Zero.

What is interesting to note is the source of the scan. The IP lookup page at www.dshield.org reports 202.99.41.15 has been reported 51 times, scanning 18 targets on 8080 during the first two weeks of Jan 2002. It is registered to CJNET, a Chinese company called Beijing Chang Jie Communication. There is no mention of port 80 scanning activity.

7. Evidence of Active targeting

This scan may be targeted at services on the subnet, but not at any one host on the subnet. The scanner does not appear interested in completing any connections if a SYN/ACK is offered. Rather it appears to be too busy scanning a large number of hosts since it fails to acknowledge the web server's first SYN/ACK with so much as a RESET, and the IP ID's are changing dramatically.

8. Severity

Scott Shinberg is credited with the formatting of this section, which is borrowed from his practical: http://www.giac.org/practical/Scoot_Shinberg_GCIA.doc

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

The host is a Honeypot intended to catch the attention of hackers so the criticality is 1. This detect succeeded in revealing vulnerable services on the host so the lethality is 3. There are some patches applied to the operating system, but it is not hardened: system counter measures are 2. This server is exposed to the Internet without the benefit of a firewall so the network counter measures are 1.

Criticality:	1	(Honeypot)
Lethality:	3	(successful scan)
System Countermeasures:	2	(some patches applied)
Network Countermeasures:	1	(none)
Severity = (1+3) – (2+1) =	1	

9. Defensive Recommendations

The 202.99.41.15 scanner should be placed on a watch list for a possible return for a more in-depth scans of the same services, or with a targeted exploit for the web proxy or web services. Since the owner of CJNET is known, the scan should be reported to the administrator. Continued network monitoring of the Honeypot may reveal more information of the attacker's intentions. Regardless Administrators of the network and other web servers, beyond the Honeypot, should be alerted to the scanner's activities so that the servers can be properly protected by turning off any un-needed web, web proxy services, apply needed security patches, and double checking website security.

10. Multiple Choice Test Question

How many different ports are typically associated with the Ring Zero Trojan?

- A) 80
- B) 3
- C) 3128
- D) 8080

The correct answer is: B


```

43 6F 6D 70 69 6C 65 64 20 54 75 65 20 30 31 2D Compiled Tue 01-
4A 75 6E 2D 39 39 20 31 37 3A 31 36 20 62 79 20 Jun-99 17:16 by
70 77 61 64 65 00 06 00 0E 63 69 73 63 6F 20 32 pwade....cisco 2
36 31 30 C0 A8 FC 00 18 0A 19 19 19 20 00 0B 00 610.....
05 01 0D 0A 3C 62 6F 64 79 3E 0D 0A 0D 0A 3C 68 ....<body>....<h
32 3E 48 54 54 50 20 45 72 72 6F 72 20 34 30 31 2>HTTP Error 401
3C 2F 68 32 3E 0D 0A 0D 0A 3C 70 3E 3C 73 74 72 </h2>....<p><str
6F 6E 67 3E 34 30 31 2E 32 20 55 6E 61 75 74 68 ong>401.2 Unauth
6F 72 69 7A 65 64 3A 20 4C 6F 67 6F 6E 20 46 61 orized: Logon Fa
69 6C 65 64 20 64 75 65 20 74 6F 20 73 65 72 76 iled due to serv
65 72 20 63 6F 6E 66 69 67 75 72 61 74 69 6F 6E er configuration
3C 2F 73 74 72 6F 6E 67 3E 3C 2F 70 3E 0D 0A 0D </strong></p>...
0A 3C 70 3E 54 68 69 73 20 65 72 72 6F 72 20 69 .<p>This error i
6E 64 69 63 61 74 65 73 20 74 68 61 74 20 74 68 ndicates that th
65 20 63 72 65 64 65 6E 74 69 61 6C 73 20 70 61 e credentials pa
73 73 65 64 20 74 6F 20 74 68 65 20 73 65 72 76 ssed to the serv
65 72 20 64 6F 20 6E 6F 74 20 6D 61 74 63 68 20 er do not match
74 68 65 20 63 72 65 64 65 6E 74 69 61 6C 73 20 the credentials
72 65 71 75 69 72 65 64 20 74 6F 20 6C 6F 67 20 required to log
6F 6E 20 74 6F 20 74 68 65 20 73 65 72 76 65 72 on to the server
2E 20 54 68 69 73 20 69 73 20 75 73 75 61 6C 6C . This is usuall
79 20 63 61 75 73 65 64 20 62 79 20 6E 6F 74 20 y caused by not
73 65 6E 64 69 6E 67 20 74 68 65 20 70 72 6F 70 sending the prop
65 72 20 57 57 57 2D 41 75 74 68 65 6E 74 69 63 er WWW-Authentic
61 74 65 20 68 65 61 64 65 72 20 66 69 65 6C 64 ate header field
2E 3C 2F 70 3E 0D 0A 0D 0A 3C 70 3E 50 6C 65 61 .</p>....<p>Plea
73 65 20 63 6F 6E 74 61 63 74 20 74 68 65 20 57 se contact the W
65 62 20 73 65 72 76 65 72 27 73 20 61 64 6D 69 eb server's admi
6E 69 73 74 72 61 74 6F 72 20 74 6F 20 76 65 72 nistrator to ver
69 66 79 20 74 68 61 74 20 79 6F 75 20 68 61 76 ify that you hav
65 20 70 65 72 6D 69 73 73 69 6F 6E 20 74 6F 20 e permission to
61 63 63 65 73 73 20 74 6F 20 72 65 71 75 65 73 access to reques
74 65 64 20 72 65 73 6F 75 72 63 65 2E 3C 2F 70 ted resource.</p>
3E 0D 0A 0D 0A 3C 2F 62 6F 64 79 3E 3C 2F 68 74 >....</body></ht
6D 6C 3E D7 C8 67 98 59 B9 9B E1 BC F3 66 A5 EF ml>..g.Y.....f..
CE 19 D3 8B ED AC B3 0E 3C 30 81 82 85 1B DA 6E .....<0.....n
A8 0E 9C E4 93 DD 62 D5 2D 2E 53 4A 02 AC AF 42 .....b.-.SJ...B
9E 85 2D 96 23 B7 C2 2D D1 8E 6D 1D DA 5B 42 1B ..-.#...-.m..[B.
1B F4 D7 04 2F 8B DC 5B 0F 04 01 00 21 F9 04 04 ..../<[....!...
0E 00 00 00 2C 00 00 00 00 64 00 1F 00 00 06 FF ....,....d.....
C0 C5 02 40 2C 1A 8F C8 A4 D2 C0 2C 32 0D 4E 00 ...@,.....,2.N.
14 2A 7D 12 9F 82 AB B5 6A C8 1A B1 D2 28 F1 31 .*).....j....(.1
7A 08 95 E8 34 1A 5A B8 0E 17 D4 69 78 1E 2D C4 z...4.Z....ix.-.
8F 50 41 FB 0B D0 1B 19 46 64 66 43 6A 85 85 06 .PA.....FdfCj...
76 57 59 7E 52 0B 02 0B 7B 88 4D 52 76 5E 92 54 vWY~R...{.MRv^.T
6D 02 54 4E 76 7B 89 46 0B 65 67 23 A4 A5 A6 A7 m.TNv{.F.eg#....
A8 A7 6B 47 76 9B 61 9B 4F 91 57 5A 88 B4 5E 4E ..kGv.a.O.WZ..^N
0C 9F 74 44 A1 83 00 A9 C0 C1 A4 86 C4 4A 84 46 ..tD.....J.F
01 C5 69 BD A3 C2 1F 12 D0 12 1F C1 45 A1 45 23 ..i.....E.E#
44 D8 CA 47 D6 DB C4 0B 16 66 D9 12 11 10 11 17 D..G.....f.....
D3 23 12 B1 4F 06 A7 1F 17 DA 00 A1 D8 C3 F3 44 .#..O.....D
1F 6F 84 67 F3 F9 BC 84 E2 E4 79 4B 13 8E 90 3A .o.g.....yK...:
08 17 06 28 6C C0 C4 8E B4 67 92 A4 6D 98 90 60 ... (l....g..m..`
83 3C 7A BF B0 59 CB 57 0F 1B 99 11 42 40 7A 1C .<z...Y.W....B@z.
61 C1 42 C6 52 03 D5 14 CC 96 6E C4 85 06 05 0A a.B.R.....n.....
0C 90 30 42 04 08 0E 22 44 7C 30 90 20 C1 04 FF ..0B..."D|0. ...
74 F1 AA 89 B4 16 AA DB AF 6C 48 8B 1E 3D 0A 72 t.....lH...=r

```

```

69 4A 34 0F C4 FD 9A 40 75 27 93 06 0A 13 64 B8 iJ4....@u'....d.
B9 15 04 C5 5C 12 B0 5E AC 77 B4 A8 C6 A5 DA E4 ....\...^..w.....
89 44 DA 54 E0 53 24 51 0D 3E EB C9 80 A6 88 05 .D.T.S$Q.>.....
0A 07 C4 1B 41 21 03 05 11 1E 0C C0 A4 29 94 E9 ....A!.....)..  

3C 8D C3 B4 85 FA 90 AD 29 80 69 43 9A 1A 7D 7B <.....).iC..}{  

24 6E B6 27 10 26 2C 00 71 F3 43 04 85 11 36 F8 $n.'.&,.q.C...6.  

BC 99 60 40 83 75 6E AB 09 39 56 88 1F 28 A9 FD ..`@.un..9V..(..  

FC 3D F8 90 0F 36 B1 07 16 0C 92 CA 09 C2 EF 82 .=...6.....  

74 1F B2 26 80 80 75 00 83 BF F6 2A 9F 62 8D 26 t..&...u....*.b.&  

39 BE 52 8C 33 86 FB 18 6E 44 F4 62 B8 0D 72 A6 9.R.3...nD.b..r.  

F0 F7 83 E8 09 1B 2E 50 EC 99 B7 00 6D 9B 20 52 .....P.....m. R  

03 20 93 6F 1A E3 D9 1F A4 CE 7E D0 78 9E 99 8F . .o.....~.x...  

F4 CC 90 32 63 C1 BA 99 F8 63 00 18 1B 7D F6 8D ...2c....c...}..  

91 5B 63 2D 91 F2 C1 78 3E 5D E0 01 34 09 A2 04 .[c-...x>]..4...  

D7 7E 90 91 61 81 7B 8F 59 67 5D 46 8B 65 48 FF .~..a.{.Yg]F.eH.  

1F 7B 1F 71 B4 1F 7B 19 4D A3 1F 49 1B FA 33 CF .{.q..{.M..I..3.  

81 27 A5 82 01 3C C2 48 A8 1C 74 BC CC 96 98 47 .'....<.H..t....G  

F4 29 D8 56 19 EB DD 38 06 8F EC 75 18 5F 90 CE .).V...8....u...  

AD A8 5B 8C 48 3A 16 C8 86 44 6A 28 92 40 A6 D0 ..[.H:...Dj](. @..  

67 62 86 8C D9 23 48 8F F1 6D F8 9C 75 B4 B1 06 gb...#H...m..u...  

CE 91 49 3A 43 A0 50 4E DA 98 65 47 00 00 00 00 ..I:C.PN..eG....  

14 40 00 A0 00 00 2E 32 31 31 2E 31 35 31 2E 32 .@.....211.151.2  

34 32 00 00 2E 32 31 31 2E 31 35 31 2E 32 34 32 42...211.151.242  

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..  

00 00 00 00 00 --Snipped--

```

3. Probability the Source Address was Spoofed

The probability of this source address being spoofed depends on the originator's intent. A simple ECHO ping request may not be spoofed, but the contents of this ping packet really puts the matter in doubt. An attempt to ping the source address resulted in a request timed out suggesting that the host is currently offline, or does not exist. The possibilities are: a ping scan, Smurf attack, or Smurf network probe (Smurf is discussed in more detail below in Section 5). The source address of a scan is usually not spoofed. Although it is doubtful this pattern is anything more than a scan, but if it is a Smurf probe or attack then there are two possibilities regarding the source.

The first is a Smurf attacker. The attacker may be spoofing the source IP to conduct a classic Smurf attack against 62.211.151.242, since it is an ICMP ECHO request directed at the broadcast addresses A.B.165.0, and A.B.165.255. However, if the attacker is a slow probe for Smurf amplifier networks, then he might want to see the echo replies generated, so he wouldn't spoof his IP. This is supported by the fact only a couple of packets are logged by the sensor on the subnet, so there is little chance the attacker is going to cause a self inflicted denial of service.

The second possibility, which is least likely, is that 62.211.151.242 is a host under attack who is not spoofing, and is returning an automated response to the perceived attackers who are the broadcast addresses. This theory is contrary to the use of ICMP ECHO, but its derived from the alleged meaning of packet contents explained below.

4. Description of Attack

This is an ICMP ECHO (Ping) request with a very large payload. It is designed to solicit an echo response (Pong) from the destination host(s). However this particular ping is directed at the broadcast addresses of A.B.165.0, and A.B.165.255, and has the Don't Fragment bit set. The A.B.165.0 address is a broadcast address for operating systems, which use a BSD derived IP stack such as some flavors of Linux. A.B.165.255 is the broadcast address of the A.B.165.0 subnet by definition. The concept of pinging the network broadcast address to discover live hosts is common in network scans. What is really unusual about these particular ping packets is their large payload size, which is misreported in the packets above, and the fact that they carry a complete HTTP coded unauthorized logon error report in each one! The Honeypot did not send any recorded stimuli for these logon error messages, nor did it reply to the broadcasts since Windows NT does not respond to ping broadcasts.

A portion of the payload (less than half!), is shown here by using the Snort -vd switch to dump the Application layer. It appears to be HTTP code stating a Cisco product is reporting an unauthorized logon failure error occurred, followed by a bunch of garbage data. When the HTTP code is cut out of the packet and placed in a file Figure 4 appears:

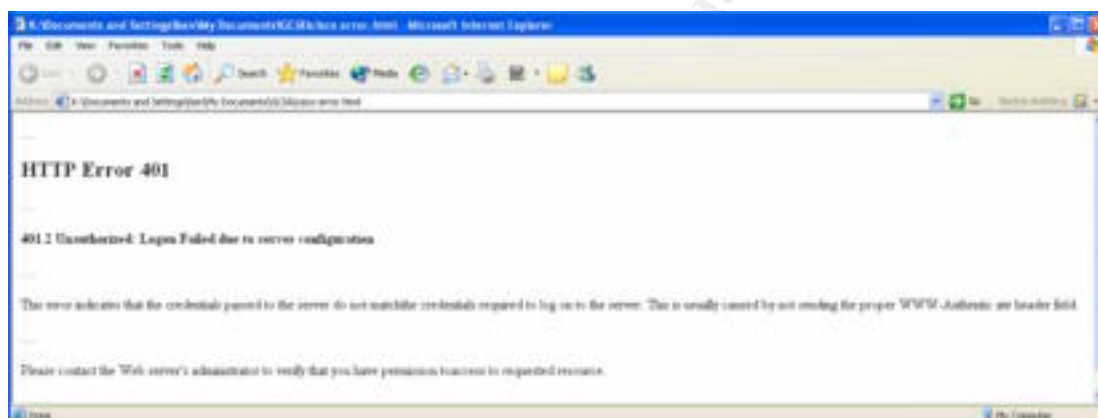


Figure 4

The CVE reference to Smurf is CVE-1999-0513.

5. Attack Mechanism

The attacker seems to be simply scanning the network for live hosts using large ICMP Echo packets carrying an HTTP error message with the DF bit set. The Scanner might be interested in both live hosts and the MTU size of the network since the Don't Fragment bit is set, explaining the large payload, but why incriminate a Cisco device as the source of a ping? If the Don't Fragment bit weren't set then perhaps this could be a Ping o' Death attack, but that requires packet fragmentation and re-assembly to occur. So assume for the moment, that the source host is in fact a Cisco device, or at least a crafted packet we are meant to believe is came from one.

This HTTP error reports suggest an attempt was made to logon to a device at 62.211.151.242 by A.B.165.0 and A.B.165.255. If this is the case, then these ECHO requests are the responses to another host spoofing the addresses and stimulating the error messages. However, it's a little

strange to access a Cisco device from a broadcast address, since the reply would be sent to the entire network. Therefore the packets are very likely spoofed source IP's raising the possibility of a Smurf attack, or perhaps a brute force logon attempt?

A Smurf attack is a denial of service attack in which an attacker spoofs the IP of the victim host and sends multiple ICMP ECHO Requests to the broadcast address of one or more Smurf amplifier networks. An amplifier network is an intermediate network, which allows hosts to reply to in-coming ICMP ECHO broadcast traffic and send their replies, now amplified by the number of responding hosts, to the victim host. The attacker continues to send the ECHO Requests, flooding the victim's network and rendering the victim host unable to process all of the incoming packets, thus suffering a denial of service until the attacker gives up or the packets are blocked.

This does not appear to be a sustained Smurf attack since there were only two packets reported in the capture. The possibility remains, however, that this was a probe for Smurf amplifier networks and their optimal MTU size. But the embedded logon error message is really confusing the matter, because it implies that the Honeypot (the entire A.B.165.0 subnet actually) stimulated the packets. The confusion is probably the intended effect and further evidence of packet craft since a well tuned, properly functioning router is not likely to ping a broadcast address with an error report. This is most likely an attacker impersonating a router with the Cisco error report while network scanning. Just in case they get caught, they could say, "It wasn't me. Blame that broken router!" It is possible that the same tool used to craft these packets could be used for scanning or Smurfing purposes.

6. Correlations

An example of network ping scanning appears in, "I am seeing odd ICMP traffic, what could this mean?" of the SANS Online Resources FAQ. No detects with this exact payload could be found at www.sans.org, or at www.yahoo.com using the search string, " "large icmp" AND Smurf AND cisco". There were no reports of this IP which is registered to TINIT-ADSL-LITE, an Italian ASDL provider, at DShield.org.

7.Evidence of Active targeting

This could be active targeting of the A.B.165.0 subnet since the packets are directed at the broadcast addresses of this subnet, and a stealthy Smurf probe would be slow and deliberate.

8.Severity

Scott Shinberg is credited with the formatting of this section, which is borrowed from his practical: http://www.giac.org/practical/Scoot_Shinberg_GCIA.doc

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

The host is a Honeypot intended to attract hackers so the criticality is 1. This probe was not successful in revealing any information from the host so the lethality is 1. Windows computers will not respond to ICMP broadcasts so system counter measures are 5. This server is exposed to the Internet without the benefit of a firewall so the network counter measures are 1.

Criticality:	1	(Honeypot)
Lethality:	1	(unsuccessful probe)
System Countermeasures:	5	(Immune OS)
Network Countermeasures:	1	(none)
Severity = (1+1) – (5+1) = -4		

9. Defensive Recommendations

The 62.211.151.0 network should be placed on a watch list for further probing activities. Also the network's border routers and firewalls should be double checked to ensure that egress filtering of ICMP ECHO Responses is in place to ensure that the A.B.165.0 network will not be used as a Smurf Amplifier network.

10. Multiple Choice Test Question

What are two necessary elements of a Smurf attack?

- A) Amplifier networks
- B) Sergeant Smurf
- C) ICMP Echo
- D) Large payloads

The correct answer is A and C.

Detect-4 Port 6112 SYN Scan

1. Source of Trace

This trace was picked up on a Honeypot Windows NT server. It was offering several unprotected IIS4.0 web pages to the Internet.

2. Detect was Generated by

This detect was generated by Snort 1.8.3 for windows logging in binary mode, and later analyzed with a recent snort rules set downloaded from <http://www.snort.org> and the -vd switch.

Port Scan Log

```
Jan 12 05:41:58 213.191.132.98:6112 -> A.B.164.3:6112 SYN *****S*
Jan 12 05:41:58 213.191.132.98:6112 -> A.B.164.77:6112 SYN *****S*
Jan 12 05:41:58 213.191.132.98:6112 -> A.B.164.119:6112 SYN *****S*
Jan 12 05:41:58 213.191.132.98:6112 -> A.B.164.198:6112 SYN *****S*
Jan 12 05:41:58 213.191.132.98:6112 -> A.B.165.1:6112 SYN *****S*
Jan 12 05:41:58 213.191.132.98:6112 -> A.B.165.9:6112 SYN *****S*
Jan 12 05:41:58 213.191.132.98:6112 -> A.B.165.5:6112 SYN *****S*
Jan 12 05:41:58 213.191.132.98:6112 -> A.B.165.8:6112 SYN *****S*
Jan 12 05:41:58 213.191.132.98:6112 -> A.B.165.4:6112 SYN *****S*
```

--Snipped--

```
Jan 16 02:23:02 211.39.32.104:6112 -> A.B.164.3:6112 SYN *****S*
Jan 16 02:23:02 211.39.32.104:6112 -> A.B.164.77:6112 SYN *****S*
Jan 16 02:23:03 211.39.32.104:6112 -> A.B.164.100:6112 SYN *****S*
```

```

Jan 16 02:23:03 211.39.32.104:6112 -> A.B.164.111:6112 SYN *****S*
Jan 16 02:23:03 211.39.32.104:6112 -> A.B.164.119:6112 SYN *****S*
Jan 16 02:23:03 211.39.32.104:6112 -> A.B.164.198:6112 SYN *****S*
Jan 16 02:23:03 211.39.32.104:6112 -> A.B.165.2:6112 SYN *****S*
Jan 16 02:23:03 211.39.32.104:6112 -> A.B.165.1:6112 SYN *****S*
Jan 16 02:23:03 211.39.32.104:6112 -> A.B.165.4:6112 SYN *****S*
Jan 16 02:23:03 211.39.32.104:6112 -> A.B.165.6:6112 SYN *****S*
Jan 16 02:23:03 211.39.32.104:6112 -> A.B.165.8:6112 SYN *****S*
Jan 16 02:23:03 211.39.32.104:6112 -> A.B.165.3:6112 SYN *****S*
Jan 16 02:23:03 211.39.32.104:6112 -> A.B.165.5:6112 SYN *****S*

```

--Snipped--

Alerts

```

[**] [1:0:0] SCAN dtspc [**]
01/12-05:41:58.300668 213.191.132.98:6112 -> A.B.164.3:6112
TCP TTL:102 TOS:0x60 ID:47715 IpLen:20 DgmLen:40
*****S* Seq: 0x7C804B0F Ack: 0x34E03498 Win: 0xB193 TcpLen: 20

[**] [1:0:0] SCAN dtspc [**]
01/12-05:41:58.419600 213.191.132.98:6112 -> A.B.164.77:6112
TCP TTL:102 TOS:0x60 ID:47715 IpLen:20 DgmLen:40
*****S* Seq: 0x7C804B0F Ack: 0x34E03498 Win: 0xB193 TcpLen: 20

[**] [1:0:0] SCAN dtspc [**]
01/12-05:41:58.507602 213.191.132.98:6112 -> A.B.164.119:6112
TCP TTL:102 TOS:0x60 ID:47715 IpLen:20 DgmLen:40
*****S* Seq: 0x7C804B0F Ack: 0x34E03498 Win: 0xB193 TcpLen: 20

[**] [100:1:1] spp_portscan: PORTSCAN DETECTED from 211.39.32.104 (THRESHOLD
4 connections exceeded in 1 seconds) [**]
01/16-12:25:24.172000

[**] [100:2:1] spp_portscan: portscan status from 211.39.32.104: 261
connections across 261 hosts: TCP(261), UDP(0) [**]
01/16-12:25:24.192000

[**] [100:3:1] spp_portscan: End of portscan from 211.39.32.104: TOTAL
time(1s) hosts(261) TCP(261) UDP(0) [**]
01/16-12:25:24.192000

```

Packet Captures

```

=====
01/12-05:41:58.942197 213.191.132.98:6112 -> A.B.165.103:6112
TCP TTL:102 TOS:0x60 ID:47715 IpLen:20 DgmLen:40
*****S* Seq: 0x7C804B0F Ack: 0x34E03498 Win: 0xB193 TcpLen: 20
=====
01/12-05:41:58.942456 213.191.132.98:6112 -> A.B.165.105:6112
TCP TTL:102 TOS:0x60 ID:47715 IpLen:20 DgmLen:40
*****S* Seq: 0x7C804B0F Ack: 0x34E03498 Win: 0xB193 TcpLen: 20
=====
01/12-05:41:58.957249 213.191.132.98:6112 -> A.B.165.116:6112
TCP TTL:106 TOS:0x60 ID:20549 IpLen:20 DgmLen:40
*****S* Seq: 0x63D9D949 Ack: 0x5E64102C Win: 0x9209 TcpLen: 20
=====
01/12-05:41:58.957517 213.191.132.98:6112 -> A.B.165.111:6112
TCP TTL:106 TOS:0x60 ID:20549 IpLen:20 DgmLen:40

```

3. Probability the Source Address was Spoofed

4. Description of Attack

50

The 213.191.132.98 source IP scans from port 6112 for port 6112, yet changes TTL's from 102 to 106, while at the same time the window size changes from 0xB193 to 0x9209. But the 211.39.32.104 source IP shows constant values of TTL =243, and the window size of 0x28 or 40 bytes. The 40 byte length packets are similar to a correlating detect below. Each scanning IP succeeded in soliciting an ACK/RESET packet from the Honeypot as shown above. "SCAN dtspc" is the Snort alert these scans generated for the rules file used, along with the port scan logs showing 261 hosts were scanned in one second.

When the packets were dumped using the Snort switch -vdX, the following appeared:

```

=====
01/12-05:41:58.759148 213.191.132.98:6112 -> A.B.165.5:6112
TCP TTL:102 TOS:0x60 ID:47715 IpLen:20 DgmLen:40
*****S* Seq: 0x7C804B0F Ack: 0x34E03498 Win: 0xB193 TcpLen: 20
0x0000: 00 E0 81 02 CD 66 00 30 94 BC 55 40 08 00 45 60 .....f.0..U@..E`
0x0010: 00 28 BA 63 00 00 66 06 CB 95 D5 BF 84 62 0A 0B .(.c..f.....b.O
0x0020: A5 05 17 E0 17 E0 7C 80 4B 0F 34 E0 34 98 50 02 .....|.K.4.4.P.
0x0030: B1 93 CF 0F 00 00 00 00 00 00 00 00 00 00 00 .....
=====
01/12-05:41:58.761832 213.191.132.98:6112 -> A.B.165.8:6112
TCP TTL:102 TOS:0x60 ID:47715 IpLen:20 DgmLen:40
*****S* Seq: 0x7C804B0F Ack: 0x34E03498 Win: 0xB193 TcpLen: 20
0x0000: 00 E0 81 02 CD 66 00 30 94 BC 55 40 08 00 45 60 .....f.0..U@..E`
0x0010: 00 28 BA 63 00 00 66 06 CB 92 D5 BF 84 62 0A 0B .(.c..f.....b.O
0x0020: A5 08 17 E0 17 E0 7C 80 4B 0F 34 E0 34 98 50 02 .....|.K.4.4.P.
0x0030: B1 93 CF 0C 00 00 00 00 00 00 00 00 00 00 00 .....
=====

```

The purpose of this data, accompanying the SYN packet is unknown. A minor change affecting only three characters in the application dump data was observed when the packets were made to scroll quickly down the screen. There was no discernable pattern in these changing bytes shown above in bold. All other characters were static in the captures from 213.191.132.98 and 211.39.32.104.

The Common Desktop Environment (CDE) Subprocess Control Service, which uses TCP port 6112, has two CVE listings: CVE-1999-0689, and CAN-2001-0803 (under review). CVE-1999-0689 reports a CDE daemon, dtspcd, "allows local users to execute arbitrary commands via a symlink attack." CAN-2001-0803 is referencing the same buffer over-flow exploit discussed in the CERT advisory on 01/14/2002 below.

5.Attack Mechanism

There are three possible attack mechanisms for this detect. 1) A high port host scan, 2) A scan for a vulnerable service on port 6112, or 3) A scan for "The Free Standard Game Server" which also lives on port 6112.

The high port scan did solicit two ACK/RESET packets from the Honeypot machine shown above, which revealed its presence on the network to the scanner. This happened because the Windows host had no services running at TCP port 6112 and it informed the scanners of this fact

by responding to the SYNs with a ACK/RESET packet. There was also no firewall present to block either the arriving SYN packets or the out going responses.

A CERT advisory was issued on January 14, 2002 regarding the vulnerability in the Common Desktop Environment (CDE) Subprocess Control Service, dtspcd on TCP port 6112. This advisory can be found at: <http://www.cert.org/advisories/CA-2002-01.html>. The notice states “there is a remotely exploitable buffer overflow vulnerability in a shared library that is used by dtspcd,” which is usually configured to run on port 6112. If the buffer over-flow is successful, a remote host can run commands with root privileges.

The final possibility to explain this scan is a scan for The Free Standard Game Server (FSGS) network, which lives at port 6112. The link to the homepage for this application is here: <http://www.fsgs.net/fsgs/about.php>. FSGS is a tool for building gaming networks on the local LAN or the Internet for Windows, Linux, or FreeBSD. The web page boasts “at the moment there are servers in more than 175 countries, building a network of over 1200 interconnected gaming nodes.” Apparently it is possible to join a FSGS server to the FSGS network and the local users will be able to interact with that network over chat, game ladders, and download the latest enhancements. It supports almost a dozen well known games including Starcraft and Diablo II. Diablo is also reported to use port TCP/UDP port 6112 according to David Ranch in his article “Linux IP Masquerade HOW TO” which can be found at this link: <http://www.thelinuxreview.com/howto/IP-MASQ/x1215.htm>

The FSGS service alone might begin to explain the increased port scans for 6112 reported on www.incidents.org in December of 2001. If this service is capable of broadcast discovery of other FSGS servers, then perhaps it is scanning the Internet from port 6112 searching for port 6112. This might begin to explain why some of the correlating detects below show port 6112 scans originating from various ports, including port 6112, such as this one. A search of the website gleaned no answers to this question.

6. Correlations

On 12/17/2001 John Sage reported Snort log rule hits titled “TCP to range 1025-60999” in which he detected port 6112 hits and identified the service as The Free Standard Gaming Service at TCP/UDP port 6112. Vicki Irwin refers to Sage’s post and adds that The Storm Center has identified other probe detects which are 40 bytes long, which is the same length as some of the packets from this detect. She also posts a log in which the source port was TCP 6112. Sage identifies the IP in his log, as belonging to the Korean ISP NETSGO owned by BNC OnLine Co., Ltd. Dshield.Org identifies the 211.39.32.104 IP from this detect as belonging to the same ISP, but owned by SK Telecom Co., Ltd. of Seoul.

Sage’s posting can be found at: <http://www.incidents.org/archives/intrusions/msg02922.html>, and Irwin’s at: <http://www.incidents.org/archives/intrusions/msg02977.html>

Donald Smith provides another log file detect of the 211.39.32.104 IP in this detect, and identifies the port 6112 as the Solaris CDE Subprocess Control Service. He argues that the packets show evidence of crafting because the IP ID’s are repeated in his log. Evidence of

similar behavior is noted above in this detect. Smith's logs are viewable here:
<http://www.incidents.org/archives/intrusions/msg03506.html>

7.Evidence of Active targeting

The two scanning IP's showed interest in the same subset of hosts in the A.B.164.0 subnet suggesting earlier recognizance might have directed their attention here. It is also possible that one scanner used two different IP's on two occasions, or two different scanners used the same tool or intelligence that focused attention on the same destination hosts. No handshake SYN/ACKs were logged to or from either scanner on either visit and both seemed only interested in services offered at TCP port 6112, but not from any particular host on the A.B.165.0 subnet. This could also be a successful live host scan designed merely to solicit RESET's.

8.Severity

Scott Shinberg is credited with the formatting of this section, which is borrowed from his practical: http://www.giac.org/practical/Scoot_Shinberg_GCIA.doc

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

The host is a Honeypot intended to attract scans and hackers so the criticality is 1. The dtspc service was not available on this host, but the scanner may be aware of the host's presence due to a response, so the lethality is 2. System counter measures are 1 since Windows sent a RESET packet for the non-listening port, and this server is exposed to the Internet without the benefit of a firewall so the network counter measures are 1.

Criticality:	1	(Honeypot)
Lethality:	2	(the service does not exist/information leak)
System Countermeasures:	1	(host sent a response)
Network Countermeasures:	1	(none)
Severity = (1+2) – (1+1) =	1	

9.Defensive Recommendations

Further investigation into the Free Standard Gaming Service's expected behavior is needed to ascertain its use of TCP/UDP port 6112. This information will help determine if the detects shown here are legitimate FSGS traffic. The network administrator should scan the local network for any hosts running CDE Subprocess dtspcd on TCP port 6112, and secure them as needed. The firewall should be configured to deny all incoming traffic, which is not requested by an internal host's session, to prevent high port TCP scanning traffic from soliciting any ACK/RESET packets. Also the firewall should drop any ICMP port unreachable messages from leaving the network to prevent port scans from receiving any response at all.

10.Multiple Choice Test Question

What is the purpose of this traffic?

```
Jan 12 05:41:58 213.191.132.98:6112 -> A.B.164.3:6112 SYN *****S*
Jan 12 05:41:58 213.191.132.98:6112 -> A.B.164.77:6112 SYN *****S*
```

Jan 12 05:41:58 213.191.132.98:6112 -> A.B.164.119:6112 SYN *****S*
Jan 12 05:41:58 213.191.132.98:6112 -> A.B.164.198:6112 SYN *****S*

- A) SYN Scan.
- B) CDE service probe.
- C) FSGS service probe.
- D) Possibly all of the above.
- E) None of the above.

The correct answer is D.

Detect-5 Scripted FTP Scan & Storage Test

1. Source of Trace

The trace was picked up on a Honeypot Windows NT server. It was offering several unprotected, default IIS4.0 services to the Internet including anonymous FTP and Telnet.

2. Detect was Generated by

This trace was generated by Snort 1.8.3 for windows logging in binary mode, and later analyzed using the -vCd switch while filtering for the source IP 217.128.88.87.

Portions of this same pattern of FTP activity were seen originating from two IP's: 62.47.164.90, and 217.128.88.87. Only traffic from 217.128.88.87 will be discussed, since its scan included additional elements, which the other IP did not.

The first portion of this detect begins with a sporadic ICMP ping scan of the A.B.164.0 subnet, and a complete scan of the A.B.165.0 subnet on 01/19.

```

=====
01/19-22:30:00.697209 217.128.88.87 -> A.B.164.3
ICMP TTL:107 TOS:0x0 ID:37260 IpLen:20 DgmLen:64
Type:8 Code:0 ID:1568 Seq:0 ECHO
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
=====
01/19-22:30:01.434862 217.128.88.87 -> A.B.164.77
ICMP TTL:107 TOS:0x0 ID:37334 IpLen:20 DgmLen:64
Type:8 Code:0 ID:1568 Seq:0 ECHO
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
=====
01/19-22:30:01.853322 217.128.88.87 -> A.B.164.119
ICMP TTL:107 TOS:0x0 ID:37376 IpLen:20 DgmLen:64
Type:8 Code:0 ID:1568 Seq:0 ECHO
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
=====
01/19-22:30:01.853557 A.B.164.119 -> 217.128.88.87
ICMP TTL:128 TOS:0x0 ID:64313 IpLen:20 DgmLen:64
Type:0 Code:0 ID:1568 Seq:0 ECHO REPLY
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
=====
01/19-22:30:02.127428 217.128.88.87 -> A.B.164.146
```


The second portion is captured FTP traffic on port 21, and related traffic on port 80. Traffic on port 20 was logged, but not reported here to be as brief as possible, as were some contents of packets and other packets not relevant to this discussion. The attacker's commands appear in bold text immediately following the three-way handshake.

55

[illegible]


```

150 Opening BINARY mode data connection for /lmbtest.ptf(1048578
bytes)...
=====
01/20-15:00:44.026053 A.B.164.119:21 -> 217.128.88.87:2993
TCP TTL:128 TOS:0x0 ID:63823 IpLen:20 DgmLen:64 DF
***AP*** Seq: 0x5046D Ack: 0xD2B4B80E Win: 0x218B TcpLen: 20
226 Transfer complete...
=====
LIST -la..
=====
150 Opening ASCII mode data connection for /bin/ls...
=====
226 Transfer complete...
=====
DELE /lmbtest.ptf..
=====
250 DELE command successful...
=====
01/20-15:00:45.614644 217.128.88.87:2993 -> A.B.164.119:21
TCP TTL:107 TOS:0x0 ID:4231 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0xD2B4B871 Ack: 0x50554 Win: 0x416C TcpLen: 20
STOR /space.asp..
=====
150 Opening ASCII mode data connection for /space.asp...
=====
01/20-15:00:46.172563 A.B.164.119:21 -> 217.128.88.87:2993
TCP TTL:128 TOS:0x0 ID:3408 IpLen:20 DgmLen:64 DF
***AP*** Seq: 0x5058C Ack: 0xD2B4B882 Win: 0x2117 TcpLen: 20
226 Transfer complete...
=====

```

Next the attacker moves to port 80:

```

=====
01/20-15:00:46.505356 217.128.88.87:2998 -> A.B.164.119:80
TCP TTL:107 TOS:0x0 ID:4240 IpLen:20 DgmLen:121 DF
***AP*** Seq: 0xD3FBA56E Ack: 0x50277 Win: 0x4470 TcpLen: 20
GET /space.asp HTTP/1.1..Host: A.B.164.119..Accept: */*..Conn
ection: close....
=====
01/20-15:00:49.605575 A.B.164.119:80 -> 217.128.88.87:2998
TCP TTL:128 TOS:0x0 ID:4176 IpLen:20 DgmLen:202 DF
***AP*** Seq: 0x50277 Ack: 0xD3FBA5BF Win: 0x21E7 TcpLen: 20
HTTP/1.1 404 Object Not Found..Server: Microsoft-IIS/4.0..Date:
Sun, 20 Jan 2002 20:00:49 GMT..Connection: close..Content-Type:
text/html..Content-Length: 461....
=====
01/20-15:00:49.664254 A.B.164.119:80 -> 217.128.88.87:2998
TCP TTL:128 TOS:0x0 ID:4432 IpLen:20 DgmLen:501 DF
***AP**F Seq: 0x50319 Ack: 0xD3FBA5BF Win: 0x21E7 TcpLen: 20
<html><head><title>Error 404</title>....<meta name="robots" cont
ent="noindex">...<META HTTP-EQUIV="Content-Type" CONTENT="text/ht
ml; charset=iso-8859-1"></head>....<body>....<h2>HTTP Error 404<
/h2>....<p><strong>404 Not Found</strong></p>....<p>The Web serv
er cannot find the file or script you asked for. Please check th
e URL to ensure that the path is correct.</p>....<p>Please conta
ct the server's administrator if this problem persists.</p>....<

```

```
/body></html>
```

Back to FTP:

```
01/20-15:00:49.838095 217.128.88.87:2993 -> A.B.164.119:21
```

```
TCP TTL:107 TOS:0x0 ID:4247 IpLen:20 DgmLen:57 DF
```

```
***AP*** Seq: 0xD2B4B882 Ack: 0x505A4 Win: 0x411C TcpLen: 20
```

```
DELE /space.asp..
```

```
250 DELE command successful...
```

```
01/20-15:00:50.038885 217.128.88.87:2993 -> A.B.164.119:21
```

```
TCP TTL:107 TOS:0x0 ID:4248 IpLen:20 DgmLen:40 DF
```

```
***A***F Seq: 0xD2B4B893 Ack: 0x505C2 Win: 0x40FE TcpLen: 20
```

3.Probability the Source Address was Spoofed

None. The source IP completed the three-way handshake shown above and then proceeded to exchange data back and forth on several occasions.

4.Description of Attack

This appears to be an example of how FTP reconnaissance information is gathered in various stages. 217.128.88.87 did a ping scan of the A.B.164.0 and A.B.165.0 subnets. The A.B.164.0 subnet was scanned by a haphazard scheme, which seems to have found the Honeypot only by chance. The A.B.165.0 subnet scan was exhaustive scanning every possible live IP.

Immediately upon receiving a response, the attacker attempted to login to the FTP site with the user name anonymous@ftp.microsoft.com and was denied access. Notice the time difference shown in orange, between the ping reply and the first FTP login attempt. It is about 2 seconds. The snort log was searched for evidence of a host scan after the ping reply, but none was found.

The attacker returns to the FTP site the next morning and successfully logs on as **anonymous** with a password of Qgpuser@home.com. Several things happen during this session. The attacker attempts to change the working directory to a number of folders that do not exist. Then he creates the directory **020120133831** and deletes it. Next he identifies the FTP server's operating system by issuing the **SYST** command to query the system. Finally the attacker sends the command, "CWD ppppppppppp..." over a series four packets with the Don't Fragment Flag set on each. This appears to be a buffer over-flow attempt. Each of the four packets result in the FTP server RESEtting the connection yet the attacker ignores the RESETs.

This same session was repeated by 62.47.164.90. The only differences were the passwords and directory names used by each: Bgpuser@home.com and **020120170942p** for 62.47.164.90, and Qgpuser@home.com and **020120133831p** for 207.128.88.87. Notice the similarities in the passwords and directory names.

```
01/20-11:07:34.198859 62.47.164.90:2292 -> A.B.164.119:21
```

```
TCP TTL:112 TOS:0x0 ID:4369 IpLen:20 DgmLen:63 DF
```

```
***AP*** Seq: 0x916AB3D6 Ack: 0x50277 Win: 0x4497 TcpLen: 20
```

```

=====
01/20-11:07:35.956315 62.47.164.90:2292 -> A.B.164.119:21
TCP TTL:112 TOS:0x0 ID:4416 IpLen:20 DgmLen:59 DF
***AP*** Seq: 0x916AB441 Ack: 0x50404 Win: 0x430A TcpLen: 20
MKD 020120170942p..
=====

```

Having anonymous FTP enabled is under review by the CVE editorial board as CAN-1999-0497. Several long CWD command exploits are listed in the CVE database, but none refer directly to IIS 4.0. There were no CVE entries for space.asp or 1mbtest.pft specifically, but CAN-1999-0360 refers to the possibility content can be uploaded to a website and execute commands via Active Server Pages if MS Site Server 2.0 is installed with IIS 4.

This scanner at 217.128.88.87 used a ping scan to search for live hosts, and then attempted to connect to any FTP services on port 21. Based on the short amount of time involved with the FTP traffic in the second session, 4 seconds, and the similarity in the passwords and directory names used from both IP's, it is very likely these detects were scripted by a tool. The commands successfully demonstrated the anonymous user's privilege level on the server of Read & Write and the freedom to change directories. The CWD vti_pvt was an attempt to access a FrontPage directory which might contain "*.pwd" password files which are world readable. If successful this would allow the attacker to retrieve the password files for offline brute force attacking, leading to compromised web user accounts and possibly web administrator access. The last command issued: CWD pppppppp... appears to be a buffer over-flow exploit, which only succeeds in resetting the connection. The tool rudely ignores the servers RESET instructions suggesting the tool has a purpose in sending the packets regardless of the response.

Space.asp is an active server page script written by Garet Jax, which was designed to reveal local drive information to a remote viewer. Active Server Pages (ASP), are web pages designed to run on Microsoft's Internet Information Services, which perform a function on the server side, and return the results back to the client. For a more complete definition see: http://whatis.techtarget.com/definition/0,289893,sid9_gcj213787,00.html.

From the output it appears this script would reveal several items to the attacker including: server name, total space and free space available on the local disks, and what type of drives they are

(fixed volumes, network shares, removable media, CD-ROM, RAM disk, or unknown), and type of file system.

In this case, Space.asp failed since the attacker didn't take the time to move the file from the FTP server's home directory, into the home directory of the web server. This was possible since the Telnet service was also available with weak password encryption had the attacker simply looked for it with a host scan.

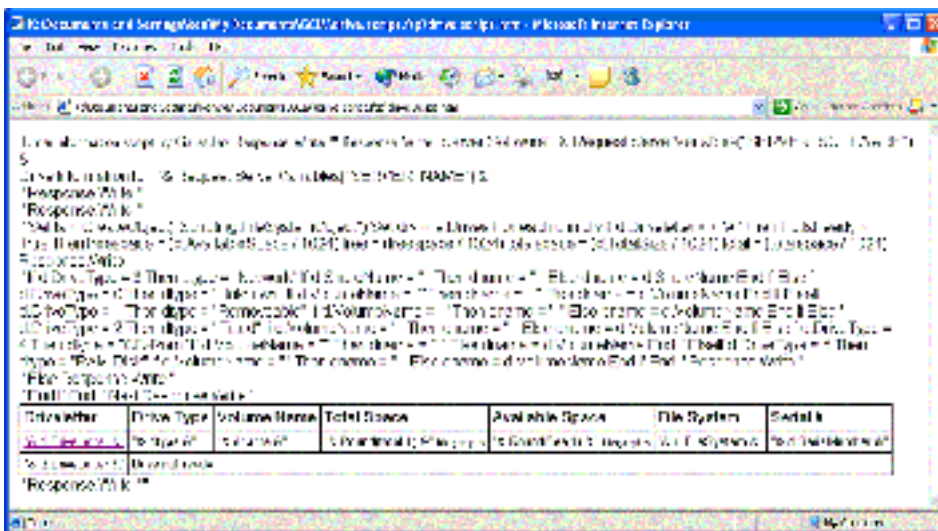


Figure 5

6. Correlations

Similar FTP scanning detects were documented by John Green on 09/10/00, and Dustin Decker on 12/28/00. They each reported scripted FTP scans, which seem to be related to the sequence of commands detected above. Interestingly enough, Decker's report includes the net name and description of his source as IP2000-ADSL-BAS and France Telecom IP2000 ADSL Broadband Access Servers, and www.incidents.org's IP lookup of 217.128.88.87 returned very similar information.

Matt Fearnow, a handler for GCIA, reported on 03/09/2001 a similar FTP session including Space.asp. His report mentions that sometimes a directory with a "funny name" is created which resembles the directories created by 62.47.164.90 and 217.128.88.87, using 12 numbers and a letter. This suggests the same tool, was in use by each attacker. Fearnow's post is at this URL: <http://www.sans.org/y2k/030901.htm>.

Ashley M. Kirchner reported Space.asp and 1mbtest.ptf, are two of three files he was seeing routinely left on his company's FTP server from script kiddies, and confirms that Space.asp tells them the server storage space if it works. His posting can be found at: <http://www.landfield.com/wu-ftp/mail-archive/wuftp-questions/2001/Jul/0164.html>.

Lastly, Kenny Persson has posted the complete Space.asp script here: <http://www.point-blank.nu/visa.asp?I=%05%EA%5C%5C%EB%D4%C0k%7D>

7.Evidence of Active targeting

The attacker most certainly actively targeted the Honeypot once it was discovered, by returning several times. The ICMP Ping Scan happened upon the FTP server by accident, but this information was used to test the anonymous logon capabilities, and next the directed FTP privilege level scan of the anonymous account on the server. Later this same information was used as the basis for the speed and storage tests, which took place in the third FTP session.

8.Severity

Scott Shinberg is credited with the formatting of this section, which is borrowed from his practical: http://www.giac.org/practical/Scoot_Shinberg_GCIA.doc

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

The host is a Honeypot intended to attract hackers so the criticality is 1. The FTP access attempt was successful in revealing useful information about the host including the operating system, so the lethality is 4. System counter measures are 1 since there were minimal patches applied and the service was running with anonymous access allowed. This server is exposed to the Internet without the benefit of a firewall so the network counter measures are 1.

Criticality:	1	(Honeypot)
Lethality:	4	(successful FTP access/information leak)
System Countermeasures:	1	(poorly configured FTP service)
Network Countermeasures:	1	(none)
Severity = (1+4) – (1+1) =	3	

9.Defensive Recommendations

The FTP and Web servers should be configured to disallow anonymous access, prevent remote, or at least anonymous users from executing files or scripts in their respective directories, and patched against known exploits. The network firewall should drop external ICMP ECHO requests before they enter the network, and the Administrator might consider restricting FTP access to designated IP addresses only. The IP 217.128.88.87 should be placed on a watch list since it will very likely return again to utilize this FTP server.

10.Multiple Choice Test Question

Which of the following is true regarding the file Space.asp?

- A) It is a HTML document with embedded scripts.
- B) Space.asp is written in a standard IIS format.
- C) It will work on Apache Web Servers with ASP support.
- D) Space.asp will leak information regarding volumes names, file types, and available free space on the FTP server.
- E) All of the above.

The correct answer is E.

References

BlackCode.com "Details of 'RC1 trojan'." 08/1997.

URL: <http://www.blackcode.com/trojans/details.php?id=1073> (Jan. 2002).

"Broadcast Address Parameters." Configuring Network Connections.

URL: http://osr5doc.ca.caldera.com:457/NetConfigG/configparamsC.broadcast_address.html (Feb. 2002).

CERT.org. "CERT[®] Advisory CA-2002-01 Exploitation of Vulnerability in CDE Subprocess Control Service." Jan. 14, 2002

URL: <http://www.cert.org/advisories/CA-2002-01.html> (Feb. 2002).

Decker, Dustin. "Global Incident Analysis Center, Detects Analyzed 12/28/00." 12/28/2000.

URL: <http://www.sans.org/y2k/122800.htm> (Jan. 2002).

Deering, Steve. "RFC 1112: Host Extensions for IP Multicasting."

Stanford University Computer Science Department Stanford. 08/1989.

URL: <http://asg.web.cmu.edu/rfc/rfc1112.html> (Jan. 2002).

Fearnow, Matt. "Global Incident Analysis Center, Detects Analyzed 03/09/01." 03/09/2001.

URL: <http://www.sans.org/y2k/030901.htm> (Jan. 2002).

Fenner, William C. "RFC 2236: Internet Group Management Protocol, Version 2."

Xerox PARC. 11/1997. URL: <http://sunsite.dk/RFC/rfc/rfc2236.html> (Jan. 2002).

Gmeiner, Wieland. "[suse-security] request for help with log entries." 12/05/2001.

URL: <http://webmail.cotse.com/nix/susesecurity/lists/susesec/current/0124.html> (Jan. 2002).

Green, John. "Global Incident Analysis Center, Detects Analyzed 9/10/00." 09/10/2000.

URL: <http://www.sans.org/y2k/091000.htm> (Jan. 2002).

Irwin, Vicki. "Handler's Diary 12/18/01." 12/19/2001.

URL: <http://www.incidents.org/archives/intrusions/msg02977.html> (Feb. 2002).

Jensen, Peter G. "Problem med iptables og Win9x maskiner." 10/30/2000.

URL: http://www.sslug.dk/emailarkiv/teknik/2000_10/msg01278.html (Jan. 2002).

Kirchner, Ashley M. "Preventing lechers." 07/29/2001.

URL: <http://www.landfield.com/wu-ftp/mail-archive/wuftp-questions/2001/Jul/0164.html> (Jan. 2002).

Lokesh, B. K. "Denial of Service Attacks - DDOS, SMURF, FRAGGLE, TRINOO."

03/01/2001. URL: http://rr.sans.org/threats/dos_attacks.php (Jan. 2002).

[The MITRE Corporation](#) "CAN-1999-0200 (under review)." Common Vulnerabilities and Exposures (CVE). 07/14/1999.

URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0200> (Feb. 2002).

[The MITRE Corporation](#) “CAN-1999-0360 (under review).” Common Vulnerabilities and Exposures (CVE). 06/23/1999

URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0360> (Feb. 2002).

[The MITRE Corporation](#) “CAN-1999-0497 (under review).” Common Vulnerabilities and Exposures (CVE). 07/28/1999.

URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0497> (Feb. 2002).

[The MITRE Corporation](#) “CVE-1999-0513.” Common Vulnerabilities and Exposures (CVE). 09/25/1999. URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0513> (Feb. 2002).

[The MITRE Corporation](#) “CVE-1999-0689.” Common Vulnerabilities and Exposures (CVE). 01/04/2000. URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0689> (Feb. 2002).

[The MITRE Corporation](#) “CVE-1999-0918.” Common Vulnerabilities and Exposures (CVE). 01/04/2000. URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0918>

[The MITRE Corporation](#) “CVE-2001-0222.” Common Vulnerabilities and Exposures (CVE). 05/07/2001. URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0222> (Feb. 2002).

[The MITRE Corporation](#) “CVE-2001-0239.” Common Vulnerabilities and Exposures (CVE). 09/18/2001. URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0239> (Feb. 2002).

[The MITRE Corporation](#) “CAN-2001-0796 (under review).” Common Vulnerabilities and Exposures (CVE). 10/23/2001.

URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0796> (Feb. 2002).

[The MITRE Corporation](#) “CAN-2001-0803 (under review).” Common Vulnerabilities and Exposures (CVE). 11/22/2001.

URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0803> (Feb. 2002).

Nanney, Jim. “Strange Activity – Help.” 02/21/2001.

URL: <http://archives.neohapsis.com/archives/incidents/2001-02/0287.html> (Jan. 2002).

Persson, Kenny. “Vad gör detta script??” 11/12/2001.

URL: <http://www.point-blank.nu/visa.asp?I=%05%EA%5C%5C%EB%D4%C0k%7D> (Jan. 2002).

Ranch, David. “Linux IP Masquerade HOWTO.” 11/11/2000

URL: <http://www.thelinuxreview.com/howto/IP-MASQ/x1215.htm> (Feb. 2002).

Sage, John. “[Logs] tcp:23, tcp:6112 FSGS, udp:137, tcp:21 probes at FinchHaven for 12/16/2001.” 12/17/2001. URL: <http://www.incidents.org/archives/intrusions/msg02922.html> (Feb. 2002).

SANS Resources. "I am seeing odd ICMP traffic, what could this mean?" Intrusion Detection FAQ. Version 1.52

URL: <http://www.sans.org/newlook/resources/IDFAQ/traffic.htm> (Jan. 2002).

Sawyer, Scott. forum posting: " [Luna] Security Puzzle of the week." 05/26/2001.

URL: <http://www.infomagic.net/pipermail/luna/2001-May/000285.html> (Jan. 2002).

Shinberg, Scoot. "Practical Assignment GCIA."

URL: http://www.giac.org/practical/Scoot_Shinberg_GCIA.doc (Jan. 2002).

Smith, Donald. "RE: (Fwd) GGI-IDR20020121.2 : CA-2002-01 @ IOTDC OTE Lab."

01/24/2002. URL: <http://www.incidents.org/archives/intrusions/msg03506.html> (Feb. 2002).

Steiner, David. R. "Global Incident Analysis Center, Detects Analyzed 7/29/00." 07/29/2000.

URL: <http://www.sans.org/y2k/072903.htm> (Jan. 2002).

Stephen Northcutt, Judy Novak, Network Intrusion Detection: An Analyst's Handbook.

September, 2000, New Riders

Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Boston: Addison Wesley Longman, Inc, 1994.

United Hackers Association 1. "UHA Magazine, Issue IV." November 01, 1998.

URL: <http://web.textfiles.com/eazines/UHA/uha1-4.txt> (Jan. 2002).

www.dshield.org IP lookup of 202.99.41.15.

URL: <http://www.dshield.org/ipinfo.php?ip=202.99.41.15&Submit=Submit> (Jan. 2002).

www.dshield.org IP lookup of 211.39.32.104.

URL: <http://www.dshield.org/ipinfo.php?ip=211.39.32.104&Submit=Submit> (Feb. 2002).

www.whatis.techtarget.com, "Active Server Page."

URL: http://whatis.techtarget.com/definition/0,289893,sid9_gci213787,00.html (Jan. 2002).

ASSIGNMENT 3- ANALYZE THIS

GIAC University Intrusion Detection Preliminary Audit

An Intrusion Detection Audit of the GIAC University Network, MY.NET, was conducted for the week of January 4, through January 8, 2002. This is a preliminary audit without the benefit of in-depth knowledge of the network topology, Intrusion Detection System configuration, or local security policy. Lacking this information the analysis assumes that a standard Snort rules set is in use, and all addresses logged are legitimate and not spoofed, nor are misrepresented by way of ARP or DNS cache poisoning to avoid frivolous speculation. Without a working knowledge of the network's firewall/router policies it is impossible to determine which traffic, on a given port, did or did not originate within MY.NET. However all MY.NET hosts will be considered internal, and all others EXTERNAL. "256.256." was substituted for "MY.NET."

The focus of this report will be on tuning the IDS to reduce the large amount of apparent false positives recorded. This is a necessary step in preparation for a more effective, comprehensive Intrusion Detection analysis to be conducted later. The "Top 10's" listings include signatures, shown in Table 2 and Table 5 that account for 99% of all alerts and over 960,000 scan alerts. They were logged during the first week of 2002 which seems excessive considering most students had not yet returned to Campus during the holiday break. To this end, only the Top 10 alerts and scans will be examined in detail for signs of misconfiguration or system compromises. Following this there will be a brief discussion of Out of Specification packets, and a list of external hosts that require further investigation into their activities. The remaining alerts are discussed only where IP addresses in common with the "Top 10's" show evidence of intrusion, compromises, or suspicious behavior. Correlating links to similar detects with further information are listed in each section, and defensive recommendations will be offered wherever required.

Executive Summary

The MY.NET network Intrusion Detection System has a poorly tuned rule set which is generating an enormous number of false scans and alerts. This is primarily due to an improperly configured Snort.conf file, and rules that are alerting on normal traffic. The port scan preprocessor in Snort.conf may be set to 1. This threshold needs to be increased. All of this is over shadowing the real alerts messages and making your Analyst less effective, and is endangering the security of the network. For instance *ICMP traceroute* the #1 Alert, is mainly generated by a single Source IP, 256.256.5.202.

To correct this situation each of the Top Ten Alerts were examined in-depth by using 3DV8 Professional to graphically view the logs for patterns and anomalies to draw conclusions after they were sorted and analyzed by Snort-Snarf, Snort-sort, and Personal Brain.

Alerts Totals		Scan Totals	
119230	Total alerts	153051	TCP Scans
71	Distinct Alert Signatures	118	Distinct TCP Scan Signatures
406	Source Hosts	808,932	UDP Scans
857	Destination Hosts	409	UDP Source Hosts
		20,469	UDP Destination Hosts

The visual graphs were created by 3DV8. The block arrows in Figure 6 show how the data from the logs was graphed. The origin of the graph is in the bottom corner. Unless indicated otherwise all figures will show the Source IP address on the X-axis increasing numerically, the Destination IP address will increase numerically on the Z-axis, and the Y-axis, which is the height of the graph, will show the Time/Date stamp increasing. 3DV8 allows up to three variables to be added to the height of the graph but in every case only a single variable was used.

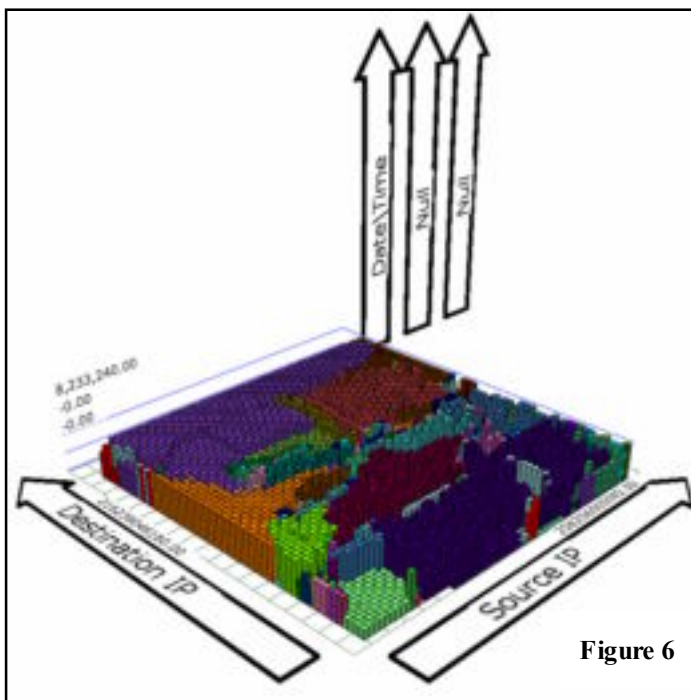


Figure 6

List of files used

Data files from January 4-8th, 2002, were selected to provide a sample analysis. The files included: Snort alert logs, scan logs, and Out of Specification packet (OOS) logs.

Table 1

Dates	Alerts	Scans	OOS
January 4, 2002	alert.020104	scans.020104	oos.Jan.4.2002
January 5, 2002	alert.020105	scans.020105	oos.Jan.5.2002
January 6, 2002	alert.020106	scans.020106	oos.Jan.6.2002
January 7, 2002	alert.020107	scans.020107	oos.Jan.7.2002
January 8, 2002	alert.020108	scans.020108	oos.Jan.8.2002

Top 10's

Alerts

The Top 10 alert signatures detected by Snort-Snarf for the dates of January 4, 2002 through January 8, 2002, account for approximately 99% of all alerts. Table 2 shows the number of alerts, the number of sending and receiving IP addresses, and percentages.

Table 2

Signature	Alerts	Sources	Dests	%
ICMP traceroute	33,556	5	3	29
connect to 515 from inside	22,409	47	2	19
spp_http_decode: IIS Unicode attack detected	22,373	78	342	19

Signature	Alerts	Sources	Dests	%
MISC Large UDP Packet	15,419	15	14	13
SNMP public access	13,338	16	137	12
INFO MSN IM Chat data	2,362	53	54	2
INFO – ICQ Access	1,915	2	42	2
High port 65535 udp – possible Red Worm – traffic	1,848	28	101	2
ICMP Router Selection	1,285	124	1	1
SMB Name Wildcard	1,078	38	33	<1
Total of All Alerts	119,230			~99%

The top 10 Source IP's generating alerts, and the top 10 Destination IP's receiving them are shown in Table 3 and Table 4. Also included is the number of alerts associated with each listed by rank, and the number of signatures each address is associated with, and finally the number or address of other associated IP addresses.

Notice IP address **256.256.150.198**, which is shown in red. It appears on both Top 10 lists, as is associated with more IP addresses than any other IP. Also note 256.256.5.202 has one signature for 33548 alerts, and is associated with only **256.256.5.1** shown in blue, which also the #1 Destination IP.

Top 10 Alert Source IPs

Table 3

Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
#1	33548 alerts	256.256.5.202	1 signatures	256.256.5.1
#2	5,724 alerts	256.256.153.114	4 signatures	(28 destination IPs)
#3	4,580 alerts	256.256.153.146	1 signatures	256.256.150.198
#4	3,767 alerts	256.256.70.177	2 signatures	(27 destination IPs)
#5	2,943 alerts	256.256.88.181	5 signatures	(29 destination IPs)
#6	2,605 alerts	216.106.166.211	1 signatures	256.256.153.45, 256.256.153.46
#7	2,294 alerts	256.256.150.198	1 signatures	(98 destination IPs)

Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
#8	2,157 alerts	256.256.153.197	2 signatures	(22 destination IPs)
#9	2,153 alerts	256.256.150.41	1 signatures	256.256.152.109
#10	2,087 alerts	256.256.153.220	1 signatures	256.256.152.109

Top 10 Alert Destination IP's

Table 4

Rank	Total # Alerts	Destination IP	# Signatures triggered	Originating sources
#1	33,548 alerts	256.256.5.1	1 signatures	256.256.5.202
#2	22,408 alerts	256.256.150.198	1 signatures	(46 source IPs)
#3	5,960 alerts	256.256.152.109	1 signatures	(4 source IPs)
#4	3,476 alerts	211.115.213.202	1 signatures	(4 source IPs)
#5	3,015 alerts	256.256.153.46	2 signatures	(4 source IPs)
#6	2,437 alerts	211.32.117.26	1 signatures	(9 source IPs)
#7	2,330 alerts	256.256.153.154	2 signatures	(4 source IPs)
#8	1,898 alerts	256.256.88.167	2 signatures	203.248.242.22
#9	1,729 alerts	211.32.117.27	1 signatures	(3 source IPs)
#10	1,536 alerts	256.256.88.165	4 signatures	(3 source IPs)

Scans

The Top 10 Scans include two signatures that account for nearly 100% of the scans reported by Snort-Snarf. Table 5 displays all of the Top 10 scans by rank, including the number of alerts reported, the number of sending and receiving IP addresses, and percentages. UDP scans were considered separately because Snort-Snarf couldn't process them together so this is actually a "Top 11" list.

It is unlikely that 342 different hosts intentionally scanned 20,469 hosts using the same UDP scan signature over the course of one week in the same network. The same goes for TCP SYN scan logs. This is evidence of a poorly tuned Intrusion Detection System. In the interest of saving time, the UDP and TCP scan traffic logged by these rules will not be examined in-depth, but the Top 5 Source and Destinations are reported. Any true scanning activity will be shrouded

in a high number of false positives, thus the focus of the remainder of this analysis will be on the Alert categories with due attention being given to the individual IP's as needed.

Table 5

Signature	# Alerts	# Sources	# Dests	%
UDP scan	808,932	342	20,469	84.1
TCP *****S* scan	152,699	383	16,584	15.8
TCP ***** scan	104	22	7	.081
TCP ****P*** scan	60	21	4	~0
TCP *2UA**** scan	10	10	2	~0
TCP ***A*R*F scan	6	2	2	~0
TCP *2*A**S* scan	6	1	1	~0
TCP *2**PR*F scan	5	4	3	~0
TCP *2UAP*** scan	4	3	2	~0
TCP *2***R** scan	4	4	2	~0
TCP *2****SF scan	4	3	2	~0
Total of All Scans	961,834			99.89

The top 10 Source IP's generating scans, and the top 10 Destination IP's receiving them are shown in Table 6 and Table 7, which also includes the number of alerts listed by rank, and the number of signatures each address is associated with, and finally the number or address of other associated IP addresses.

The TCP *****S* scan was the only signature for all Top 10 Sources, and 9 of the Top 10 Destinations with a minimum of 938 alerts each! This makes it very difficult to determine which of these scans are the most critical to examine since they all have the same potential to be either true or false positives. All of the Top Source IP's are internal to MY.NET, and 6 of the Top Destination IP's are external with three of those living on the same subnet.

Top 10 Scanning Source IPs

Table 6

Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
#1	12,411 alerts	256.256.150.143	1 signatures	(672 destination IPs)

Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
#2	6,515 alerts	256.256.88.162	1 signatures	(2,815 destination IPs)
#3	4,295 alerts	256.256.150.165	1 signatures	(1,031 destination IPs)
#4	3,959 alerts	256.256.153.148	1 signatures	(1,238 destination IPs)
#5	2,870 alerts	256.256.153.46	1 signatures	(719 destination IPs)
#6	2,829 alerts	256.256.153.143	1 signatures	(851 destination IPs)
#7	2,731 alerts	256.256.88.181	1 signatures	(701 destination IPs)
#8	2,554 alerts	256.256.153.152	1 signatures	(296 destination IPs)
#9	2,517 alerts	256.256.153.211	1 signatures	(472 destination IPs)
#10	2,499 alerts	256.256.153.45	1 signatures	(712 destination IPs)

Top 10 Scanning Destination IPs

Table 7

Rank	Total # Alerts	Destination IP	# Signatures triggered	Originating sources
#1	2,143 alerts	131.118.254.39	1 signatures	(190 source IPs)
#2	2,092 alerts	131.118.254.38	1 signatures	(188 source IPs)
#3	1,385 alerts	256.256.11.4	1 signatures	(104 source IPs)
#4	1,362 alerts	256.256.153.111	2 signatures	(4 source IPs)
#5	1,322 alerts	256.256.5.83	1 signatures	(23 source IPs)
#6	1,062 alerts	203.165.89.8	1 signatures	256.256.150.220, 256.256.150.143
#7	1,060 alerts	61.210.89.59	1 signatures	256.256.150.220, 256.256.150.143
#8	1,051 alerts	131.118.254.40	1 signatures	(161 source IPs)
#9	961 alerts	62.179.147.165	1 signatures	256.256.150.220, 256.256.150.143
#10	938 alerts	256.256.104.139	1 signatures	256.256.5.92

Top 10 Talkers in MY.NET

Finally, no list of Top 10's is complete without the Top 10 Talkers list. Table 8 show the Top Talkers ranked by number of alerts, with signatures associated, and other IP address associated with each. This list combines both Alert and Scanning (shown in green) hosts or Destination hosts, shown in red or blue, as reported by Snort-Snarf.

Table 8

Rank	Total # Alerts	IP Address	# Signatures triggered	Other IPs
#1	33,548 alerts	256.256.5.202	1 signatures	256.256.5.1 Dest.
#2	33,548 alerts	256.256.5.1	1 signatures	256.256.5.202
#3	22,408 alerts	256.256.150.198	1 signatures	(46 source IPs)
#4	12,411 alerts	256.256.150.143	1 signatures	(672 destination IPs)
#5	6,515 alerts	256.256.88.162	1 signatures	(2815 destination IPs)
#6	5,960 alerts	256.256.152.109	1 signatures	(4 source IPs)
#7	5,724 alerts	256.256.153.114	4 signatures	(28 destination IPs)
#8	4,580 alerts	256.256.153.146	1 signatures	256.256.150.198
#9	4,295 alerts	256.256.150.165	1 signatures	(1031 destination IPs)
#10	3,959 alerts	256.256.153.148	1 signatures	(1238 destination IPs)

Top10 Alerts Signatures

ICMP traceroute

Description of Attack

ICMP traceroute is a windows command designed to show the path of all routing nodes traversed between two points on the network. There are a total of 5 Source hosts for three Destination hosts for a total of 33555 alerts shown in Table 9 and Table 10.

Table 9

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
256.256.5.202	33,548	33,548	1	1
256.256.88.179	4	4	1	1
256.256.150.121	2	5	1	2
256.256.88.139	1	1	1	1
256.256.88.167	1	88	1	10

Table 10

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
256.256.5.1	33,548	33,548	1	1
256.256.1.3	6	20	2	9
256.256.88.129	2	2	2	2

Address 256.256.5.202, which is the #1 talker in MY.NET accounts for a full 29% of all alerts in this analysis. It is chirping traceroute packets once every ten seconds for the entire logging period to 256.256.5.1, the #1 Alert Destination. The Snort-sort log demonstrates this below. The reason for this traffic is unknown, but 256.256.5.1 might be a border router since many administrators use .1's for routers, and it is possible the .5.202 host is automatically mapping portions of the internet via the .5.1 routing device. However, this might also be a heartbeat, quality of service control, or a misconfiguration.

- 01/04-16:44:38.095023 256.256.5.202: -> 256.256.5.1:
- 01/04-16:44:48.090718 256.256.5.202: -> 256.256.5.1:
- 01/04-16:44:58.087134 256.256.5.202: -> 256.256.5.1:
- 01/05-00:35:07.622197 256.256.5.202: -> 256.256.5.1:
- 01/05-00:35:17.624925 256.256.5.202: -> 256.256.5.1:
- 01/05-00:35:37.616308 256.256.5.202: -> 256.256.5.1:

Defensive Recommendations

If this traffic is not an expected QoS tool, then the 256.256.5.202 host should be examined for suspicious files or running processes, and the primary user questioned regarding the traceroute activity. A filter could be applied to the border router/firewall to not allow ICMP type 30, code 0 packets to leave the network if this traffic is deemed unnecessary for the MY.NET domain.

Correlation

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

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

connect to 515 from inside

Description of Attack

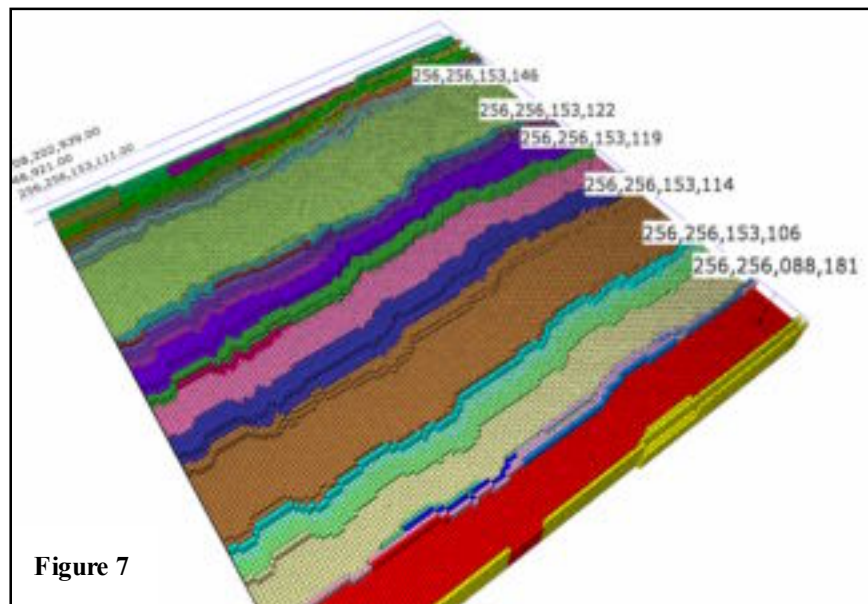
TCP port 515 is typically associated with print spooler services. In this case there are a total of 47 source hosts connecting to two internal destination hosts in Table 11. This signature is a bit confusing since it is usually expected that internal hosts will connect to a networked printer at port 515. It may be that this Snort rule needs to be tuned to ignore internal source traffic, and look for *connect to 515 from outside*.

Table 11

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
--------------	----------------	------------------	--------------	----------------

256.256.150.198	22,408	22,408	46	46
256.256.153.111	1	11	1	3

The 256.256.150.198 address is #3 Top Talker, appears to have a “regular traffic” traffic pattern as expected by a printer. There are various internal MY.NET hosts across varied ports connecting to it in short bursts of time. This “regular pattern” is seen here in this 3DV8 plot of the traffic. 256.256.150.198 is probably a shared printer. If this is true, then this is a false positive.



However there were periods of sustained between a single static source IP's and the “printer” shown in Figure 7 as wide bands of color. 256.256.150.198 is also the source of 2294 instances of SNMP public access primarily pointed at 256.256.151.114:161, from port 1025 (blackjack). This port is also used by the following Trojans: NetSpy, Maverick's Matrix, and RemoteStorm.

If 256.256.150.198 has SNMP services turned on, then it appears to be set to the default community string. See SNMP public access below.

The second address 256.256.153.111 appears to be the subject of inquiry for another host who is using NMAP.

Table 12

01/07-20:05:14.333597 [**] ICMP Echo Request Nmap or HPING2 [**] 256.256.253.10 -> 256.256.153.111
01/07-20:05:14.845068 [**] SUNRPC highport access! [**] 256.256.253.10:48921 -> 256.256.153.111:32771
01/07-20:05:15.017117 [**] connect to 515 from inside [**] 256.256.253.10:48921 -> 256.256.153.111:515

Defensive Recommendations

If 256.256.150.198 is a network print device then the print logs should be routinely inspected for any unusual activity. If it is not then the host should be inspected for signs of a possible compromise. SNMP services should be patched and secured from further public string access.

The operator of the 256.256.253.10 host should be counseled regarding the use of NMAP on the MY.NET network if this is against policy.

The alert rule generating this signature might produce fewer false positives, which should be redundant to the print logs anyway, if it was rewritten to ignore internal traffic, and focus on external connection attempts only. The new rules might look something like this:

```
alert tcp $EXTERNAL_NET any -> $MY_NET 515 (msg:"connect to 515");
alert udp $EXTERNAL_NET any -> $MY_NET 515 (msg:"connect to 515");
```

Correlation

http://www.sans.org/y2k/practical/Mike_Bell_GCIA.doc

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

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

spp_http_decode: IIS Unicode attack detected

Description of Attack

There were a total of 78 source hosts connecting to 342 destinations for a combined total of 22,373 alerts. This alert is generated by Snort's http_decode packet pre-processor. The Snort User's Manual states it is used to decode "HTTP URI strings and convert their data to non-obfuscated ASCII strings." Unicode Vulnerabilities are listed as SANS' #1 Windows security vulnerability in the Top20. CVE-2000-884 states there is a vulnerability in IIS4.0 and IIS5.0, which allows remote hackers to read documents outside of the web root, and potentially execute commands via malformed URL's containing Unicode. In Figure 8 the graph's origin is located in the far left corner and the Source IP's are displayed in color.

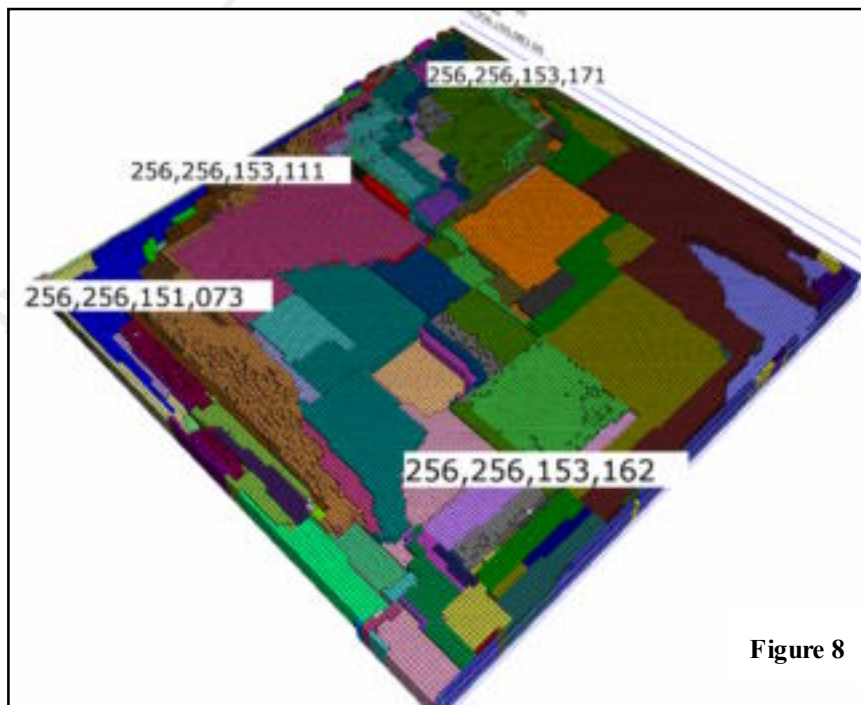


Figure 8

Notice the flat areas of solid color vs. the jagged stepped areas of light green, gray, maroon, and light brown identified by their IP's. The IP's depicted by the flat colors had all of their alerts on the same day. The flagged IP's showed the most variation over time and had numerous alerts spread out throughout the week. Notice the pronounced "square block" pattern of some colors. This may suggest communication between a static source IP and multiple hosts in a subnet address range. This would be consistent with the targeting scheme of the Code Red worm, but could also indicate a scan of a subnet range if the scan triggered this alert.

Figure 9 is the same graph with the Destination IP's displayed in color. Notice the contour of the graph has not changed. Unfortunately for 3DV8, there were 342 Destination IP's in this graph, but it can only display up to 128 color groups. Still, the purple, and gray color blocks seem to

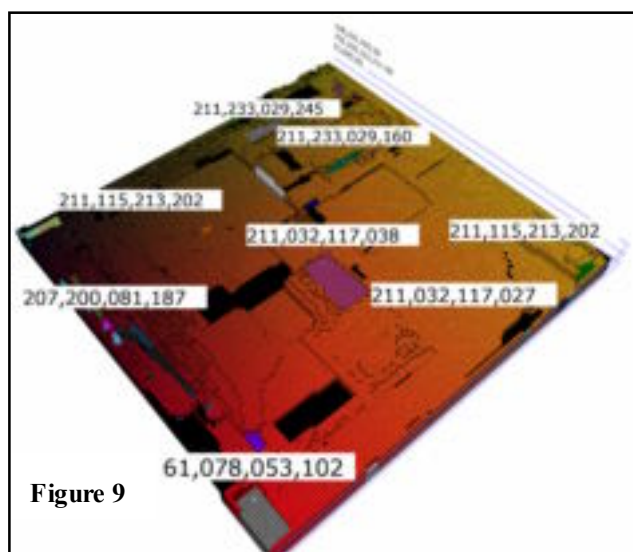


Figure 9

indicate that there was sustained traffic between these static IP's over the course of a single day. The flagged Destination IP's represent the contoured regions they point to.

Defensive Recommendations

The MY.NET.153.0 subnet seems to be frequented by this signature. A web farm may live at this subnet. If so, the entire web farm should be investigated for further evidence of the Code Red worm, since Snort-Snarf reported three of the four Source addresses above as destination of it. The stepped areas of Figure 8 may represent infected web servers, which have not been attended too for several days.

1 signature is present for **256.256.153.111** as a source

- 1197 instances of [spp_http_decode: IIS Unicode attack detected](#)

6 different signatures are present for **256.256.153.111** as a destination

- 1 instances of [SUNRPC highport access!](#)
- 1 instances of [connect to 515 from inside](#)
- 1 instances of [ICMP Echo Request Nmap or HPING2](#)
- 1 instances of [INFO - Possible Squid Scan](#)
- 3 instances of [NMAP TCP ping!](#)
- 4 instances of [SCAN Proxy attempt](#)

2 different signatures are present for **256.256.153.171** as a source

- 96 instances of [spp_http_decode: CGI Null Byte attack detected](#)
- 1121 instances of [spp_http_decode: IIS Unicode attack detected](#)

1 signature is present for **256.256.153.171** as a destination

- 1 instances of [High port 65535 udp - possible Red Worm - traffic](#)

3 different signatures are present for **256.256.153.162** as a source

- 1 instances of [High port 65535 udp - possible Red Worm - traffic](#)
- 2 instances of [ICMP Destination Unreachable \(Protocol Unreachable\)](#)

- 350 instances of [spp_http_decode: IIS Unicode attack detected](#)
- 4 different signatures are present for **256.256.153.162** as a destination
 - 3 instances of [INFO - Possible Squid Scan](#)
 - 14 instances of [SCAN Proxy attempt](#)
 - 22 instances of [High port 65535 udp - possible Red Worm - traffic](#)
 - 72 instances of [Watchlist 000220 IL-ISDNNET-990517](#)

Correlation

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

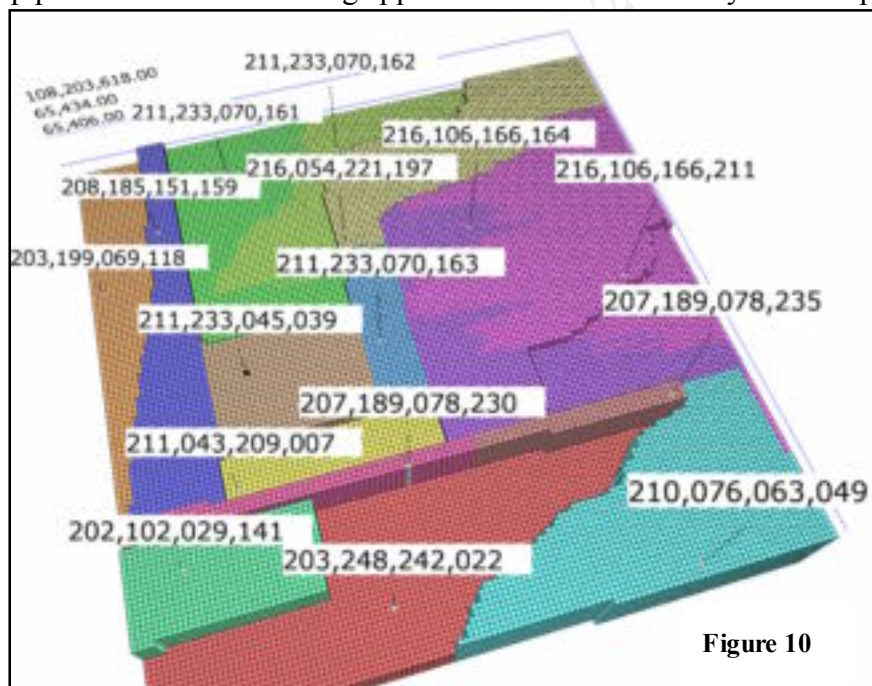
MISC Large UDP Packet

Description of Attack

There were 15 source hosts connecting to 14 destinations for a total of 15,419 alerts. This alert is concerned with any incoming UDP traffic a large payload. The Snort rule for this traffic obtained from <http://www.snort.org/snort-db/sid.html?id=499>, looks for a data gram size larger than 800 bytes:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Large ICMP
Packet"; dsize: >800; reference:arachnids,246; classtype:bad-unknown;
sid:499; rev:1;)
```

A host receiving this packet might suffer from a system crash or depleted bandwidth in a small pipe. Some load balancing applications also use 1500 byte ICMP packets to measure latency along alternate paths.



Fourteen of those source hosts are pointed out in Figure 10. Only 216.106.166.164 and 216.106.166.211 had alerts on two different days according to this graph. Snort-Snarf confirms this and shows that they were they also the only hosts speaking to more than one Destination host. Neither was the source or destination of any other alert signatures. The 216.106.166.211 address ranks #6 on the Top 10

Alert Source list. Four other source hosts were destinations of *ICMP Fragment Reassembly Time Exceeded* alerts probably related to the same large UDP packets.

In this situation, large UDP packets sent out by these sources were being fragmented to conform to the networks MTU (maximum transmit size), and some packets were lost or dropped in transit

to the receiving hosts. Those receiving hosts then sent an ICMP error code 11,1 (ICMP Fragment Reassembly Time Exceeded) back. This behavior is a normal function of TCP/IP, but since each host must retain information for each fragment received, an attacker can exploit this by consuming the available resources of the destination host and cause a denial of service condition. The external hosts receiving the ICMP Fragment reassembly Time Exceeded alert were:

Table 13

211.233.45.39	211.233.70.161
211.233.70.162	216.54.221.197

Defensive Recommendations

Applying system patches should harden the destination hosts from this exploit. The operators of these systems should be interviewed for signs of system slow downs or crashing. It would be advisable to place the address in Table 13 on a watch list for further activity.

Correlation

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

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

SNMP public access

Description of Attack

There were 16 source hosts talking with 137 destination hosts, across 892 source ports, for a total of 13,338 alerts. Simple Network Management Protocol uses TCP/UDP ports 161 for communication between SMNP Managers and Agents. SNMP is used to configure various networked devices such as computers, printers, routers, and perform data collection of the same. An unencrypted password called a “community string,” is the only form of authentication, which has a default setting of “public” in most devices. The community string is vulnerable to sniffing, replay attacks, brute force and dictionary guessing. If an attacker can sniff SNMP traffic it will reveal a great deal of information about the network’s makeup, devices, and services offered which will help in the planning of more focused attacks. If they have the community string, then they can cause all sorts of mischief including: denials of service, DNS cache poisoning, and replay attacks. [The Twenty Most Critical Internet Security Vulnerabilities](#), Version 2.502 January 30, 2002, puts default SNMP strings as #7 on its list.

The origin of Figure 11 below is in the far left corner and six colored IP’s of interest are flagged. The first thing to notice about this graph is the varied textures apparent in it. The first pattern includes light brown, brown, and green colors, which show a stepped pattern progressing up over time in bands in the top third of the graph. This area is populated by 256.256.150.041, which is #9 on the Top Alert Source IP list; 256.256.150.245, which is #15 on the Top Alert Source list; 256.256.153.220, which is #10 on the Top Alert Source list.

The next pattern is mainly in the spiky middle section of the graph consisting of the orange and purplish colors, where 256.256.088.240 and 256.256.150.198, seem to be talking on every date available “in the same place.”

The 256.256.150.198 host is in four different Top 10 lists: Alert Source & Destination, Scanning

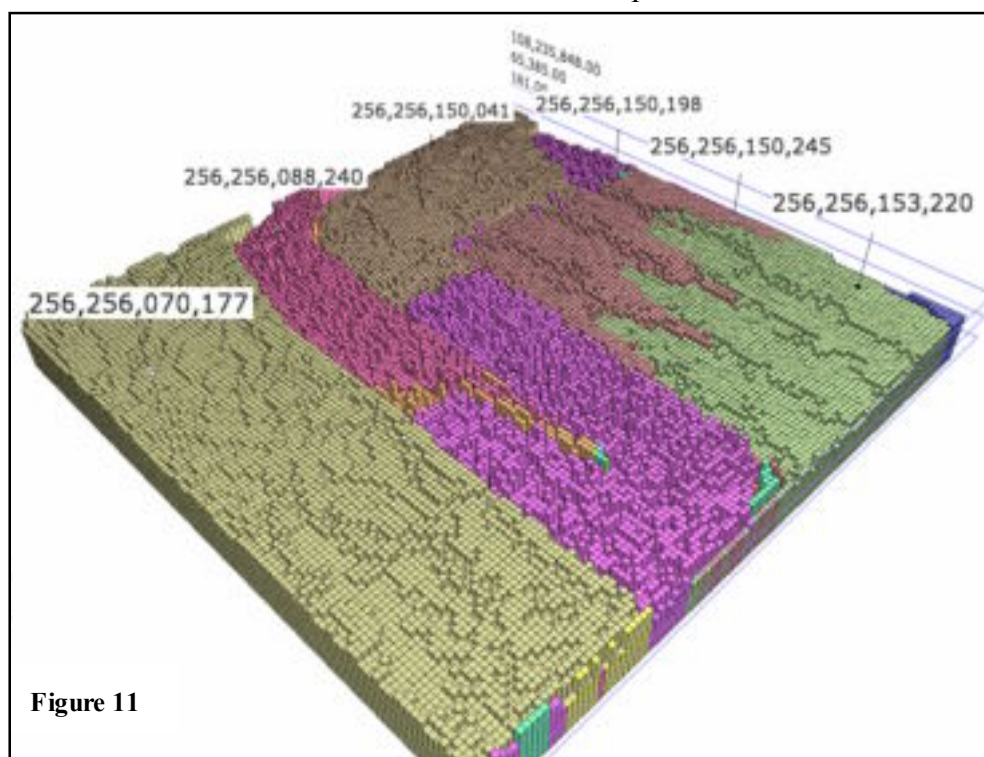


Figure 11

Destination, and is #3 on the Top Talkers list! It only has this one signature associated with it as a source, but it counts for 2294 alerts across 46 destinations, which are seen below in Figure 12. It is also the destination of 22,408 instances of connect to 515 from inside above. This information

supports the theory that it 256.256.150.198 is a printer, and it is misconfigured, or in this case still configured with the default “public” community string.

The third pattern is present in the lower third of the graph, which is predominantly 256.256.070.177 and it accounts for 3754 of the alerts of this signature, across 27 destinations on January 7-8th. It is using 2 different ports, which is visible in Figure 13, which suggests this host is attached to a Cognex brand In-Sight visual sensor.

- Port 1069 COGNEX-INSIGHT, speaking to port 161 on multiple hosts of the MY.NET.5.0 subnet.
- Port 3072 ContinuumStor Monitor speaking to only too 256.256.5.96

It is also associated with 13 instances of the SMB Name Wildcard signature.

All of these hosts are suspected misconfigured SNMP devices, since the alerting rule is believed to have been specific to a port, and payload content match.

Figure 12 below is the same plot, but now the colors represent the Destination IP address. Here we see four dominate destination addresses. Notice the areas with no block color pattern at all. These are the same areas as the 256.256.150.98 address, but now it's evident that it favors 256.256.151.114 shown in purple. This address is #13 on the Top Alert Destination list, and is also associated with 3 instances SMB Name Wildcard as a destination. The 256.256.151.114 host is receiving a great deal of SNMP information from 256.256.150.198, which may indicate an information leak in progress unless this relationship is administrative.

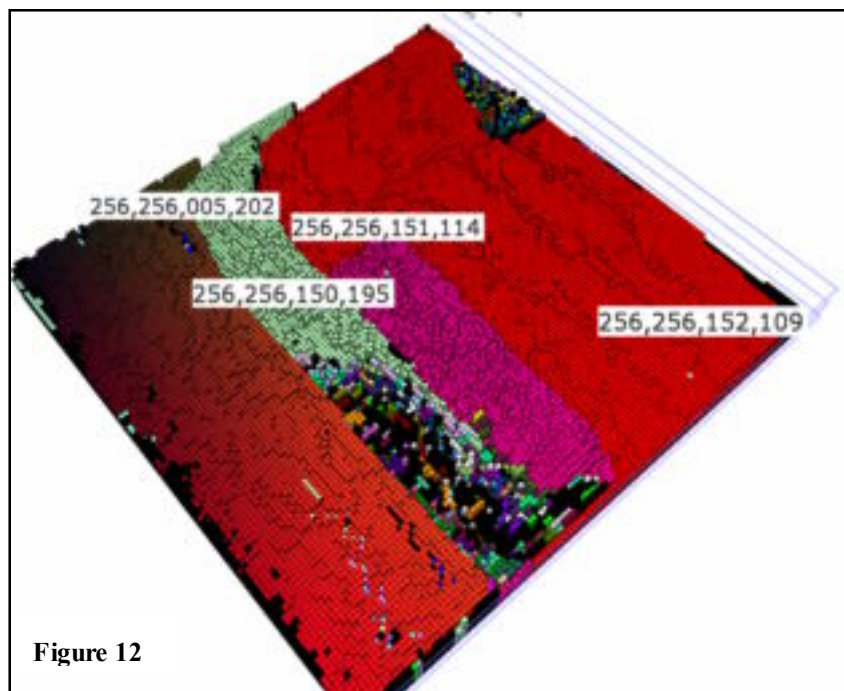


Figure 12

Snort-Snarf confirms that 256.256.152.109 is the sole destination of 256.256.150.141, 256.256.153.220, and one of two destinations for 256.256.150.245. 256.256.152.109 is the #6 Top Talker in MY.NET, and has 5960 alerts of only this signature from four different IP addresses. It is not a source address. In this case it would be better to call 256.256.152.109 a Top Listener. But what it's listening too, and what its doing with the information is unknown. Since the window of time in this analysis is a

brief five days, it is possible that 256.256.152.109 is storing information for a later transmission.

The 256.256.150.195 address is the sole recipient of the 256.256.088.240 alerts. It is not a source address. The 256.256.005.202 address is flagged in Figure 12 since it is the #1 Alert Source IP, but it only has nine instances SNMP public access from 256.256.070.171.

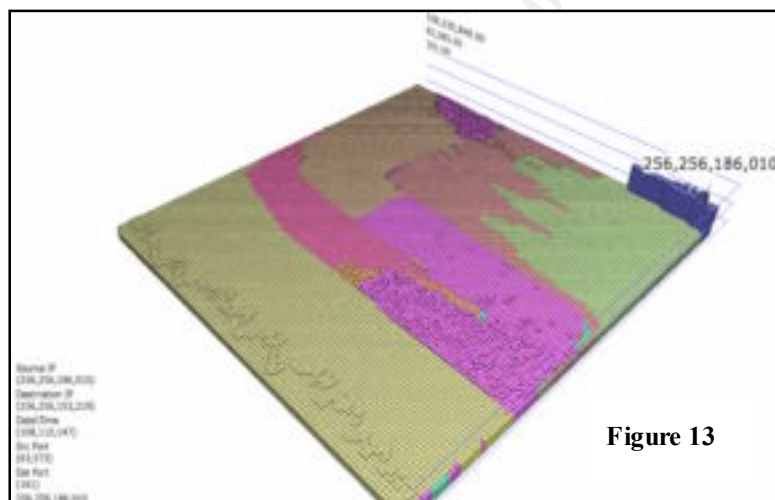


Figure 13

Figure 13 is meant to point out an anomaly in the ports. Here the Y-axis represents source ports. The event could be coincidental, or by design. Most all of the source ports recorded in the logs of this signature are relatively low ephemeral ports with the exception of 256.256.186.010 shown here. Its source port varied from 4339 to 65385 as if it was port scanning, but always its destination was port 161 of 256.256.153.219. It was the source of 71 alerts for this

signature. It was not a destination host. This behavior was seen on January 4th-8th. The 256.256.153.219 host collects 97 instances of the SNMP public access signature from three hosts. The other two were 256.256.150.198 and 256.256.150.205. Neither used a high source port like 256.256.186.010.

Also its interesting to notice that 256.256.150.198 again shown in purple, has a numerous source ports which presumably speak to 161 as a destination. This might indicate 256.256.150.198 was scanning multiple hosts from random ports, looking for port 161.

Defensive Recommendations

All unnecessary SNMP services should be disabled, and those that are required should be patched against the latest vulnerability published in Feb. 2002, and secured with non-default community strings. 256.256.150.198 should be given special attention, since it appears to be a print device, which is scanning for port 161.

Table 14 shows the addresses that should be inspected for signs of SNMP misconfiguration or compromise.

Table 14

256.256.070.171	256.256.150.041	256.256.150.198	256.256.150.245
256.256.151.114	256.256.152.109	256.256.153.220	

It is curious to note 256.256.070.171 may have a camera attached to it, and is sending targeted traffic to hosts in the MY.NET.5.0 subnet. If this is unexpected then a meeting with its primary user is in order to secure it and determine the purpose traffic. Also curious is the nature of the port 3072 traffic directed solely at 256.256.5.96 from this same host. Its purpose is unknown, but may be an information leak.

Correlation

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

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

INFO MSN IM Chat data

Description of Attack

There were 2,362 alerts from 53 hosts talking to 54 hosts. All legitimate MSN IM traffic is transmitted between hosts via Microsoft's proxy servers so the likelihood of this traffic being spoofed is low. This traffic is benign, but may be against network policy. The biggest danger is unencrypted payloads and nosey administrators, or other eavesdroppers on the network with a packet sniffer. This traffic could be used indirectly to finger print operating systems, and leak other kinds of information via social engineering if the user isn't careful. It is a great cover for a covert channel if packets were crafted to resemble IM traffic. The Snort rule which generated this detect might look something like this:

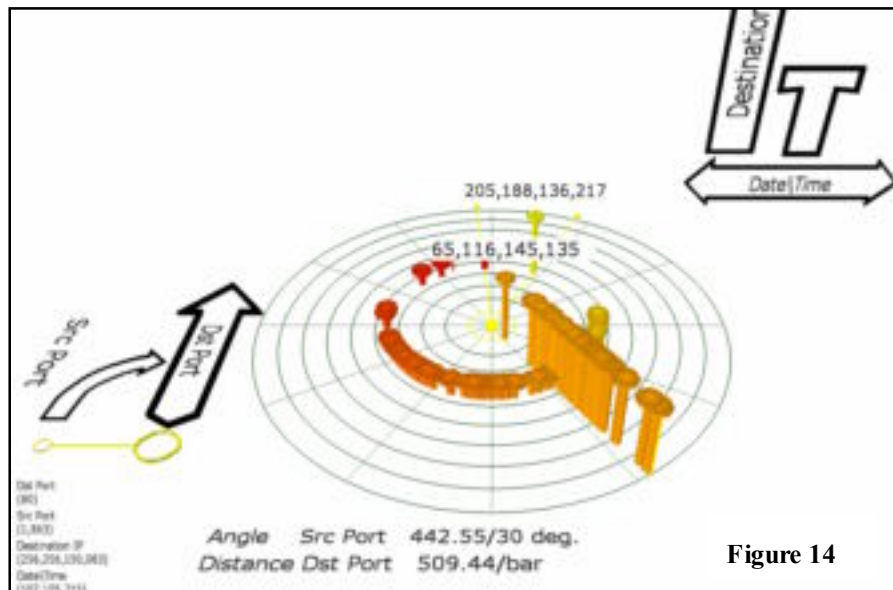
```
alert tcp any any <-> any 1863 (msg:"INFO msn chat
access";flags: A+; content:"|746578742F706C61696E|"; depth:100;
classtype:not-suspicious; sid:540; rev:2;)
```

All alerts consisted of internal MY.NET hosts using ephemeral ports and external hosts on port 1863 that fall into the range of 64.4.12.150-191. An ARIN WHOIS host lookup reports that this range belongs to Hotmail.com, which strongly suggests the traffic is legitimate IM traffic and false positives. However two source IP's, flagged in the radial nails graph Figure 14,

65.116.145.135 and 205.188.136.217, are not from either Hotmail or MY.NET's networks. Here we see that nail height is the destination IP, the source port is the angle from the yellow diameter lines, and the distance from center represents the destination port. The two flagged nails stick out because they are placed away from, and out of line with the other nails similar in height and destination port.

The 205.188.136.217 alert might be innocent web traffic, which happens to use port 1863:

```
01/07-17:36:35.448356 [**] INFO MSN IM Chat data [**] 205.188.136.217:80 ->
256.256.150.206:1863
```



The 65.116.145.135 host's IM chat alert is directed toward the 256.256.150.083 address, which is also the recipient of 7 alert signatures, and 2 scans which suggests it is an endangered web server at best. The alerts come about five hours apart, and there is no sign of a stimulus, or IM response from 256.256.150.83 as would be expected. This could be a covert channel

communication and is worthy of further investigation.

Table 15

01/07-05:33:24.204462 [**] INFO MSN IM Chat data [**] 65.116.145.135:1863 -> 256.256.150.83:80
01/07-10:57:15.149547 [**] INFO MSN IM Chat data [**] 65.116.145.135:1863 -> 256.256.150.83:80

7 different signatures are present for **256.256.150.83** as a destination

- 1 instances of [WEB-FRONTPAGE_vti_rpc access](#)
- 1 instances of [WEB-IIS_vti_inf access](#)
- 1 instances of [IDS552/web-iis_IIS ISAPI Overflow ida nosize](#)
- 2 instances of [INFO MSN IM Chat data](#)
- 5 instances of [WEB-CGI_formmail access](#)
- 52 instances of [WEB-MISC Attempt to execute cmd](#)

2 different signatures are present for **256.256.150.83** as a destination

- 9 instances of [TCP *2UA**** scan](#)
- 303 instances of [TCP *****S* scan](#)

Defensive Recommendations

The 256.256.150.83 box should be taken off-line and investigated, rebuilt if needed, and patched against further Web attacks. The Snort rule which generated this alert should be modified to screen out false positives by adding the content string in the rule above, and adding a Hotmail_NET\$ variable address group in the snort.conf file.

```
var Hotmail_NET 64.4.0.0/18
```

```
alert tcp MY_NET any <-> !$Hotmail_NET 1863 (msg:"INFO msn chat  
access"; flags: A+; content:"|746578742F706C61696E|"; depth:100;  
classtype:not-suspicious; sid:540; rev:2;)
```

Correlation

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

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

INFO - ICQ Access

Description of Attack

There were 1915 alerts stemming from 2 source hosts to 42 destination hosts. ICQ is another online chat client hosted by America Online. A possible snort rule generating these alerts is:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"INFO ICQ access"; flags:  
A+; content: "User-Agent\:ICQ"; classtype:misc-activity; sid:541; rev:3;)
```

The alerts hosts are:

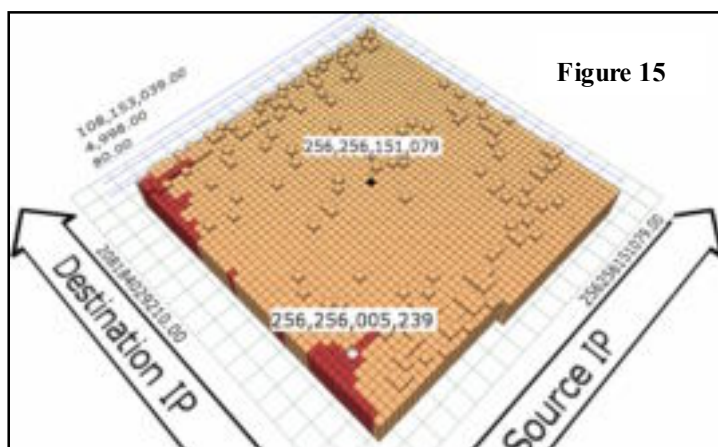
Table 16

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
256.256.151.79	1,858	1,883	42	46
256.256.5.239	57	57	6	6

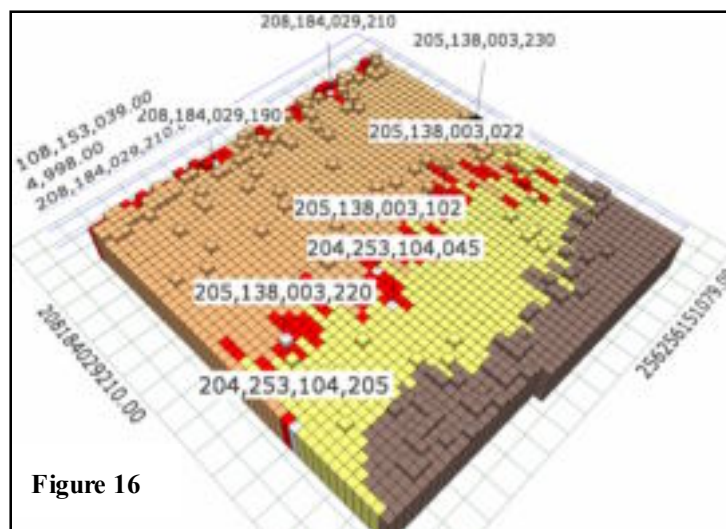
Figure 15 is a simple plot showing the Source IP's by color. The 256.256.151.079 IP has the predominate number of alerts.

This traffic is interesting because the AOL address ranges which ICQ are expected to use are:
64.12.0.0 - 64.12.255.255,
152.163.0.0 - 152.163.255.255,
and 205.188.0.0 -
205.188.255.255.

But what we find in Figure 16 with the Destination IP's grouped in color by address ranges is that a number of IP's in red fall outside



these address ranges. Therefore either the ARIN WHOIS information is out of date, or these detects are not legitimate ICQ traffic, but possibly another form of stealthy traffic. Although it's not apparent in these figures, Snort-Snarf confirms that only 256.256.151.079 was in communication with the red-tagged IP's.



This address is associated with 3 different alerts.

- 12 instances of [ICMP Router Selection](#)
- 13 instances of [INFO Possible IRC Access](#)
- 1858 instances of [INFO - ICQ Access](#)

Table 17 displays the destinations outside of the expected ICQ address range.

Table 17

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
208.184.29.190	18	18	1	1
208.184.29.210	16	16	1	1
205.138.3.62	15	15	1	1
205.138.3.230	14	14	1	1
205.138.3.22	13	13	1	1
205.138.3.42	10	10	1	1
205.138.3.220	10	10	1	1
205.138.3.82	10	10	1	1
205.138.3.142	8	8	1	1
205.138.3.102	8	8	1	1
204.253.104.205	7	7	1	1
204.253.104.220	5	5	1	1
204.253.104.15	2	2	1	1

208.184.29.90	2	2	1	1
208.184.29.110	2	2	1	1
208.184.29.70	2	2	1	1
204.253.104.45	1	1	1	1
206.65.183.140	1	1	1	1
206.65.183.25	1	1	1	1
206.65.183.40	1	1	1	1

Defensive Recommendations

The ARIN WHOIS lookup of the ICQ network needs to be verified. Then 256.256.151.79 should be examined for signs of compromise, and its primary user(s) should be questioned regarding their use of ICQ and IRC clients. A number of Trojans are known to use ICQ as a communication vehicle, and it has a number of known exploits posted in the CVE entries in Table 18. The network policy regarding instant chat programs should also be reviewed. The Snort rule could be modified to examine only traffic NOT sent to the known ICQ address ranges.

```
var ICQ_NET [64.12.0.0/23,152.163.0.0/24,205.188.0.0/24]

alert tcp $MY_NET any <-> !$ICQ_NET any (msg:"INFO ICQ access"; flags:
A+; content: "User-Agent\.:ICQ"; classtype:misc-activity; sid:541;
rev:3;)
```

Table 18

CVE-1999-0474	CAN-1999-1440	CAN-1999-1418	CAN-1999-1342
CVE-2000-0552	CAN-2000-0046	CAN-2001-0367	CAN-2000-1078
CAN-1999-1289	CAN-2000-0564	CAN-2002-0028	

Correlation

None found at <http://www.google.com/>.

High port 65535 udp – possible Red Worm – traffic

Description of Attack

There were 28 Source hosts, and 101 destination hosts generating 1,848 alerts. Jeff Holland has an excellent write up regarding it in his practical and this section borrows heavily from him. The Red Worm, also known as the Adore Worm, was first discovered in April of 2001. It scans the Internet looking for vulnerable Linux hosts with any of these well-known exploits: LPRng, rpc-statd, wu-ftpd and BIND. Incidents.org has posted a security advisory of the worm written by Matt Fearnow from the SANS Institute, and William Stearns of the Dartmouth Institute for Security Technology Studies here: <http://www.incidents.org/react/adore.php>. Basically the worm will replace the (ps) system binary file with a trojaned version, and will attempt to send

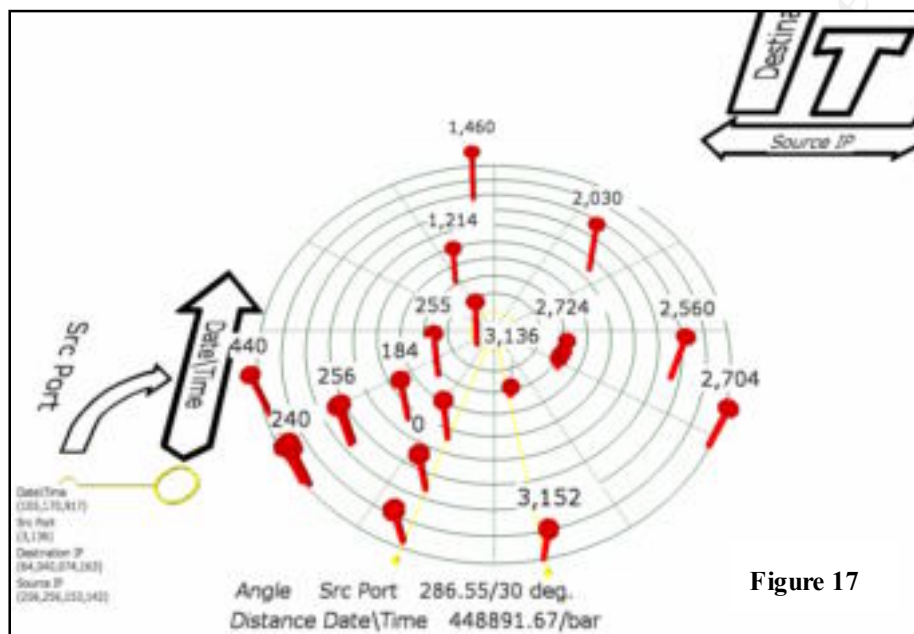
email to these addresses: adore9000@21cn.com, adore9000@sina.com, adore9001@21cn.com, and adore9001@sina.com.

It will attempt to send the following information:

- /etc/ftpusers
- ifconfig
- ps -aux (using the original binary in /usr/bin/adore)
- /root/.bash_history
- /etc/hosts
- /etc/shadow

The Adore Worm listens at TCP and UDP port 65535. The RC1 Trojan also listens at port 65535. The worm is not known to affect Windows systems, but the SANS advisory notes that

the code is easily modified so it should not be assumed this Trojan will always be found at port 65535.



In this detect every alert had 65535 as either a source or destination port, or both. Several ports other than 65535 were used, but none at SMTP 25 as shown in Figure 17. This is a radial nail graph showing flagged Source ports below 3500. The lack of port 25 or 110

suggests no hosts were infected by the Adore worm between the dates of January 4-7th. A sample table of the ports is included in Table 19 with sample hosts.

Table 19

Port	Service	Source IP	Destination IP
0	none	256.256.006.050 256.256.006.051	256.256.152.176
16	?	256.256.006.051	256.256.152.176
255	?	256.256.006.050 256.256.006.051	256.256.006.049 256.256.153.203
256	RAP	256.256.006.048 256.256.006.049	256.256.152.159 256.256.152.172
440	sgcp	256.256.006.049	256.256.152.175

515	printer	256.256.006.050	256.256.153.177
1,000	cadlock2	256.256.006.048 256.256.006.052	256.256.153.199 256.256.153.180
2,030	device2	256.256.006.050	256.256.153.203
2,560	labrat	256.256.006.049	256.256.152.183
2,724	qotps	256.256.153.142	256.256.153.142
3,136	Grub Server Port	256.256.153.142	256.256.153.142
15,441	?	256.256.006.062	256.256.153.207

256.256.6.50 is the #1 source IP for this alert and is associated with 2 others.

- 1 instances of [Port 55850 udp - Possible myserver activity - ref. 010313-1](#)
- 1 instances of [Attempted Sun RPC high port access](#)
- 509 instances of [High port 65535 udp - possible Red Worm - traffic](#)

256.256.6.49 is the #2 source IP for this alert and is associated with 3 others including Back Orifice and myserver traffic.

- 1 instances of [Port 55850 udp - Possible myserver activity - ref. 010313-1](#)
- 1 instances of [Back Orifice](#)
- 3 instances of [Attempted Sun RPC high port access](#)
- 424 instances of [High port 65535 udp - possible Red Worm - traffic](#)

01/07-11:29:48.335125 [**] Back Orifice [**] 256.256.6.49:26465 ->
256.256.153.204:31337

01/08-14:02:43.098704 [**] Port 55850 udp - Possible myserver activity - ref.
010313-1 [**] 256.256.6.49:108 -> 256.256.153.207:55850

Defensive Recommendations

The MY.NET firewall should be configured to block traffic at 31337 and 65535 since they are known Trojan ports, and the likelihood of legitimate traffic using them is remote. The MY.NET mail server should be configured to block the email addresses above. Finally all MY.NET hosts in Table 19 should be inspected and cleaned with the tool written by William Stearns, which can be found here: http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm. 256.256.6.49 and 256.256.6.50 should be removed from the network immediately, and inspected for possible Trojan activity. They may require rebuilding if found to be compromised. Following this network logs should be monitored carefully for continued 65535 port traffic.

Correlation

http://www.giac.org/practical/Christof_Voemel_GCIA.txt
http://www.giac.org/practical/Jeff_Holland_GCIA.doc
http://www.giac.org/practical/James_Hoover_GCIA.doc
http://www.giac.org/practical/David_Leach_GCIA.doc

ICMP Router Selection

Description of Attack

There were 124 source hosts for 1 destination, 224.0.0.2, and 1,285 alerts. The Snort rule that generated this traffic might look like this:

```
alert icmp any any -> any any (msg:"ICMP Router Selection"; itype: 10; icode: 0;)
```

This rule is filtering for normal, local ICMP router auto-discovery traffic, which every host might send out at startup. 224.0.0.2 is a multi-cast address for all routers according to Max Vision, who's posting can be seen in the correlation section, which hosts can query to discover local routers. This behavior is defined in RFC 1256: <http://www.ietf.org/rfc/rfc1256.txt>. This is normal behavior for internal hosts, and could be dropped from the IDS rule set unless there is reason to suspect it is being abused.

It would seem that 256.256.150.1, presumably the local router, has been configured to deny this traffic from 256.256.150.24.

Earliest: **17:05:37.590445** on 01/04/2002

Latest: **22:48:25.132889** on 01/08/2002

1 signature is present for **256.256.150.24** as a source

- 167 instances of [ICMP Router Selection](#)

Earliest: **16:44:40.992373** on 01/04/2002

Latest: **23:32:40.330506** on 01/08/2002

1 signature is present for **256.256.150.24** as a destination

- 329 instances of [ICMP Destination Unreachable \(Communication Administratively Prohibited\)](#)

Defensive Recommendations

Review the local router's ACL, if 256.256.150.24 does not need to be blocked anymore. A modification for this rule will decrease false positives and still watch for attempted information probes from external sources. The new rule should look like this one taken from <http://www.snort.org/snort-db/sid.html?id=443> rules database:

```
alert icmp $EXTERNAL any -> $MY_NET any (msg:"ICMP Router Selection"; itype: 10; icode: 0; reference:arachnids,174; sid:443; classtype:misc-activity; rev:4;)
```

Correlation

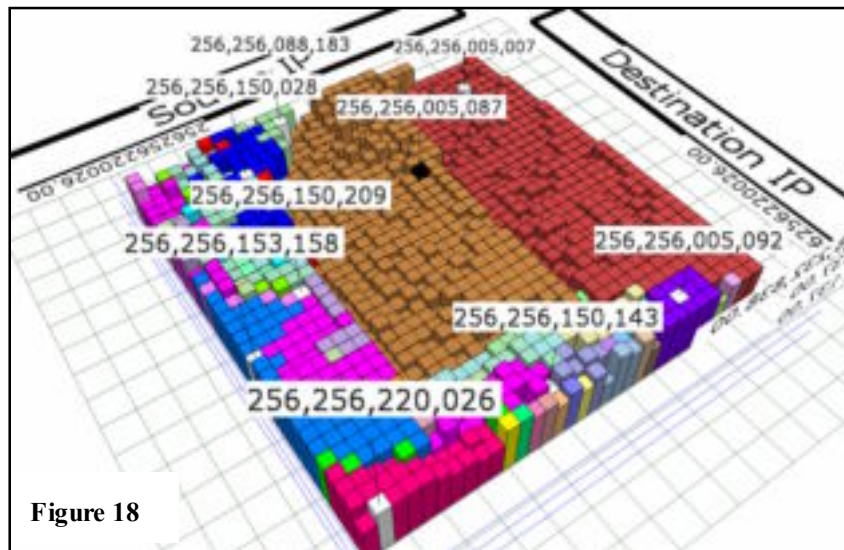
<http://archives.neohapsis.com/archives/snort/2001-03/0168.html>

SMB Name Wildcard

Description of Attack

There were 38 source hosts for 33 destination hosts, generating 1,078 alerts. The SMB Name Wildcard is a normal request for NetBIOS file sharing information from one Windows host to another. The Snort rule might look something like this given the observation that all alerts were concerning internal hosts exclusively on port 137:

```
alert UDP $INTERNAL any -> $INTERNAL 137 (msg: "IDS177/netbios_netbios-name-query"; content: "CKAAAAAAAAAAAAAAAAAAAAAAAAAAAA|00 00|";)
```

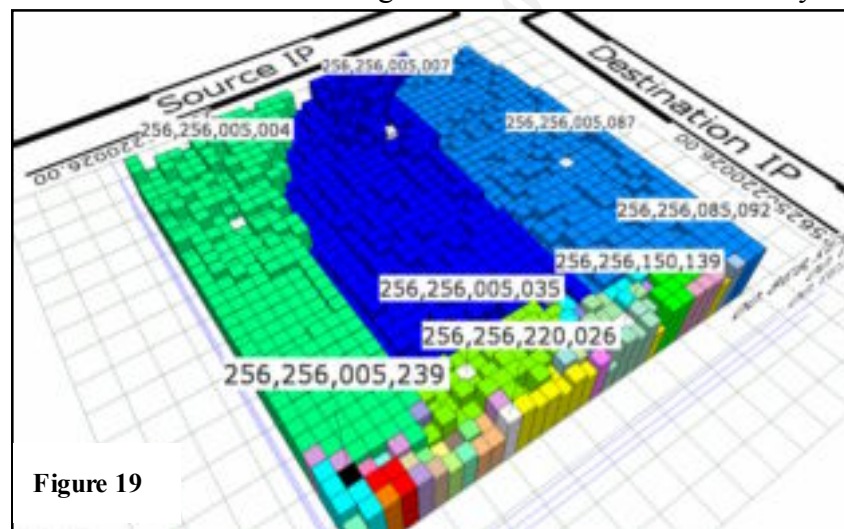


The origin of Figure 18 is in the top corner. Source IP's are grouped by color and the top 5 talkers are flagged along with others of interest. Notice that the addresses come from different subnets of MY.NET.

Figure 19 is the same graph with the Destination hosts grouped by color and the top 6 receiving hosts flagged. Here it is evident that 256.256.5.7 and

256.256.5.87 has been talking to each other almost exclusively. The 256.256.5.4 host appears to

be the recipient of numerous source hosts. Snort-Snarf confirms that 256.256.5.4 received the most varied alerts from 14 different sources. Snort-Snarf also reveals that 24 hosts were listed as both sources and destinations of this signature, there were no external addresses, and all traffic was on port 137. Given all this it would seem that 256.256.5.4 is a Windows file server, and there is no indication of



malicious traffic here.

Yet the individual addresses involved with this detect show some concerning behaviors.

The 256.256.5.7 host, which appears to be a Windows2000 host, is the #1 source of the SMB Name Wildcard signature, and has been pinging 256.256.5.87 at various intervals throughout the week. Windows 2000 ping requests are reported as ICMP Echo Request L3retriever Pings. This may be a misconfiguration, but it is generating unnecessary alerts.

2 different signatures are present for **256.256.5.7** as a source

- 343 instances of [*ICMP Echo Request L3retriever Ping*](#)
- 347 instances of [*SMB Name Wildcard*](#)

The 256.256.150.143 address is the #1 scanning source of the week, and #4 Top Talker overall, has been busy scanning the Internet for Napster, port 6699 as seen in Table 20.

Table 20

Jan 4 17:22:48	256.256.150.143:3805	->	61.207.178.43:6699	SYN	*****S*
Jan 4 17:22:49	256.256.150.143:3808	->	24.250.193.9:6699	SYN	*****S*
Jan 4 17:22:50	256.256.150.143:3809	->	61.210.89.59:6699	SYN	*****S*
Jan 4 17:23:05	256.256.150.143:3811	->	80.135.173.236:6699	SYN	*****S*

3 different signatures are present for **256.256.150.143** as a source

- 1 instances of [*ICMP Echo Request Windows*](#)
- 2 instances of [*SMB Name Wildcard*](#)
- 26 instances of [*INFO MSN IM Chat data*](#)

8 different signatures are present for **256.256.150.143** as a destination

- 1 instances of [*SCAN Synscan Portscan ID 19104*](#)
- 1 instances of [*INFO Napster Client Data*](#)
- 2 instances of [*SMB Name Wildcard*](#)
- 3 instances of [*EXPLOIT x86 setgid 0*](#)
- 7 instances of [*EXPLOIT x86 setuid 0*](#)
- 12 instances of [*Incomplete Packet Fragments Discarded*](#)
- 26 instances of [*INFO MSN IM Chat data*](#)
- 593 instances of [*MISC Large UDP Packet*](#)

There has been contact with several points on the Internet. It appears that there has been inbound file sharing from 202.102.29.141, possibly from Napster clients, and attempts to access Kazaa file sharing services on port 1214 from other IP's as well. Kazaa is a peer-to-peer file sharing utility similar to Napster, both of which can introduce a plethora of security issues into a network. Most disturbing about these port 1214 packets are the signatures they matched in Table 21. Exploit x86 setgid is an attempted system call, but it could also be a false positive from a gif and WinZip file transfers according to H.D. Moore in his posting here:

<http://archives.neohapsis.com/archives/snort/2001-05/0331.html>

Table 21

01/07-17:18:33.363053	[**]	MISC Large UDP Packet	[**]	202.102.29.141:1260	->	256.256.150.143:1568
01/08-21:35:49.318880	[**]	EXPLOIT x86 setgid 0	[**]	128.93.24.104:1214	->	256.256.150.143:3034
01/08-21:48:12.628264	[**]	EXPLOIT x86 setgid 0	[**]	128.93.24.104:1214	->	256.256.150.143:3262
01/08-23:03:17.998441	[**]	EXPLOIT x86 setuid 0	[**]	147.231.56.78:1214	->	256.256.150.143:3826
01/08-23:32:29.256445	[**]	EXPLOIT x86 setuid 0	[**]	24.157.195.138:1250	->	256.256.150.143:1214

The most troublesome address yet seen is 256.256.5.92. The single alert for Subseven qualifies this host for a closer inspection regardless of the high number of alerts signatures associated with it including local: FTP, web, and pc-anywhere services.

6 different signatures are present for 256.256.5.92 as a source

- 1 instances of [SUNRPC highport access!](#)
- 1 instances of [ICMP Echo Request L3retriever Ping](#)
- 1 instances of [IDS50/trojan trojan-active-subseven](#)
- 3 instances of [WEB-IIS Unauthorized IP Access Attempt](#)
- 13 instances of [WEB-MISC 403 Forbidden](#)
- 14 instances of [SMB Name Wildcard](#)

11 different signatures are present for 256.256.5.92 as a destination

- 1 instances of [WEB-CGI redirect access](#)
- 2 instances of [ICMP Echo Request L3retriever Ping](#)
- 2 instances of [MISC traceroute](#)
- 2 instances of [MISC PCAnywhere Startup](#)
- 3 instances of [IDS52/web-iis IIS ISAPI Overflow ida nosize](#)
- 4 instances of [SMB Name Wildcard](#)
- 6 instances of [FTP passwd attempt](#)
- 9 instances of [spp http decode: IIS Unicode attack detected](#)
- 10 instances of [ICMP Echo Request Cisco Type.x](#)
- 21 instances of [WEB-MISC Attempt to execute cmd](#)
- 235 instances of [SNMP public access](#)

Defensive Recommendations

The Snort rule should be tuned to decrease false positives by modifying it to examine only external traffic entering the network as such:

```
alert UDP $EXTERNAL 137 -> $MY_NET 137 (msg: "IDS177/netbios_netbios-  
name-query"; content: "CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA|00 00|";)
```


Alternatively it could be written to focus only on internal hosts of interest per the local security policy. All unnecessary Windows file sharing should be turned off, and the network firewall(s) should be configured to prevent SMB traffic ports 137-139, from entering or leaving the network.

The 256.256.5.7 host should be inspected to determine if file sharing and trust relationships are secured properly.

The primary user responsible for 256.256.150.143 should be counseled regarding the network usage policy if Napster like file sharing services is forbidden. Also the host should be inspected for any signs of a compromise, patched, and secured for any known vulnerabilities since the likelihood for this behavior to continue is high.

The 256.256.5.92 host needs to be removed from the network immediately. There is evidence of targeted Trojan activity originating from outside the MY.NET domain, and attempts to access root privileged services such as pc-anywhere. It may require a clean re-installation to properly secure it from any compromises. Also the offending hosts: 216.150.152.145, 12.91.166.128, and 130.94.19.198 should be blocked at the network firewall from any further contact with MY.NET.

Correlation

http://www.sans.org/newlook/resources/IDFAQ/port_137.htm
<http://cert.uni-stuttgart.de/archive/incidents/2001/05/msg00037.html>
http://www.cert.org/incident_notes/IN-2000-02.html

Top 10 Scans

The Top 5 Source and Destination Hosts are displayed in Tables 22 – 32 for the Top 10 scans. In every case these are attempted reconnaissance and information leaks. The differences between each scan are evident in the Scan rule title. For instance UDP Scan indicates that UDP packets are the vehicles of the scanning packets, and TCP SYN scans are TCP packets with only the SYN flag set. As evidenced by 961,834 scans over 5 days, generating over 60 megabytes of Snort log files, the IDS's logs are too unwieldy to be properly analyzed. The Snort.conf file appears to be set to record single packets from a given host as a scan, the vast majority of them UDP and TCP SYN scans. The SYN scans are predominately directed at port 80 which could be two things: either a large number of port 80 scans occurred, or else the logs include every SYN packet from every handshake between January 4 and January 8th. This is evident in the table below where "Top 10" scan signatures amount to single packet source hosts beginning with the TCP *2UA**** scan.

UDP scan

Description of Attack

Table 22

Sources	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
<u>256.256.60.43</u>	369,031	369,031	161	161

256.256.6.49	49,237	49,237	60	60
256.256.6.50	46,390	46,390	47	47
256.256.6.52	26,618	26,618	65	65
256.256.6.51	25,077	25,077	50	50
Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
256.256.1.3	26,869	26,869	321	321
256.256.1.4	20,265	20,265	232	232
256.256.153.189	14,834	14,834	8	8
256.256.153.142	14,717	14,717	8	8
256.256.153.203	12,865	12,865	8	8

Defensive Recommendations

The port scan preprocessor, which is activated in the Snort.conf, seems to be set to monitor a threshold of 1 port connection over 1 seconds from any IP address, on “all” ports judging from the log’s contents. Legitimate traffic, including single packet dns requests, was logged in this scan. It is highly suggested that this be reset to a higher threshold for the number of ports accessed per detection period, measured in seconds, to log a more accurate scanning record. The network’s DNS servers should also be excluded from this setting. The number of ports checked for scans should be reduced to only those services which are provided on MY.NET, and not already filtered or blocked by border firewalls/routers.

This is the recommended setting:

```
preprocessor portscan: $MY_NET 4 3 portscan.log
preprocessor portscan-ignorehosts: $DNSSERVERS
```

This setting should be adjusted accordingly to meet the needs of MY.NET’s security policies. This single change should drastically reduce the log size from any five-day period, while still logging the actual scan alerts.

TCP *****S* scan

Description of Attack

Table 23

Sources	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
256.256.150.143	12,411	12,411	672	672
256.256.88.162	6,515	6,515	2,815	2,815

256.256.150.165	4,295	4,295	1,031	1,031
256.256.153.148	3,959	3,59	1,238	1,238
256.256.153.46	2,870	2,870	719	719
Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
131.118.254.39	2,143	2143	190	190
131.118.254.38	2,092	2,092	188	188
256.256.11.4	1,385	1,385	104	104
256.256.153.111	1,360	1,362	4	4
256.256.5.83	1,322	1,322	23	23

Defensive Recommendations

It is suggested that the port scan preprocessor be set to a higher threshold for the number of ports accessed per detection period, measured in seconds. These are the default settings:

```
preprocessor portscan: $MY_NET 4 3 portscan.log
preprocessor portscan-ignorehosts: $DNSSERVERS
```

This same recommendation will hold for the remaining scan types.

TCP ***** scan

Description of Attack

Table 24

Sources	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
256.256.186.16	68	68	1	1
144.122.42.38	7	44	1	1
62.131.53.136	5	5	1	1
24.158.117.251	4	42	1	1
195.132.240.41	3	36	1	1
Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
256.256.150.137	68	68	1	1
256.256.88.162	15	122	7	24

256.256.150.220	8	26	8	15
256.256.150.204	6	104	3	19
256.256.151.18	5	9	1	5

TCP ****P*** scan

Description of Attack

Table 25

Sources	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
148.63.238.182	20	20	1	1
148.63.95.200	9	9	1	1
148.63.22.119	7	7	1	1
148.64.1.217	3	4	1	1
148.63.231.81	2	2	1	1
Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
256.256.150.204	50	104	12	19
256.256.88.162	8	122	7	24
256.256.153.148	1	8	1	5
256.256.150.220	1	26	1	15

TCP *2UA**** scan

Description of Attack

Table 26

Sources	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
65.129.59.20	1	2	1	1
65.129.44.238	1	2	1	1
65.129.34.188	1	2	1	1
65.129.33.1	1	2	1	1
65.129.48.82	1	2	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
256.256.150.83	9	312	9	78
256.256.150.204	1	104	1	19

TCP ***A*R*F scan

Description of Attack

Table 27

Sources	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
210.143.130.229	5	5	1	1
24.132.81.46	1	3	1	1
Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
256.256.153.206	5	14	1	5
256.256.88.162	1	122	1	24

TCP *2*A**S* scan

Description of Attack

Table 28

Sources	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
64.70.32.61	6	6	1	1
Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
256.256.153.122	6	9	1	4

TCP *2**PR*F scan

Description of Attack

Table 29

Sources	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
195.132.240.41	2	36	1	1
24.158.117.251	1	42	1	1
144.122.42.38	1	44	1	1

212.120.85.196	1	2	1	1
Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
256.256.88.162	3	122	2	24
256.256.150.143	1	18	1	7
256.256.150.204	1	104	1	19

TCP *2UAP*** scan

Description of Attack

Table 30

Sources	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
24.158.117.251	2	42	1	1
130.104.19.73	1	12	1	1
66.68.58.90	1	4	1	1
Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
256.256.88.162	2	122	2	24
256.256.150.204	2	104	1	19

TCP *2***R** scan

Description of Attack

Table 31

Sources	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
24.158.117.251	1	42	1	1
130.104.19.73	1	12	1	1
195.132.240.41	1	36	1	1
66.121.247.51	1	5	1	1
Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
256.256.88.162	2	122	2	24
256.256.150.204	2	104	2	19

TCP *2***SF scan

Description of Attack

Table 32

Sources	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
66.121.247.51	2	5	1	1
130.104.19.73	1	12	1	1
144.122.42.38	1	44	1	1
Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
256.256.88.162	2	122	2	24
256.256.150.204	2	104	1	19

Out of Specification Packets

There were a total of 64 packets logged as Out of Specification for TCP/IP traffic from 14 unique source hosts communicating with 6 unique destinations, across 16 ports. These hosts appear in Table 33 with the number of packets logged as the frequency. The top 4 sources also appear frequently in the Top 10 Scans. In every case the packets' TCP flags were artificially set to be "Out of Specification" with TCP/IP behavior. There does not appear to be any packet corruption or hardware failures, since all other fields were reasonable and valid. Together the Flags and TCP option combinations suggested packet crafting. Also the most prevalent destination port, shown in Table 33, was 1214 Kazaa. Dshield – The Movie, <http://www.dshield.org/dshieldmovie.html>, has reported Kazaa port 1214 as one of the most scanned ports on the Internet during early 2002. It is possible that these packets are signs of stealthy probes, or slow scans.

Table 33

Source IPs	Freq.	Destination IPs	Freq.	Dest port	Freq.
144.122.42.38	16	256.256.88.162	36	1214	38
195.132.240.41	10	256.256.150.204	13	4216	6
24.158.117.251	9	256.256.150.143	8	6699	5
130.104.19.73	8	256.256.153.206	5	48668	2
4.61.46.216	6	256.256.150.220	1	2106	2
192.116.55.2	5	256.256.153.148	1	9876	1
66.121.247.51	3			4915	1
68.41.218.102	1			4780	1
213.67.0.17	1			4336	1
200.207.18.19	1			4307	1
193.226.113.248	1			34450	1

Source IPs	Freq.		Destination IPs	Freq.		Dest port	Freq.
213.77.129.108	1					3343	1
62.198.104.68	1					1883	1
66.68.58.90	1					1697	1
						1127	1
						1110	1

External sources Requiring Further Investigation

The following external IP addresses require further investigation due to the severity of the alerts they generated. These alerts are symptoms of possible root level compromises, or intensive scans of internal hosts. In all cases the MY.NET hosts affected will need to be investigated for compromises, and secured against the same. Also the MY.NET firewall/router should be configured to drop traffic to or from these sources. Dshield.org provided the WHOIS lookups.

1 different signature is present for **12.91.166.128** as a destination

- 1 instance of [IDS50/trojan_trojan-active-subseven](#)

There is 1 distinct source IP in the alerts of the type.

01/08-21:05:14.184567 [**] [IDS50/trojan_trojan-active-subseven](#) [**] [256.256.5.92:1243](#) -> [12.91.166.128:1341](#)

IP Address: 12.91.166.128

HostName: 128.washington-32rh15rt.dc.dial-access.att.net

DShield Profile: Country:

Contact E-mail:

Total Records against IP:

Number of targets:

Date Range: to

Ports Attacked (up to 10): Port Attacks

Whois: AT&T ITS (NET-ATT)

200 Laurel Avenue South

Middletown, NJ 07748

US

Netname: ATT

Netblock: 12.0.0.0 - 12.255.255.255

Maintainer: ATTW

Coordinator:

Kostick, Deirdre (DK71-ARIN) help@IP.ATT.NET
(888)613-6330

Domain System inverse mapping provided by:

DBRU.BR.NS.ELS-GMS.ATT.NET 199.191.128.106

DMTU.MT.NS.ELS-GMS.ATT.NET 12.127.16.70

CBRU.BR.NS.ELS-GMS.ATT.NET 199.191.128.105

CMTU.MT.NS.ELS-GMS.ATT.NET 12.127.16.69

Record last updated on 06-Nov-2000.
Database last updated on 20-Mar-2002 19:58:52 EDT.

3 different signatures are present for **12.25.239.5** as a source

- 1 instance of [TFTP - Internal UDP connection to external tftp server](#)
- 2 instances of [EXPLOIT NTPDX buffer overflow](#)
- 9 instances of [High port 65535 udp - possible Red Worm - traffic](#)

There is 1 distinct destination IP in the alerts of the type.

01/07-15:18:27.544514	[**]	EXPLOIT NTPDX buffer overflow	[**]	12.25.239.5:123	->	256.256.150.120:123
01/07-15:29:36.160739	[**]	High port 65535 udp - possible Red Worm - traffic	[**]	12.25.239.5:65535	->	256.256.150.120:65535
01/07-16:17:26.909814	[**]	TFTP - Internal UDP connection to external tftp server	[**]	12.25.239.5:69	->	256.256.150.120:9984

IP Address: 12.25.239.5

HostName: 12.25.239.5

DShield Profile: Country: US

Contact E-mail: travisd@inflow.com

Total Records against IP: 24

Number of targets: 7

Date Range: 2002-01-31 to 2002-02-11

Ports Attacked (up to 10): Port Attacks

Whois: AT&T ITS (NET-ATT)

200 Laurel Avenue South

Middletown, NJ 07748

US

Netname: ATT

Netblock: 12.0.0.0 - 12.255.255.255

Maintainer: ATTW

Coordinator:

Kostick, Deirdre (DK71-ARIN) help@IP.ATT.NET

(888)613-6330

Domain System inverse mapping provided by:

DBRU.BR.NS.ELS-GMS.ATT.NET 199.191.128.106

DMTU.MT.NS.ELS-GMS.ATT.NET 12.127.16.70

CBRU.BR.NS.ELS-GMS.ATT.NET 199.191.128.105

CMTU.MT.NS.ELS-GMS.ATT.NET 12.127.16.69

Record last updated on 06-Nov-2000.

Database last updated on 20-Mar-2002 19:58:52 EDT.

Inflow (NETBLK-ATT137321616-232)

710 N. TUCKER BLVD SUITE 610

ST LOUIS, MO 63101

US

Netname: ATT137321616-232
Netblock: 12.25.232.0 - 12.25.239.255
Maintainer: NFLO
Coordinator:
Deatherage, Travis (TD255-ARIN) travisd@inflow.com
303-942-2832
Record last updated on 06-Jun-2001.
Database last updated on 20-Mar-2002 19:58:52 EDT.

3WK Radio (NETBLK-INFLOW-12254-4841)
710 N Tucker Blvd suite 610
St Louis, MO 63101
US
Netname: INFLOW-12254-4841
Netblock: 12.25.239.0 - 12.25.239.15
Coordinator:
Inflow, Joe (JI133-ARIN) hostmaster@inflow.com
(303) 942-2800
Record last updated on 02-Oct-2001.
Database last updated on 20-Mar-2002 19:58:52 EDT.

1 signature is present for **216.150.152.145** as a source

- 2 instances of [MISC PCAnywhere Startup](#)

There is 1 distinct destination IP in the alerts of the type.

01/08-15:22:01.293368 [**] MISC PCAnywhere Startup [**] 216.150.152.145:2108 -> 256.256.5.92:5632
--

IP Address: 216.150.152.145
HostName: wiredforlife5.spyral.net
DShield Profile: Country: US
Contact E-mail: s_oestreicher@xand.COM
Total Records against IP:
Number of targets:
Date Range: to
Ports Attacked (up to 10): Port Attacks
Whois: Xand Corporation (NET-XAND-BLK-1)
11 Skyline Drive
Hawthorne, NY 10532
US
Netname: XAND-BLK-1
Netblock: 216.150.128.0 - 216.150.159.255
Maintainer: XAND
Coordinator:
Xand Corporation (ZX8-ARIN) dnsadmin@xand.com
914-592-8282
Domain System inverse mapping provided by:

AUTH01.DNS.XAND.COM 216.150.131.196
AUTH02.DNS.XAND.COM 216.150.131.197
Record last updated on 31-Jan-2002.
Database last updated on 20-Mar-2002 19:58:52 EDT.

SpyralNet, LLC (NETBLK-SPYRALNET-152)
99 Main St.
Nyack, NY 10960
US
Netname: SPYRALNET-152
Netblock: 216.150.152.0 - 216.150.159.255
Coordinator:
Beckwith, Ted (TB346-ARIN) ted@SPYRAL.NET
914-348-7676
Domain System inverse mapping provided by:
NS1.SPYRAL.NET 12.20.196.11
NS2.SPYRAL.NET 12.20.196.12
Record last updated on 29-Jul-2000.
Database last updated on 20-Mar-2002 19:58:52 EDT.

2 different signatures are present for **203.229.99.115** as a source to 256.256.5.92

- 6 instances of [spp_http_decode: IIS Unicode attack detected](#)
- 11 instances of [WEB-MISC Attempt to execute cmd](#)

There are 2 distinct destination IPs in the alerts of the type.

```
01/07-19:11:49.977412 [**] WEB-MISC Attempt to execute cmd [**] 203.229.99.13:1723 ->  
256.256.5.96:80
```

```
01/07-19:11:51.634892 [**] spp\_http\_decode: IIS Unicode attack detected [**]  
203.229.99.13:1816 -> 256.256.5.96:80
```

IP Address: 203.229.99.115
HostName: 203.229.99.115
DSshield Profile: Country: KR
Contact E-mail: shon@samyang.co.kr
Total Records against IP:
Number of targets:
Date Range: to
Ports Attacked (up to 10): Port Attacks
Whois:
IP Address : 203.229.96.0-203.229.99.255
Connect ISP Name : NOWCOM
Connect Date : 19990210
Registration Date : 19990224
Network Name : SAMYANGDATA
[Organization Information]
Orgnization ID : ORG52645

Name : Samyang Data System Co., Ltd
 State : SEOUL
 Address : 263 Yeonji-Dong Chongno-GU
 Zip Code : 110-725
 [Admin Contact Information]
 Name : Hyunho Son
 Org Name : Samyang Data System Co., Ltd
 State : SEOUL
 Address : 263, Yeonji-Dong, Chongno-GU
 Zip Code : 110-725
 Phone : 02-740-7103
 Fax : 02-740-7098
 E-Mail : shon@samyang.co.kr
 [Technical Contact Information]
 Name : Hyunho Son
 Org Name : Samyang Data System Co., Ltd
 State : SEOUL
 Address : 263, Yeonji-Dong, Chongno-GU
 Zip Code : 110-725
 Phone : 02-740-7103
 Fax : 02-740-7098
 E-Mail : shon@samyang.co.kr

#1 OOS Source using Kazaa dst port to 256.256.88.162

4 different signatures are present for **144.122.42.38** as a source

- 1 instance of [SYN-FIN scan!](#)
- 2 instances of [SCAN FIN](#)
- 2 instances of [SCAN XMAS](#)
- 14 instances of [Null scan!](#)

There is 1 distinct destination IP in the alerts of the type.

01/07-04:58:38.018264	[**]	Null scan!	[**]	144.122.42.38:4719	->	256.256.88.162:1214
01/07-05:22:33.673383	[**]	SCAN XMAS	[**]	144.122.42.38:0	->	256.256.88.162:4915
01/07-16:14:55.701263	[**]	SYN-FIN scan!	[**]	144.122.42.38:2718	->	256.256.88.162:1214
01/07-21:58:18.882579	[**]	SCAN FIN	[**]	144.122.42.38:2106	->	256.256.88.162:1214

IP Address: 144.122.42.38

HostName: 1207.odtukent.metu.edu.tr

DShield Profile: Country: TR

Contact E-mail: hostmaster@METU.EDU.TR

Total Records against IP:

Number of targets:

Date Range: to

Ports Attacked (up to 10): Port Attacks

Whois: Middle East Technical University (NET-METU-NET)

METU Computer Center Inonu Bulvari - ODTU

Ankara, 06531

TR

Netname: METU-NET

Netblock: 144.122.0.0 - 144.122.255.255

Coordinator:

METU Hostmaster (MH2-ORG-ARIN) hostmaster@METU.EDU.TR

+90 312 2103330

Fax- +90 312 2101120

Domain System inverse mapping provided by:

NS1.METU.EDU.TR 144.122.199.90

NS2.METU.EDU.TR 144.122.199.93

NS1-AUTH.SPRINTLINK.NET 206.228.179.10

AUTH60.NS.UU.NET 198.6.1.181

Record last updated on 27-Oct-1998.

Database last updated on 3-Mar-2002 19:56:53 EDT.

24.158.117.251 appears in several Top 10 Scans and OOS

Earliest: **08:05:42** on 1/8/2002

Latest: **09:11:01** on 1/8/2002

33 different signatures are present for **24.158.117.251** as a source

- 1 instances of [TCP ***A**SF scan](#)
- 1 instances of [TCP *2U*P*** scan](#)
- 1 instances of [TCP *2U*PR** scan](#)
- 1 instances of [TCP I***P*** scan](#)
- 1 instances of [TCP **U*PRS* scan](#)
- 1 instances of [TCP I**AP*** scan](#)
- 1 instances of [TCP I**APRS* scan](#)
- 1 instances of [TCP *2U***** scan](#)
- 1 instances of [TCP *2*APR** scan](#)
- 1 instances of [TCP *****S* scan](#)
- 1 instances of [TCP I*****F scan](#)
- 1 instances of [TCP I****RSF scan](#)
- 1 instances of [TCP *2UA**SF scan](#)
- 1 instances of [TCP I**A***F scan](#)
- 1 instances of [TCP I2U*P*** scan](#)
- 1 instances of [TCP I2*APR** scan](#)
- 1 instances of [TCP *2U**RS* scan](#)
- 1 instances of [TCP I*UAP**F scan](#)
- 1 instances of [TCP *2UAP**F scan](#)
- 1 instances of [TCP *2**PR*F scan](#)
- 1 instances of [TCP *2***R** scan](#)
- 1 instances of [TCP *2UA**** scan](#)
- 1 instances of [TCP *2UAPR*F scan](#)
- 1 instances of [TCP *****SF scan](#)

- 1 instances of [TCP *****R*F scan](#)
- 1 instances of [TCP I*U*PR** scan](#)
- 2 instances of [TCP *2UAP*** scan](#)
- 2 instances of [TCP *2*A**** scan](#)
- 2 instances of [TCP I*U***** scan](#)
- 2 instances of [TCP I*UAP*SF scan](#)
- 2 instances of [TCP I2*A***F scan](#)
- 2 instances of [TCP ****P*** scan](#)
- 4 instances of [TCP ***** scan](#)

Jan 8 08:06:44 [24.158.117.251:1323](#) -> [256.256.150.204:1214](#) VECNA *2U*****
RESERVEDBITS

IP Address: 24.158.117.251

HostName: kpt-c-24-158-117-251.chartertn.net

DShield Profile: Country: US

Contact E-mail: ipaddressing@chartercom.com

Total Records against IP:

Number of targets:

Date Range: to

Ports Attacked (up to 10): Port Attacks

Whois: Charter Communications, Inc. (NETBLK-CHARTER-NET-2BLK)

12405 Powerscourt

St. Louis, MO 63131

US

Netname: CHARTER-NET-2BLK

Netblock: 24.158.0.0 - 24.158.255.255

Maintainer: CC04

Coordinator:

Charter Communications (ZC119-ARIN) ipaddressing@chartercom.com

314-965-0555

Domain System inverse mapping provided by:

NS1.CHARTER.COM 24.196.241.11

NS2.CHARTER.COM 24.213.60.79

NS3.CHARTER.COM 24.197.48.54

NS4.CHARTER.COM 24.205.1.12

ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

Record last updated on 05-Feb-2002.

Database last updated on 29-Mar-2002 19:57:39 EDT.

Charter Communications (NETBLK-KNGPT-TN-24-158-112)

105 Jack White Drive

Kingsport, TN 37664

US

Netname: KNGPT-TN-24-158-112

Netblock: 24.158.112.0 - 24.158.127.255

Coordinator:

Charter Communications (ZC119-ARIN) ipaddressing@chartercom.com
314-965-0555

Record last updated on 02-Feb-2002.

Database last updated on 29-Mar-2002 19:57:39 EDT.

Description of the analysis process

This was by far the hardest part of the practical for me mostly because I had no idea where to begin looking at all the data. I've used Snort-Snarf before, so I set it up to run immediately. I tried out several perl scripts from various practicals, and discovered most wouldn't run for me "as is" on my systems, or I just didn't want to feel like a chimpanzee running scripts with no understanding of scripting myself. I learned Linux command line editing real quick with grep, sed, and even vi (shudders). I knew I wanted to visualize the data, and I spent considerable time waiting looking for the holy grail of graphing tools. Lenny Zeltser used The Personal Brain for the link graph and that was interesting, and turned out to be very easy to use for correlating all events. OpenDX is a very promising, powerful data visualization tool, if someone had the time to learn how to format the data, or if they wrote a snort module to output log files in a format for it. Finally my girlfriend discovered www.3DV8.com, and I down loaded a trial version of 3DV8 and I was so happy I bought it. Thank you sweetie! I'm sure my use of it was clumsy and awkward. It's hard to look at those pictures and choose what to point out, because there was so much information available to see by just pointing and clicking. I hope other will try it, and go farther with it than I did.

Systems and Tools Used

Systems	Snort tools	CMD tools	Editing Tools	Visualization
1 Ghz Athlon Thunderbird 1 Gig Ram RedHat 7.1	Snort, Snort-Snarf, Snort-sort	cat, grep, vi, perl	gedit	OpenDX (didn't use in paper)
1.7 Pentium4 512 Mb Ram Win2k	Snort, Snort-Snarf, Snort-sort,	cygwin tools, perl	MS Word 97 Ms Excel WordPad	PersonalBrain 3DV8
1Ghz duel-boot Dell Laptop 512 Mb Ram WinXP, RedHat 7.2	Snort, Snort-Snarf, Snort-sort	cygwin tools, Perl	MS Word 97 Ms Excel WordPad	PersonalBrain 3DV8
		cat, grep, vi, perl	gedit	OpenDX (didn't use in paper)

The Procedure

My goal was to find a way to visualize the analyzed data. To do this I needed to find tools appropriate to the task, and then format the data to use them. I used Jeff Holland's practical to get me started with a cat cmd:

```
cat alert.020104 alert.020105 alert.020106 alert.020107 alert.020108 > alert.MY.NET.txt
```

Next I used Jeff's vi command to change the "MY.NET" to "256.256" so that Snort-Snarf would run correctly, and I saved the output to alert.256.256.

```
:1,$s/MY.NET/256.256/g
```

(Looking back 256 was not the best choice for my analysis method of graphing. A lower number might have been better for graphing purposes.)

First I used "199.256" as Jeff did, believing I had grep'ed each file to make sure that that number didn't turn up. Turns out one scan file used it in the seconds column since the '.' is apparently a wild card. That mistake cost me a week's worth of time trying to get Snort-Snarf to parse out the scan.199.256 file. Then I ran the alert.256.256 file through Snort-Snarf, which completed in under an hour. The cat'd scans file however, took over one week to run on the 1.7 P4 and I eventually stopped it after seven days of no activity. The cure was to grep out the 50 megabytes UDP lines, which made up over 80% of the whole file, and run the TCP scans separately. I already knew the #1 scan was UDP so at that point I didn't care about Snarfing any more.

I down loaded Snort-Sort, which ran on the alert.256.256 file just fine. This sorted out the alerts by type and allowed a quick view of the alerts.

Next I entered the IP's of the Top 20 Alerts and Scans by source and destinations, and a Top 10 Talkers list into The Personal Brain, I down loaded the free 30-day trial period. This quickly showed me linked IP's between the lists. I also loaded some alerts into The Personal Brain with their corresponding source and destination IP's, which began associating all the IP's between the alerts and Top 20's. This helped to see and confirm relationships faster than Snort-Snarf.

Now I wanted to import the data into Excel, and I discovered that I needed to seriously reformat it. Bill Royds used a sed statement, which separates fields by semi colons, and prepares the alert file for the next step. Bill used this:

```
sed -e 's/ *\[*\] */;/g' -e 's/ -> /;/' -e 's/-/ /' -e 's^\([0-9]\[0-9\]*\.[0-9]\[0-9\]*\):\([0-9]\)\1;/2/g' -e 's:\([0-6]\[0-9]\)\.:\1;0/' sansAlert.txt |sort -k 1d >snortalerts.tab
```

But I used this:

```
sed -e 's/ *\[*\] */;/g' -e 's/ -> /;/' -e 's/-/ /' -e 's:\([0-6]\[0-9]\)\.:\1;0/' alert.020104.txt |sort -k 1d >alert.256.royds.tab.txt
```

because I didn't use his entire process. The resulting alert file reads like such:

```
01/04 16:44:38;0.095023;ICMP traceroute;199.256.5.202;199.256.5.1
01/04 16:44:40;0.992373;ICMP Destination Unreachable (Communication
Administratively Prohibited);199.256.150.1;199.256.150.24
01/04 16:44:48;0.090718;ICMP traceroute;199.256.5.202;199.256.5.1
01/04 16:45:41;0.723897;SNMP public access;199.256.88.240;1029;
199.256.150.195;161
```

Then I used a combination of vi, and gedit to further refine the format to:

```
0104164438,0.095023,ICMP traceroute,256.256.5.202,256.256.5.1
0104164440,0.992373,ICMP Destination Unreachable (Communication
Administratively Prohibited),256.256.150.1,256.256.150.24
0104164448,0.090718,ICMP traceroute,256.256.5.202,256.256.5.1
0104164541,0.723897,SNMP public access,256.256.88.240,1029,
256.256.150.195,161
```

Now it was time to split the alert.256.royds.tab.csv file into parts. I simply grepped out the top 10 occurring alerts into .csv files.

Then came the tricky part. In order to import the data into 3DV8, I needed to ‘normalize’ the IP addresses, and still keep the correct octet numbers. This was important because 3DV8 wasn’t really a good tool for handling IP addresses as text strings so I needed to convert them to whole numbers. In a grand attempt to avoid learning perl scripting, I succeeded in delaying the inevitable by discovering that Excel has a feature, which allows you to format cells!

So I brought each file into Excel as a comma separated value file (.csv). First I deleted the second’s column. Next I inserted columns where the comma’s had been and copied semi-colons into them. Then select the source & destination IP columns individually and choose “text to columns” under the Data toolbar. Delimitate them by the periods between octets and you’ll get four new columns for each. Then highlight each octet column and right-click (or left-click for you weird lefties) and choose “format cells”. Then highlight “custom” under category, and type “000.” for type. Finally save the file as a .csv.

Next, grep for any part of the alert string. This removes the extra semi-colons, which Excel thought you wanted copied to all 65535 rows in the columns created above! Then go back into gedit and first remove all remaining periods and commas. Lastly, replace the semi-colons with commas, and add the header to your fields. I wound up with the following:

```
Date\Time,Alert,Source IP,Src Port,Destination IP,Dst Port
105121847,INFO MSN IM Chat data,256256153113,2768,064004012159,1863
105121909,INFO MSN IM Chat data,256256153113,2769,064004012184,1863
105121916,INFO MSN IM Chat data,256256153113,2769,064004012184,1863
```

You can now import this data into 3DV8. A couple things to remember though is this tool does not “prove anything”, nor are the visuals even useful unless the data can show a meaningful relationship. Generally I found it was most helpful in pointing out trends, glaring anomalies in traffic, and when looking at an alert file with less than 128 items such as source port, alert type, port etc. The too few or too many items made neat pictures, but were a waste of time for generating the visuals for analysis. Still the pictures do say a thousand words when you consider the “gee-wiz!” factor of presenting the analysis this way. I discovered yawning friends, and co-workers who were sick of watching me write this practical suddenly take a new interest and say, “Hey, what’s all that?” They still didn’t care about the analysis or maybe understand it, but I got their attention. Also I attempted to correlate and verify everything I “think I saw” with Snort-Snarf or Snort-sort as well as The Brain.

The Scans were simply copied out of Snort-Snarf into tables and examined. There were too many to examine by hand once I realized the port scan preprocessor must be misconfigured. The OOS files were cat'd together and then examined with the scripts from Chris Baker, and Scott Shinberg's practicals.

References

Alexander, Bryce. "Port 137 Scan." Intrusion Detection FAQ. 05/10/2000.

URL: http://www.sans.org/newlook/resources/IDFAQ/port_137.htm (Mar. 2002).

Bell, Mike. "GCIA Practical."

URL: http://www.sans.org/y2k/practical/Mike_Bell_GCIA.doc (Mar. 2002).

Blackhat.com Presentations. "Security in Network Management – Security in Distributed and remote Network Management Protocols."

URL: <http://www.blackhat.com/presentations/bh-usa-97/Jeremy-snmp.ppt> (Mar. 2002).

CERT.org. "CERT[®] Incident Note IN-2000-02 - Exploitation of Unprotected Windows Networking Shares." 04/07/2000.

URL: http://www.cert.org/incident_notes/IN-2000-02.html (Mar. 2002).

Deering, Stephen E. "RFC 1256: ICMP Router Discovery Messages." 09/1991

URL: <http://www.ietf.org/rfc/rfc1256.txt> (Mar. 2002).

Farschchi, Jamil. "GCIA Practical."

URL: http://www.giac.org/practical/Jamil_Farshchi_GCIA.doc (Mar. 2002).

Fearnow, Matt., Stearns, William. "Global Incident Analysis Center, Adore Worm." 04/12/2001.

URL: <http://www.incidents.org/react/adore.php> (Mar. 2002).

Fearnow, Matt., Stearns, William. "Adore Worm Detection and Removal Tool."

URL: http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm (Mar. 2002).

Holland, Jeff. "GCIA Practical."

URL: http://www.giac.org/practical/Jeff_Holland_GCIA.doc (Mar. 2002).

Hoover, James. "GCIA Practical."

URL: http://www.giac.org/practical/James_Hoover_GCIA.doc (Mar. 2002).

Leach, David. "GCIA Practical."

URL: http://www.giac.org/practical/David_Leach_GCIA.doc (Mar. 2002).

Leweling, Martin "Re: [suse-security] dienst blackjack?" 12/02/200.

URL: <http://lists.suse.com/archive/suse-security/2000-Dec/0026.html> (Mar. 2002).

Lukacs, Steve. "GCIA Practical."

URL: http://www.giac.org/practical/Steve_Lukacs_GCIA.doc (Mar. 2002).

Moor, H.D. "Re: [Snort-users] Shellcode x86 setgid 0." 05/13/2001.
URL: <http://archives.neohapsis.com/archives/snort/2001-05/0331.html> (Mar. 2002).

Portsdb.org. "The Internet Ports Database." 01/17/2002.
URL: <http://www.portsdb.org/bin/portsdb.cgi?portnumber=256&protocol=ANY&String> (Mar. 2002).

Roesch, Martin. "HTTP Decode." Snort Users Manual Snort Release: 1.8.4.
URL: http://www.snort.org/docs/writing_rules/chap2.html#tth_sEc2.4.2 (Mar. 2002).

Roesch, Martin. "Portscan Detector." Snort Users Manual Snort Release: 1.8.4.
URL: http://www.snort.org/docs/writing_rules/chap2.html#tth_sEc2.4.2 (Mar. 2002).

Royds, Bill. "GCIA Practical."
URL: http://www.giac.org/practical/Bill_Royds.zip (Mar. 2002).

Rubin, Yotam. "Spoofed SMB name wildcard probes." 05/04/2001.
URL: <http://cert.uni-stuttgart.de/archive/incidents/2001/05/msg00037.html> (Mar. 2002).

SANS.com Resources. "The Twenty Most Critical Internet Security Vulnerabilities (Updated) The Experts' Consensus Version 2.501." 01/30/2002.
URL: <http://www.sans.org/top20.htm> (Mar. 2002).

SCITECH.com. "Machine Vision Systems."
URL: <http://www.scitech.com.au/MVS/mvs.html> (Mar. 2002).

Snort.org "ICMP Router Selection." Signature Database. 03/13/2002.
URL: <http://www.snort.org/snort-db/sid.html?id=443> (Mar. 2002).

Snort.org. "MISC Large ICMP Packet." Signature Database. 3/13/2002.
URL: <http://www.snort.org/snort-db/sid.html?id=499> (Mar. 2002).

Spitzner, Lance. "[Snort-users] Win98 on reboot." 03/11/2001.
URL: <http://archives.neohapsis.com/archives/snort/2001-03/0168.html> (Mar. 2002).

[The MITRE Corporation](#) "CVE-2000-0884." Common Vulnerabilities and Exposures (CVE). 01/22/2001. URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=2000-0884> (Mar. 2002).

Voemel, Christof. "GCIA Practical."
URL: http://www.giac.org/practical/Christof_Voemel_GCIA.txt (Mar. 2002).