GIAC Certified Intrusion Analyst (GCIA) Practical Assignment Version 4.0

Brian M. Estep
GCIA / Orlando FL
November 24, 2004

Table of Contents

List of Figures

Abstract

This document provides an assessment covering seven days worth of network captures that have been moderately obscured by SANS/GIAC. This report is presented to 'The University' as a security assessment. The analysis was performed on seven files obtained from the RAW logs section of the Internet Storm Center maintained by SANS.

The detailed analysis section proceeds as in a manner that addresses the requirements of Part II of the Practical Assignment even though the order is not the same as the ordered list provided. The method used here ties together the network information gathering and correlating phases before introducing and discussing alert signatures.

References to specific file names will appear 30 days out of sync with the data contained within the actual files. This has been reported in almost every similar assessment performed using this log repository.

Throughout the document suggestions and recommendations will be made in an attempt to bolster "The University's" security posture.

Document Conventions

As you read this assessment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

`Command`	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
`Filename`	Filenames, paths, and directory names are represented in this style.
`Computer output`	The results of a command and other computer output are in this style unless the elongated output is presented.
`Elongated Computer Output`	The results of an elongated output or command are presented in this style.
URL	Web URL's are shown in this style. Unless the URL is contained in a footnote, if this happens the font size will decrease by one.
Quotation	A citation or quotation from a book or web site is in this style.
Highlighting	Highlighted text is intended to draw attention to or single out data points.

Part I: Executive Summary

A secure networking environment is best created in an environment where enforceable security policies are communicated, network designs are updated to address security issues, and network or security resources are able to perform thorough triage of anomalous events. Assessments like this one provide the justification for improving the security posture.

At the request of The University, one week's worth of traffic captures was reviewed and analyzed. Later sections of this document will cover the specifics of the analysis including exact file names, traffic statistics, IDS alerts, and network configuration information previewed here:

- 7 daily capture files represented 17.5 megabytes of data over a period of 149 hours
- Each packet averaged just over 1KB in size, for an average of 256 bps (bits per second) during the 7 day period
- 3,049 packets were logged entering the network, while 12,021 packets were logged exiting the network
- HTTP (web) services appear to be the protocol destination represented in 88.9% of outbound traffic packets
- Multiple reconnaissance probes were noted and analyzed
- More than one source of spyware was noted
- 4,873 alerts were logged across 21 unique alert categories
- The alert list included port scans, DNS probes, possible responses by internal hosts to outside attacks, multiple external threat sources, and likely internal exploit attempts
- Over 3,700 alerts were triggered by incoming and outgoing HTTP (web) traffic
- Multiple peer-to-peer file sharing applications are active on the network

Information collection is an essential part of any network operation. From the statistics above, it is clear The University is missing some logging opportunities because almost 19 hours of logs are missing. This void could lead to 6 weeks of missing data in a year's time. This presents a major risk The University's management is encouraged to review regardless of any other finding presented within this document.

Finally, the following items should be considered in conjunction with the detailed findings presented in the remainder of this report:

- Begin a Security Awareness program and educate faculty and students
- Modify existing security policy to allow for more intrusion detection systems and log correlation stations
- Develop a standard list of allowed protocols / applications and consider limiting P2P traffic before piracy or privacy issues arise
- Request the Service Provider perform filtering of known bad traffic

Part II: Detailed Analysis

No information was provided relating to the network layout, IP addressing schema, IDS configuration, firewall placement, or security policies, other than the capture files. It will be possible to reconstruct portions of the network where traffic was collected; sufficient details should be attainable such that a high level network diagram and basic network configuration is discernable.

Network Architecture / Configuration

Tcpdump is a widely used utility to capture network traffic; tcpdump is also able to read a variety of capture files. Here is a listing of the files for this assessment originally downloaded from http://isc.sans.org/logs/RAW/

```
> ls -l 2002.4.1* 2002.4.20 | awk '{print $9,$6,$7,$8," ",$5}'
2002.4.14 Apr 5 2004   561246
2002.4.15 Apr 5 2004   3655456
2002.4.16 Apr 5 2004   3264654
2002.4.17 Apr 5 2004   2808869
2002.4.18 Apr 5 2004   260778
2002.4.19 Apr 5 2004   348129
2002.4.20 Apr 5 2004   2622843
```

Prior to running tcpdump, it is important to verify the file format:

```
> file 2002.4.1* 2002.4.20| cut –d ‘ ‘ –f 1-4,7-
2002.4.14: tcpdump capture file version 2.4 (Ethernet, capture length 1514)
2002.4.15: tcpdump capture file version 2.4 (Ethernet, capture length 1514)
2002.4.16: tcpdump capture file version 2.4 (Ethernet, capture length 1514)
2002.4.17: tcpdump capture file version 2.4 (Ethernet, capture length 1514)
2002.4.18: tcpdump capture file version 2.4 (Ethernet, capture length 1514)
2002.4.19: tcpdump capture file version 2.4 (Ethernet, capture length 1514)
2002.4.20: tcpdump capture file version 2.4 (Ethernet, capture length 1514)
```

Using tcpdump, the following time periods are contained within these files:

```
2002-05-14 18:45:12.004488
2002-05-14 23:59:17.664488
2002-05-15 00:00:49.414488
2002-05-15 23:59:52.364488
2002-05-16 00:05:52.074488
2002-05-16 23:53:40.944488
2002-05-17 00:03:34.584488
2002-05-17 23:28:26.454488
2002-05-18 00:05:52.524488
2002-05-18 23:48:42.054488
2002-05-19 00:26:35.754488
2002-05-19 23:46:04.054488
2002-05-20 00:12:39.154488
2002-05-20 23:59:47.824488
```

Note the dates within the log files are actually different than the files names of each file and the first file does not begin until 18:45. This creates a window of almost 19 hours were no logging was recorded. This is an issue that should be looked into. Logging is as important for real-time analysis of problems as it is for doing log reviews after an incident has occurred.

Tcpdump Parsing

Tcpdump[1] will be used to parse the Ethernet capture files provided for this assessment. First the merged capture file will be used to get an idea of the number of MAC Addresses active in the network. Once this is done, the devices discovered will be mapped out according to the reported traffic sources and destinations.

Tcpdump is called with the r, t, q, e, n options mapping to:

`-r`	tells tcpdump to read from the specified capture file, in this case the merged file containing the 7 individual files is being read
`-t`	removes timestamp information from the output
`-q`	produces a subset of information otherwise output from *tcpdump*
`-e`	prints the link-level headers in each packet, in this case Ethernet MAC headers are printed
`-n`	prevents hostname lookups for every IP Address
`-c #`	instructs tcpdump to only parse the specified number (#) of packets

Let's combine these options with common UNIX parsing commands to create a MAC Inventory (please see the appendices for information on these commands):

Network Reconnaissance Using Tcpdump

Looking at the MAC Addresses from the output we see only two unique MAC Addresses visible from the IDS:

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen| cut -d ' ' -f 1-
3|sort| uniq -c| tr , ' '
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
  12021 00:00:0c:04:b2:33 > 00:03:e3:d9:26:c0
   3049 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33
```

There are 12,021 instances of packets moving from `00:00:0c:04:b2:33` to `00:03:e3:d9:26:c0`, while there are 3,049 instances of packets transiting the other direction. This additional information is helpful in understanding the behavior of the network under surveillance. It is possible to cross reference the above MAC Addresses with the IEEE Organizationally Unique Identifiers (OUI) [2] to further define the network topology:

```
00-00-0C   (hex)            CISCO SYSTEMS, INC.
00000C     (base 16)        CISCO SYSTEMS, INC.
                            170 WEST TASMAN DRIVE
                            SAN JOSE CA 95134-1706

00-03-E3   (hex)            Cisco Systems, Inc.
0003E3     (base 16)        Cisco Systems, Inc.
                            170 West Tasman Dr.
                            San Jose CA 95134
```

[1] Tcpdump MAN page. URL: http://www.tcpdump.org/tcpdump_man.html (October 1, 2004)
[2] IEEE OUI Search Website. URL: http://standards.ieee.org/regauth/oui/index.shtml (October 17, 2004)

```
UNITED STATES
```

From this information the network topology must consist of at least two Cisco devices and the IDS must sit in between to be able to provide these captures.

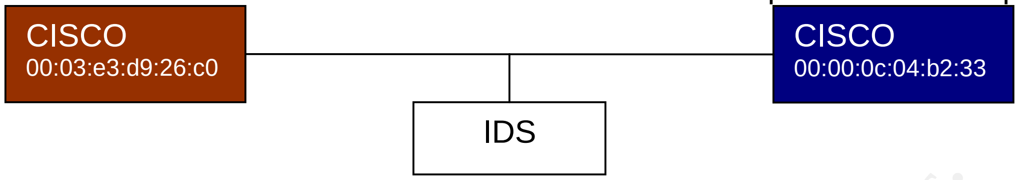


Figure 1: Early Network Topology using MAC Addresses

Additional Analysis of Sensor Data

A frame of reference exists now for further analysis; however, it is not possible to determine what network segments are located behind each Cisco device. The next step is to understand the IP Addressing schema in the above diagram in order to label the Cisco devices appropriately. These traces are color-coded and coincide with the diagram above for easy reference.

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen ether src 0:0:c:4:b2:33
| cut -d ' ' -f 8| cut -d . -f 1-4|sort| uniq -c
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
     28 78.37.212.165
  11993 78.37.212.28
```

The `ether src` option has been included to filter on packets originating from only the specified MAC Address, in this case `0:0:c:4:b2:33` short-handed version of the longer `00:00:0c:04:b2:33`. The `ether dst` option will also be used, which will filter on the destination MAC Address using the same convention. When using the `–q` option in `tcpdump`, field 8 represents the source IP Address and port, while field 10 represents the destination IP Address and port.

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen ether src
0:0:c:4:b2:33| cut -d ' ' -f 10| cut -d . -f 1-4|sort| uniq -c
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
      1 12.109.100.230
      1 12.219.10.157
      1 12.224.253.86
      1 12.225.70.243
       …
      2 80.56.32.250
      1 80.59.217.24
      3 80.60.103.52
```

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen ether src
0:3:e3:d9:26:c0| cut -d ' ' -f 8| cut -d . -f 1-4|sort| uniq -c
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
      1 12.153.224.21
      1 12.217.213.171
      5 12.218.181.36
       …
      2 80.4.52.195
      1 80.4.93.24
      1 80.59.217.24
```

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen ether src
0:3:e3:d9:26:c0| cut -d ' ' -f 10| cut -d . -f 1-4|sort|uniq -c
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
      1 78.37.0.1
      1 78.37.0.113
       …
      1 78.37.253.31
      1 78.37.2.57
      1 78.37.26.121
       …
      1 78.37.99.20
      1 78.37.99.248
```

Data Flow

Using the data from the last section, a clearer picture of the network design can be achieved; however, even with this dataset more questions remain.

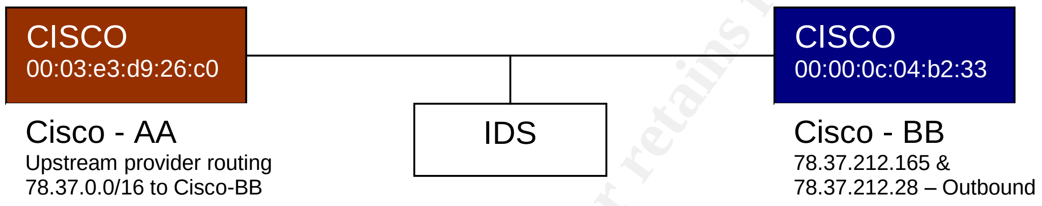


Figure 2: Network Topology with IP Addressing

The captures show a myriad of IP Addresses residing behind the Cisco – AA device routing an entire Class B network address to the Cisco – BB device, while only two IP Addresses source traffic through the Cisco – BB device. This suggests Cisco – AA is an edge network device and most likely the Border Router. Given the Ethernet headers in the capture files, the IDS is most likely located between the two Cisco devices via a hub or spanning port. Further investigation of the IP traffic being sourced from Cisco – BB is needed. Looking at destination ports originating at the Cisco –AA device will reveal any DNS, HTTP, FTP, or other public services provided by the 78.37.0.0/16 network.

Filtering the source port number on packets sourced from the Cisco – BB device, reveals 3,036 ports used for traffic originating from the 78.37.0.0/16 network. Further investigation of the actual port numbers used shows:

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen ether src
0:0:c:4:b2:33| cut -d ' ' -f 8| cut -d . -f 5| sort -n|uniq -c
     28 http
      4 61002
      5 61003
       …
      4 63181
      3 63182
       …
      4 65093
      3 65094
```

This suggests one of the internal IP Addresses is responding to web requests, while the other IP Address is performing NAT. Given the Cisco – BB device is a Cisco product, it is possible this device is a PIX Firewall or Cisco router performing NAT Overload since only one IP Address with high ephemeral ports. Looking at the destination ports passing through the Cisco – AA device we see:

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen ether src
0:3:e3:d9:26:c0| cut -d ' ' -f 10| cut -d . -f 5| sort -n|uniq -c
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
      4
    277 domain:
    349 ftp:
    528 http:
      4 netbios-ns:
    264 printer:
     59 smtp:
    661 socks:
     48 squid:
      1 356:
      …
```

From this port list, it appears the access-lists on the Cisco-AA device could be tightened up a bit. This listing shows 4 inbound packets that appear to have an invalid TCP header, 4 inbound packets to Windows NetBIOS Name Service, and a curious packet on port 356 typically used to access Amiga PC's. Interestingly, the largest recommended packet when accessing the Amiga Explorer application is 512 bytes, this lone packet is over 576 bytes[3]. There is also traffic to known gnutella ports suggesting loose Peer2Peer controls. Comparing the incoming ports associated with http, dns, and ftp to the same outbound port traffic illustrates additional areas where access-lists could be tightened up. Servers providing services should be prevented from initiating outbound service requests except on approved ports. It should be noted the NAT function could be provided by the Cisco – BB device as a NAT Overload statement [4]or additional network devices could reside behind the Cisco – BB element.

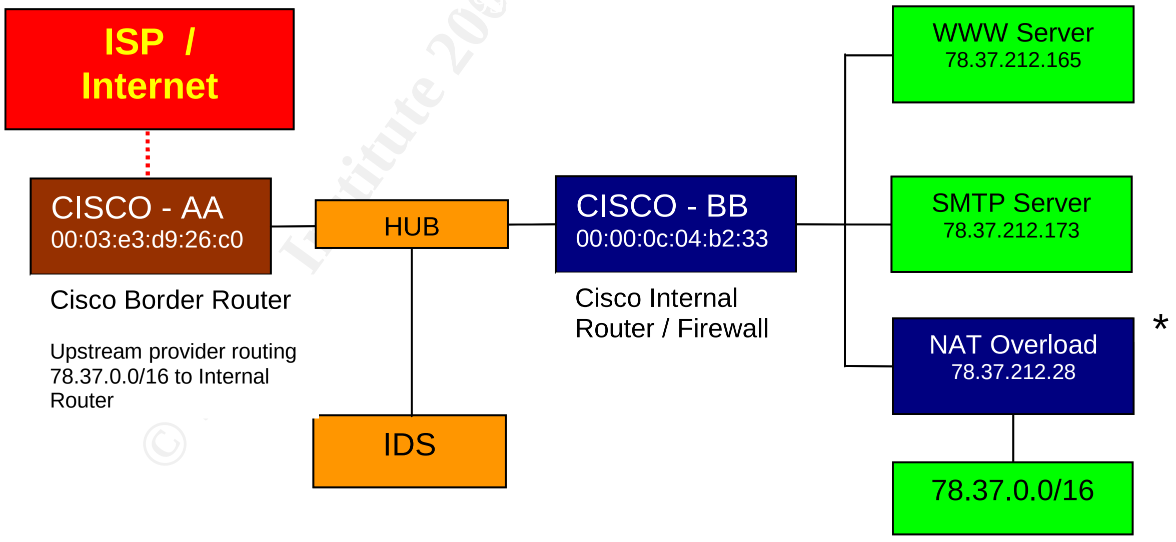


Figure 3: Derived Network Topology

[3] AmigaForever Website. URL: http://www.amigaforever.com/kb/4-109.html (October 11, 2004)
[4] Cisco NAT Setup URL: http://www.cisco.com/warp/public/556/12.html#topic3 (October 5, 2004)

Top Talkers

It is important to understand what service types are being consumed by a given network. This aids in troubleshooting and it also helps diagnose problems that arise due to bandwidth congestion or network latency. When reviewing the destination port matrix from the Cisco – BB device, the initial response prompted additional sorting of the dataset:

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen ether src
0:0:c:4:b2:33| cut -d ' ' -f 10| cut -d . -f 5| sort -n|uniq -c| sort -
rn | tr : ` ` |head –n 5
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
  10654 http
    551 6347
    329 1863
    133 6348
     38 6349
```

This is a listing of the *Top 5 Outbound Destination Ports* sourced from The University's network. 10,654 http packets represent more than 88% of the total 12,021 packets originating from the internal network. The second most prevalent application on The University's network appears to be gnutella using the various 6300 ports. Conspicuously missing was any icmp, pop3, smtp, ftp, or dns traffic. There were traces of MSN Messenger traffic; TCP Port 1863.

After reviewing the destination ports, the previous information on ports destined for the internal network was sorted to provide the following *Top 5 Inbound Destination Ports:*

```
    661 socks
    528 http
    349 ftp
    277 domain
    264 printer
```

The `socks` traffic is likely to be probing for open proxy servers[5] and the initial parsing of destination hosts seems to support that:

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen port 1080 | cut -d ' '
-f 10 | cut -d . -f 1-4 | sort | uniq -c| sort -rn | less
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
     19 78.37.153.174
     18 78.37.180.227
     18 78.37.130.165
      9 78.37.239.138
      9 78.37.176.146
      6 78.37.19.143
      …
```

A snapshot of the source traffic seems to lend even more support to this theory. Keep in mind unless a proxy server is being used as a reverse proxy where outside connections are directed to a single port, traffic should never be destined from the external network to the internal proxy server's proxy port. Allowing such traffic to enter the network is only asking for trouble:

[5] Kurt Seifried Website. URL: http://www.seifried.org/security/ports/1000/1080.html (October 1, 2004)

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen port 1080 | cut -d ' '
-f 8 | cut -d . -f 1-4 | sort | uniq -c| sort -rn
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
    541 216.232.36.98
     24 208.177.5.232
     23 195.119.1.180
     18 64.228.107.49
     18 193.231.96.42
      …
```

Analyzing destination hosts targeted from host 216.232.36.98 confirms the probing theory, at least with this particular host:

```
      1 78.37.0.19
      1 78.37.100.17
       …
      1 78.37.253.18
      1 78.37.253.31
      1 78.37.26.27
       …
      1 78.37.99.20
      2 78.37.162.12
```

Reviewing the source ports from the other source addresses suggests more addresses are scanning TCP port 1080:

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen port 1080 and not host
216.232.36.98|cut -d ' ' -f 8 | cut -d . -f 5 | sort -n| uniq -c
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
      3 1092
      3 1243
       …
      1 39651
      1 39652
      1 39653
       …
      1 57170
      3 59702
```

The http, ftp, and dns inbound traffic is not as concerning here as the 264 packets destined for the printer service (port 515). There are few reasons to allow, trust external connections to internal print servers. Even if this could only result in a printer shooting paper all over the place at 2am in the morning, this is not recommended practice. This traffic is likely to be scan-related; however, without any information on the current security controls, allowed print server listings, and partial logging of outbound traffic from the internal network, this particular stat warrants a closer look.

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen port 515| cut -d ' ' -f
8,10| less
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
255.255.255.255.31337 78.37.83.81.printer:
255.255.255.255.31337 78.37.177.36.printer:
255.255.255.255.31337 78.37.185.215.printer:
255.255.255.255.31337 78.37.179.19.printer:
…
```

Needless to say, this output is extremely troubling. The Border Router should not allow traffic sourced from 255.255.255.255 to enter the network. A malicious packet taking advantage of a broadcast address like this is not expecting a response to this packet. The source port suggests malicious intent, as 31337 is hacker-speak for elite and is typically used by nefarious applications. This host appears to be scanning, but the target selection doesn't fit any pattern at first glance. This traffic should be further reviewed by snort for alerts.

This analysis aimed at identifying network behavior and determining the network design has already identified two interesting external hosts. Let's look at the five most active external hosts with traffic destined for the internal network:

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen ether src
0:3:e3:d9:26:c0|cut -d ' ' -f 8| cut -d . -f 1-4| sort| uniq -c| sort -
rn| head –n 5
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
    541 216.232.36.98
    264 255.255.255.255
    134 192.12.12.14
    115 62.42.179.131
     96 164.164.60.11
```

216.232.36.98 was identified as the TCP Port 1080 scanner from the Top 5 Inbound Destination Ports list. Although this host is identified as a top talker based on packet counts, *this host is also considered highly suspicious because of the nature of the packets sent by this host and the inability to determine the success rate of its scans*. It is not known whether any of these external probes were successful in reaching internal hosts. All these probes are output of some script or program as evidenced by the following IP Header information:

```
      2  length 62: IP tos 0x0, ttl 112, id 1168
    539  length 62: IP tos 0x0, ttl 113, id 1168
```

The two different packets with a ttl of 112 may be the result of a routing update along the path taken by these packets. The IP Address is registered to TELUS Communications, an ADSL provider located in British Columbia:

```
OrgName:    TELUS Communications Inc.
OrgID:      TACE
Address:    #2600 4720 Kingsway Avenue
City:       Burnaby
StateProv:  BC
PostalCode: V5N-4N2
Country:    CA
ReferralServer: rwhois://rwhois.telus.net:4321
NetRange:   216.232.0.0 - 216.232.255.255
CIDR:       216.232.0.0/16
NetName:    TELAC-BLK10
NetHandle:  NET-216-232-0-0-1
```

Coming in at #2, 255.255.255.255 was also identified as a contributor to the Top 5 Inbound Destination Ports. Although the actual source of these packets is not known, *the data contained in this packet make this host the most suspicious external host*. These mysterious packets are directed at a port used for remote printing; all the packets contain RSTs and share the following characteristics:

```
    264 R 0:3(3) ack 0 win 0
    236 length 60: IP (tos 0x0, ttl  13, id 0, offset 0, flags [none], proto 6,
length: 43
    28  length 60: IP (tos 0x0, ttl  14, id 0, offset 0, flags [none], proto 6,
length: 43
```

Here again the only difference in 28 of the packets is an extra hop that may have resulted from a routing update. The ttl for these packets is incredibly low; however, all of these packets were crafted. The source address is spoofed:

```
OrgName:    Internet Assigned Numbers Authority
OrgID:      IANA
Address:    4676 Admiralty Way, Suite 330
City:       Marina del Rey
StateProv:  CA
PostalCode: 90292-6695
Country:    US
NetRange:   240.0.0.0 - 255.255.255.255
CIDR:       240.0.0.0/4
NetName:    RESERVED-240
NetHandle:  NET-240-0-0-0-0
```

In third place, 192.12.12.14 is better known as The Santa Fe Institute, www.santafe.edu. The traffic patterns from this host are all http related to hosts residing behind the NATed 78.37.212.28 address. The exact nature of the sessions is not known, but there are no apparent signs of malicious activity contained within the 134 packets logged.

```
OrgName:    The Santa Fe Institute
OrgID:      SFI
Address:    1120 Canyon Road
City:       Santa Fe
StateProv:  NM
PostalCode: 87501
Country:    US
NetRange:   192.12.12.0 - 192.12.12.255
CIDR:       192.12.12.0/24
NetName:    SANTAFE
NetHandle:  NET-192-12-12-0-1
Parent:     NET-192-0-0-0-0
```

Number 4 on the list: 62.42.179.131 attempted anonymous ftp login 115 times from 2002-05-17 17:35:02.594488 until 2002-05-17 18:05:54.524488 to 78.37.212.165. It is highly unlikely the source IP Address is spoofed because a login is being attempted. No ftp data transfer attempts were logged. The IP Address had no hits at dshield.org[6]; however, the 62.0.0.0/8 net block is allocated to Cableuropa – ONO serving Spain:

```
inetnum:      62.42.128.0 - 62.42.191.255
netname:      ONO-DIAL-1
descr:        Cableuropa - ONO
descr:        ONO net in whole Spain
country:      ES
admin-c:      OIM1-RIPE
tech-c:       OIM1-RIPE
remarks:      mail spam reports: abuse@ono.com
remarks:      security incidents: security@ono.com
notify:       ripe-tech@ono.es
```

[6] Dshield Reports Website. URL: http://www.dshield.org/reports.php (June 10, 2004)

Last on the Top 5 list, 164.164.60.11 appears to be following in the footsteps of #4 except that the anonymous ftp logins are taking place over 8-minute periods spanning 3 days:

```
2002-05-15 20:22:47.524488 through 2002-05-15 20:30:19.184488
2002-05-16 21:45:59.364488 through 2002-05-16 21:53:20.074488
2002-05-20 09:09:33.364488 through 2002-05-20 09:17:36.484488
2002-05-20 19:41:26.704488 through 2002-05-20 19:49:29.424488
```

It is highly unlikely the source IP Address is spoofed because a login is being attempted. There are no logged responses to these packets from the targeted host (78.37.212.165) and no additional network traffic from this host. 164.164.60.11 appears registered to an Indonesian company:

```
inetnum:      164.164.0.0 - 164.164.255.255
netname:      SOFTNET
country:      IN
admin-c:      BN47-AP
tech-c:       BN47-AP
remarks:      inetnum:     164.164.0.0 - 164.164.255.255
remarks:      netname:     SOFTNET
remarks:      org-id:      STPB
remarks:      status:      assignment
remarks:      rev-srv:     STPB.SOFT.NET
remarks:      tech-c:      BVN-ARIN
remarks:      reg-date:    1993-03-19
notify:       tech@stpb.soft.net
mnt-by:       MNT-SOFTNET-IN
source:       APNIC
person:       Bandaru Naidu
address:      Software Technology Park Block III,KSSIDC Complex KEONICS
Elctronics City
              Hosur Road
              Bangalore, 158
```

Top 3 Suspicious External Hosts

255.255.255.255 was selected as the top suspicious host as well as one of the Top 5 External Hosts and has been covered in the preceding section.

216.232.36.98 was also covered earlier and was noted as a suspicious host. This threat posed by this host is more the lack of forensic data to prove whether or not this host is accessing proxy servers inside the 78.37.0.0/16 network. If such access were possible, this could lead to a complete compromise of internal trust as well as provide a stepping stone for external attacks that would now be sourced from the 78.37.0.0/16 network.

The final suspicious host was noted much earlier in this report when over 10,000 outbound TCP port 80 packets were logged. 64.154.80.51 received in excess of 6,380 connections from the internal network. 64.154.80.51 resolves as:

```
ehg.hitbox.com = [ 64.154.80.250 ]
WebSideStory  Inc.
10182 Telesis Ct.
San Diego  CA 92121
```

```
Email: nic@websidestory.com
Registrar Name....: REGISTER.COM  INC.
Registrar Whois...: whois.register.com
Registrar Homepage: www.register.com
Domain Name: hitbox.com
Created on..............: Fri  May 02  1997
Expires on..............: Sat  May 03  2008
Record last updated on..: Fri  Nov 19  2004
```

Hitbox started as a web site tracking tool and as more and more spyware and adware appeared, ehg.hitbox.com has since been dubbed spyware, by virtue of its cookie tracking mechanism[7]. In this case there appear to be more tracking reports uploaded to this server than actual http traffic. This presents a clear danger to the internal network because spyware is not typically detected by anti-virus software and some spyware can actually contain malicious code.

Top Internal Hosts
As presented in an earlier section of this report only two internal hosts are logged transmitting data out of the internal network via Cisco – BB. These are:

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen ether src 0:0:c:4:b2:33
|cut -d ' ' -f 8| cut -d . -f 1-4| sort| uniq -c | sort -rn
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
  11993 78.37.212.28
     28 78.37.212.165
```

78.37.212.28 is either an interface on the Cisco – BB device with NAT Overload configured, which uses a single IP Address on a physical interface as the source address for NATing. The other option is an additional firewalling or routing device located behind the Cisco – BB device is performing the NATing.

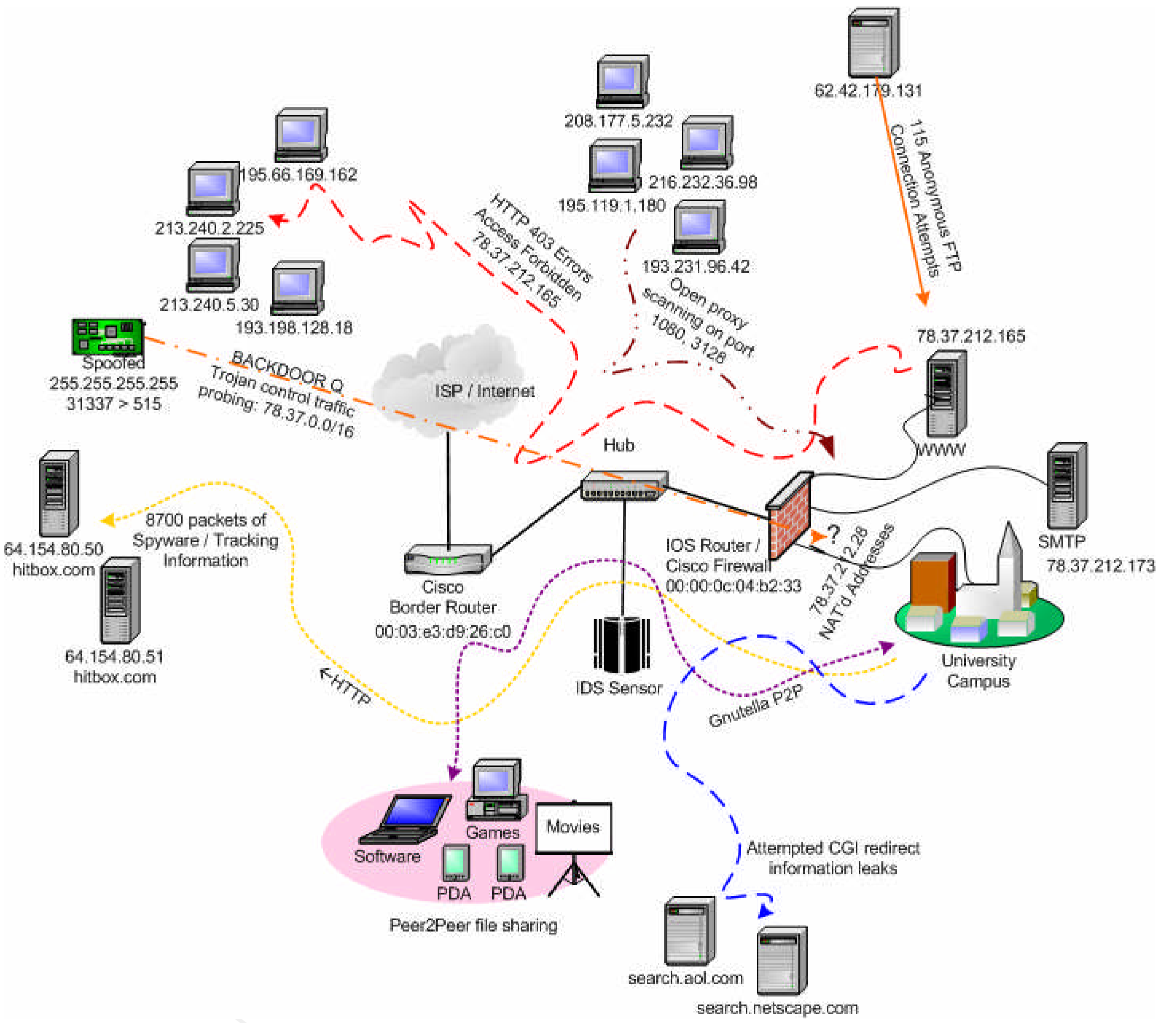
78.37.212.165 provides http service and is logged sending error messages to external hosts, which reveals important data about the server. Here we see the relevant portion of the tcpdump file with a full hex and ascii printout formatted to fit within the margins of this report for easier viewing:

```
        0x0000:  4500 024a 369b 4000 3f06 b9a9 4e25 d4a5  E..J6.@.?...N%..
        0x0010:  c342 a9a2 0050 1024 f7a9 679a f293 5800  .B...P.$..g...X.
        0x0020:  8018 7c70 1595 0000 0101 080a 0014 d7ff  ..|p............
        0x0030:  0971 4d32 4854 5450 2f31 2e31 2034 3033  .qM2HTTP/1.1.403
        0x0040:  2046 6f72 6269 6464 656e 0d0a 4461 7465  .Forbidden..Date
        0x0050:  3a20 5468 752c 2031 3620 4d61 7920 3230  :.Thu,.16.May.20
        0x0060:  3032 2030 373a 3438 3a34 3620 474d 540d  02.07:48:46.GMT.
        0x0070:  0a53 6572 7665 723a 2041 7061 6368 652f  .Server:.Apache/
        0x0080:  312e 332e 3132 2028 556e 6978 2920 2028  1.3.12.(Unix)..(
        0x0090:  5265 6420 4861 742f 4c69 6e75 7829 206d  Red.Hat/Linux).m
        0x00a0:  6f64 5f6a 6b20 6d6f 645f 7373 6c2f 322e  od_jk.mod_ssl/2.
        0x00b0:  362e 3620 4f70 656e 5353 4c2f 302e 392e  6.6.OpenSSL/0.9.
        0x00c0:  3561 2050 4850 2f34 2e30 2e31 706c 3220  5a.PHP/4.0.1pl2.
        0x00d0:  6d6f 645f 7065 726c 2f31 2e32 3420 4672  mod_perl/1.24.Fr
        0x00e0:  6f6e 7450 6167 652f 342e 302e 342e 330d  ontPage/4.0.4.3.
      …
```

This host is running RedHat Linux, Apache 1.3.12 with various modules including: mod_jk, mod_ssl, Open_SSL, PHP, mod_perl, and FrontPage. Apache

[7] CA Spyware Encyclopedia Website. URL: http://www3.ca.com/securityadvisor/pest/pest.aspx?id=453076483 (September 25, 2004)

1.3.12 and OpenSSL 0.9.5a were available with RedHat Linux version 6.2 distributions. It is worth noting OpenSSL 0.9.5a has a private key sequencing vulnerability that allows attackers to determine the private key used. This should be patched quickly if this server is still running this version of OpenSSL. RedHat advisory is: RHSA-2003:101-15[8]. The PHP version on the server is vulnerable to unauthorized remote file uploads as described in PHP BugID #16128[9].



62.42.179.131
115 Anonymous FTP Connection Attempts
208.177.5.232
216.232.36.98
195.119.1.180
193.231.96.42
Open proxy scanning on port 1080, 3128
195.66.169.162
213.240.2.225
213.240.5.30
193.198.128.18
HTTP 403 Errors Access Forbidden 78.37.212.165
Spoofed 255.255.255.255 31337 > 515
BACKDOOR Q Trojan control traffic probing: 78.37.0.0/16
ISP / Internet
78.37.212.165
WWW
Hub
64.154.80.50 hitbox.com
8700 packets of Spyware / Tracking Information
64.154.80.51 hitbox.com
Cisco Border Router 00:03:e3:d9:26:c0
IOS Router / Cisco Firewall 00:00:0c:04:b2:33
?
78.37.212.28 NAT'd Addresses
SMTP 78.37.212.173
University Campus
IDS Sensor
←HTTP
Gnutella P2P
Software
Games
Movies
PDA
PDA
Peer2Peer file sharing
Attempted CGI redirect information leaks
search.aol.com
search.netscape.com

Figure 4: Link Graph

Network Detects

Every piece of data to this point has been analyzed using tcpdump and various UNIX commands in an attempt to illustrate the relationships between various hosts and ports. Next the capture files will be analyzed by *snort,* only the most severe alerts triggered will be reviewed after an overview of all alerts triggered. It

[8] RedHat Errate Website. URL: http://rhn.redhat.com/errata/RHSA-2003-101.html (August 14, 2004)
[9] PHP Website. URL: http://bugs.php.net/bug.php?id=16128 (August 14, 2004)

should be noted for this section that the only modifications made to the default snort configuration file were to ensure every available rule was enabled.

Snort Alert Overview

Count	Snort Alert Triggered
3	1:1390:5 SHELLCODE x86 inc ebx NOOP
6	1:1394:5 SHELLCODE x86 NOOP
496	1:1616:6 DNS named version attempt
528	1:184:7 BACKDOOR Q access
1	1:1882:10 ATTACK-RESPONSES id check returned userid
1	1:498:6 ATTACK-RESPONSES id check returned root
2	1:504:7 MISC source port 53 to <1024
8	1:523:5 BAD-TRAFFIC ip reserved bit set
4	1:621:7 SCAN FIN
9	1:648:7 SHELLCODE x86 NOOP
1	1:650:8 SHELLCODE x86 setuid 0
3	1:653:9 SHELLCODE x86 0x90 unicode NOOP
34	116:46:1 (snort_decoder) WARNING: TCP Data Offset is less than 5!
86	119:12:1 (http_inspect) APACHE WHITESPACE (TAB)
148	119:13:1 (http_inspect) NON-RFC HTTP DELIMITER
52	119:15:1 (http_inspect) OVERSIZE REQUEST-URI DIRECTORY
2	119:16:1 (http_inspect) OVERSIZE CHUNK ENCODING
4	119:18:1 (http_inspect) WEBROOT DIRECTORY TRAVERSAL
76	119:2:1 (http_inspect) DOUBLE DECODING ATTACK
3268	119:4:1 (http_inspect) BARE BYTE UNICODE ENCODING
138	119:7:1 (http_inspect) IIS UNICODE CODEPOINT ENCODING

These alerts are sorted with category 1 alerts first. Each alert is categorized with #:###:# format, where the lower the initial number, the higher the severity of the alert and the higher the trailing number the more revising the rule has undergone.

Detect #1: BACKDOOR Q access

Description of detect

This detect indicates a possible trojan, Q, is being activated. The traces originate from 255.255.255.255, which has already been identified in this report as being suspicious. Now snort is indicating the reasons for additional concern relating to these packets. One of the 528 instances of the alert looks like this:

```
[**] [1:184:7] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
05/14/02-12:58:16.134488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
len:0x3C
255.255.255.255:31337 -> 78.37.83.81:515 TCP TTL:14 TOS:0x0 ID:0
IpLen:20 DgmLen:43
***A*R** Seq: 0x0 Ack: 0x0 Win: 0x0 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
```

The rule that triggered this alert is:

```
alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR Q
access"; flow:stateless; dsize:>1; flags:A+; reference:arachnids,203;
classtype:misc-activity; sid:184; rev:7;)
```

Viewing the actual capture file confirms the alert is valid:

```
2002-05-14 18:58:16.134488 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype
IPv4 (0x0800), length 60: IP (tos 0x0, ttl  14, id 0, offset 0, flags [none],
proto 6, length: 43, bad cksum 529f (->b58)!) 255.255.255.255.31337 >
78.37.83.81.printer: R [bad tcp cksum 6c7 (->bf7f)!] 0:3(3) ack 0 win 0 [RST
cko]
        0x0000:  4500 002b 0000 0000 0e06 529f ffff ffff  E..+......R.....
        0x0010:  4e25 5351 7a69 0203 0000 0000 0000 0000  N%SQzi..........
        0x0020:  5014 0000 06c7 0000 636b 6f00 0000       P.......cko...
```

All 528 instances of these packets contain ttl's less than 15, destination ports of 515, and 'cko' in the payload. The source address is a restricted address that should not appear. The ip ids are 0 and every packet is 60 bytes in length.

Reason this detect was selected
This backdoor references a trojan that uses encryption to obscure its data streams, making further investigation of these packets extremely difficult. These packets sourced from 255.255.255.255 are also interesting because the RST flag is set. This behavior seems a bit odd and warrants investigation of any host found to respond to these packets.

Detect was generated by
This detect was generated using Fedora Core 2 running Snort 2.2.0 with the latest rules dated August 11, 2004. Tcpdump version 3.8 is supported by libpcap version 0.8.3.

The default rule base triggered on this event:

```
alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR Q
access"; flow:stateless; dsize:>1; flags:A+; reference:arachnids,203;
classtype:misc-activity; sid:184; rev:7;)
```

$HOME_NET is set to any. Snort alert is triggered by source traffic of 255.255.255.0/24 with a data payload greater than 1 byte, and having the ACK flag set. If a detect is made, a snort alert msg of "BACKDOOR Q access" is logged with a classification of misc-activity and snort id#184 is referenced. Arachnids #203 is also referenced.

Probability the source address was spoofed
The snort rule triggered ignores TCP state when matching for this detect, which is consistent with Whitehats.com. This is important because it appears the rule itself is expecting spoofed traffic. Broadcasting is not a normal characteristic of TCP communication streams. The source port is 31337, which corresponds to common ports utilized by hackers (31337 stands for elite in hacker speak).

Attack Mechanism
These crafted packets having the RST flag set is most curious from an attack perspective, because the attacker must have some mechanism outside a direct response to these packets to confirm, interact, or activate the backdoor program. If the payload contained within these packets is nothing more than a wake-up call, it would seem the attacker could have devised a more covert means of

communicating; not only is the source address reserved, not only does TCP not make use of broadcasts in this manner, but the source port of 31337 screams malicious intent. Additionally, the destination port (line printer) is a trigger for review because of the improbability that external traffic would be allowed to access internal print servers.

The Q backdoor receives its commands via raw packets and executes them independent of an active network session. The backdoor is designed to redirect input and run remote commands as root. The program takes advantage of encryption in the payload, which would make decoding remote instructions next to impossible.[10]

Correlations

Message thread at http://lists.jammed.com/incidents/2001/05/0038.html discusses some additional information relating to these scans suggesting that the scans are somehow related to IRC because of the following traces:

```
output of  %snoop -i 255255255255.log
1   0.00000    BROADCAST -> ***.***.33.183 PRINTER C port=31337cko
2 2903.93550    BROADCAST -> ***.***.32.36 PRINTER C port=31337cko
3 13652.70806    BROADCAST -> ***.***.25.143 PRINTER C port=31337 cko
4 4689.02603    BROADCAST -> ***.***.141.208 PRINTER C port=31337 cko
```

Mike Wyman was dealing with almost identical traffic in this post at dshield: http://www.dshield.org/pipermail/intrusions/2003-January/006748.php. Danny Walker provided similar analysis on this traffic in his post to dshield: http://www.dshield.org/pipermail/intrusions/2002-November/005972.php

Evidence of Active Targeting

There are 528 detects over 7 days all sourced from the same IP Address and port hitting apparently random hosts across the entire 78.37.0.0/16 network. This is most likely a scan looking for active or remaining backdoor servers because the payload is the same across all these packets.

Severity

severity = (criticality + lethality) (system countermeasures + network countermeasures)

Criticality=4. The hosts being targeted are most likely servers, although none of the packets targeted the known servers existing on the 78.37.212.0/24 subnet.

Lethality=5. This detect involves a backdoor application that receives remote commands and executes them as root. The backdoor is capable of using encryption to mask the commands. Given that this backdoor, if activated, runs commands as root – this is extremely lethal.

[10] Les Gordon, Intrusion Detection – Director's Cut. URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc (October 20, 2004)

System Countermeasures=4. Although the systems targeted did not appear to respond, without full network and system logging there is no way to verify.

Network Countermeasures=2. Two problems with this traffic entering the network: First, no traffic should be allowed to enter being sourced from 31337. Second, the Border Router should only allow 78.37.0.0/16 traffic and 224.0.0.x traffic if multicast is being used. Rating is a 2 because the IDS logged some portion of the traffic.

severity =(4 + 5) – (4 + 2) = 3. This corresponds to a medium level threat

Detect #2: DNS named version attempt

Description of detect

This detect corresponds to an attempt to determine the bind version running on a given dns server. If successful, this could provide information that would be used with a working exploit to take over a dns server. The actual detect looks like this:

```
[**] [1:1616:6] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
05/14/02-02:11:30.474488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
len:0x48
203.122.47.137:13891 -> 78.37.243.52:53 UDP TTL:36 TOS:0x0 ID:7045
IpLen:20 DgmLen:58
Len: 30
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=10028][Xref =>
http://www.whitehats.com/info/IDS278]
```

This alert was generated by the following rule:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named version
attempt"; content:"|07|version"; offset:12; nocase; content:"|04|bind";
offset:12; nocase; reference:arachnids,278; reference:nessus,10028;
classtype:attempted-recon; sid:1616; rev:6;)
```

Viewing the actual network trace confirms the attempt:

```
    2002-05-15 02:11:30.474488 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype
    IPv4 (0x0800), length 72: IP (tos 0x0, ttl  36, id 7045, offset 0, flags
    [none], proto 17, length: 58, bad cksum 8419 (->3ed1)!) 203.122.47.137.13891
    > 78.37.243.52.domain:  4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)
            0x0000:  4500 003a 1b85 0000 2411 8419 cb7a 2f89  E..:....$....z/.
            0x0010:  4e25 f334 3643 0035 0026 9dbb 1234 0080  N%.46C.5.&...4..
            0x0020:  0001 0000 0000 0000 0776 6572 7369 6f6e  .........version
            0x0030:  0462 696e 6400 0010 0003                 .bind.....
    2002-05-15 03:01:19.014488 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype
    IPv4 (0x0800), length 72: IP (tos 0x0, ttl  46, id 62718, offset 0, flags
    [none], proto 17, length: 58, bad cksum 6a2d (->21e8)!) 203.122.47.137.17164
    > 78.37.44.164.domain:  4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)
            0x0000:  4500 003a f4fe 0000 2e11 6a2d cb7a 2f89  E..:......j-.z/.
            0x0010:  4e25 2ca4 430c 0035 0026 5a80 1234 0080  N%,.C..5.&Z..4..
            0x0020:  0001 0000 0000 0000 0776 6572 7369 6f6e  .........version
            0x0030:  0462 696e 6400 0010 0003                 .bind.....
```

There are 47 instances of bind version queries from this host and 248 queries destined for the 78.37.0.0/16 network. Unless the DNS server is configured not to provide the information, bind servers will respond with the version information.

Reason this detect was selected
This detect was selected because events like this happen 24x7 and go largely unnoticed. The data is archived and made ready for the next bind vulnerability to be announced because few administrators configure bind to ignore these queries. Once the vulnerability is announced and exploit code is available, hackers take these lists of vulnerable servers and begin executing the exploit. This probe single-handedly confirms the ability to reach a given DNS server while the response provides the version information for that same server.

Detect was generated by
This detect was generated using Fedora Core 2 running Snort 2.2.0 with the latest rules dated August 11, 2004. Tcpdump version 3.8 is supported by libpcap version 0.8.3.

The default rule base triggered on this event:
```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named version
attempt"; content:"|07|version"; offset:12; nocase; content:"|04|bind";
offset:12; nocase; reference:arachnids,278; reference:nessus,10028;
classtype:attempted-recon; sid:1616; rev:6;)
```

This rule looks at any traffic (because $EXTERNAL_NET and $HOME_NET are set to 'any') destined for port 53 that contains 'version.bind' starting 12 bytes into the payload. For any triggered detect, the alert message "DNS named version attempt" is passed and classified as an 'attempted recon' sighting snort id #1616, arachnids #278, and nessus id 10028.

Probability the source address was spoofed
The source is not likely spoofed as this would defeat the purpose of the bind server reconnaissance, even though UDP packets do not require the three-way handshake that TCP requires.

Attack mechanism
This reconnaissance gathering is simple enough and can be executed using something as simple as nslookup, which is installed on Windows NT, 2K, XP, and most Linux/UNIX OS flavors by default. The queries can be accomplished by setting the record type to txt, the class to chaos, setting a dns server to query, and entering the 'version.bind' command in nslookup.

The traffic captures indicate queries are spread across a range of IP Addresses and only one query is observed entering the 78.37.0.0/16 network every 45minutes to an hour. This would be simple to accomplish without using a script although harvesting, managing, and maintaining this data would be more difficult.

Correlations
The references provided with the snort rule provide valuable information in understanding what is being attempted with this probe as well as countermeasures to prevent a dns server from responding to queries. Those are

located at: nessus: http://cgi.nessus.org/plugins/dump.php3?id=10028 and Whitehats: http://www.whitehats.com/info/IDS278. Additionally, Mike Ellis and Ian Marks reviewed similar reconnaissance and posted analysis at http://www.dshield.org/pipermail/intrusions/2003-April/007411.php and http://lists.sans.org/pipermail/intrusions/2004-July/008200.html respectively.

Evidence of Active Targeting
The following packets suggest additional reconnaissance is in progress while the 78.37.0.0/16 network is being probed:

```
20:11:30.474488 ttl  36, id 7045, 203.122.47.137.13891 > 78.37.243.52.domain
21:01:19.014488 ttl  46, id 62718, 203.122.47.137.17164 > 78.37.44.164.domain
23:58:05.634488 ttl  46, id 32539, 203.122.47.137.29831 > 78.37.47.59.domain
00:12:12.224488 ttl  46, id 48861, 203.122.47.137.21098 > 78.37.137.205.domain
00:20:54.154488 ttl  46, id 58699, 203.122.47.137.29260 > 78.37.208.195.domain
00:24:30.254488 ttl  46, id 63212, 203.122.47.137.10652 > 78.37.250.182.domain
00:43:43.234488 ttl  41, id 18563, 203.122.47.137.28702 > 78.37.197.86.domain
00:44:51.074488 ttl  41, id 19716, 203.122.47.137.29770 > 78.37.131.6.domain
01:17:20.224488 ttl  41, id 57080, 203.122.47.137.16309 > 78.37.165.66.domain
02:39:07.464488 ttl  41, id 16971, 203.122.47.137.27623 > 78.37.168.132.domain
03:04:33.094488 ttl  41, id 46520, 203.122.47.137.29596 > 78.37.232.234.domain
04:01:54.214488 ttl  39, id 43439, 203.122.47.137.17662 > 78.37.223.172.domain
05:08:19.894488 ttl  41, id 50500, 203.122.47.137.14461 > 78.37.108.53.domain
05:51:50.744488 ttl  41, id 29426, 203.122.47.137.11617 > 78.37.21.69.domain
06:41:12.354488 ttl  41, id 14677, 203.122.47.137.14285 > 78.37.131.188.domain
```

The ip id in these packets changes quickly, it is likely a script is being run to collect this information. The ttls change around during this trace, while the source ports appear to randomly cycle through.

Severity
severity = (criticality + lethality) (system countermeasures + network countermeasures)

Criticality=3. DNS servers are a critical piece of any network infrastructure. This probe will not result in a breach; however, unless steps have been taken any responding DNS server will obligingly release its version information.

Lethality=1. These detects reviewed here represent only reconnaissance. Querying for the bind version is not likely to crash a DNS server.

System Countermeasures=3. A 3 is warranted here because once again, no information on the security posture and server configuration is available. There is no data to determine whether the DNS server is running a flawed version of bind.

Network Countermeasures=1. Short of blocking inbound DNS queries from external hosts, there is no network countermeasure available because these packets are legitimate traffic.

severity = (3 + 1) – (3 + 1) = 0. This threat is not impacting until an exploit is released.

Detect #3: SHELLCODE x86 NOOP

Description of detect

This detect means that snort detected NOP instructions for the Intel x86 architecture and an alert was fired. In this case several alerts were fired, but the one of interest is the following alert:

```
[**] [1:648:7] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
05/20/02-18:23:43.044488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
len:0x5DE
204.146.167.81:58325 -> 78.37.212.28:61662 TCP TTL:43 TOS:0x10 ID:15486
IpLen:20 DgmLen:1488
***A**** Seq: 0xF98F189F Ack: 0x52339A5F Win: 0xFFFF TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS181]
```

This alert was triggered from the following rule:

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 NOOP"; content:"|90 90 90 90 90 90 90 90 90 90 90
90 90 90|"; depth:128; reference:arachnids,181; classtype:shellcode-
detect; sid:648; rev:7;)
```

Viewing the actual capture file confirms the alert is valid:

```
   2002-05-20 18:23:43.044488 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype
   IPv4 (0x0800), length 1502: IP 204.146.167.81.58325 > 78.37.212.28.61662: .
   4186904735:4186906183(1448) ack 1379113567 win 65535
        0x0000:  4510 05d0 3c7e 0000 2b06 fcbc cc92 a751  E...<~..+......Q
        0x0010:  4e25 d41c e3d5 f0de f98f 189f 5233 9a5f  N%..........R3._
        0x0020:  5010 ffff 9126 0000 5500 83c4 0485 dbc6  P....&..U.......
        0x0030:  443a 0400 7411 53e8 ecea ffff 83c4 0483  D:..t.S.........
        0x0040:  c8ff 5f5e 5d5b c333 c05f 5e5d 5bc3 5f5e  .._^][.3._^][._^
        0x0050:  5dc7 0520 7441 0009 0000 00c7 0524 7441  ]...tA.......$tA
        0x0060:  0000 0000 0083 c8ff 5bc3 9090 9090 9090  ........[.......
        0x0070:  9090 9090 9090 9090 568b 7424 088b 460c  ........V.t$..F.
        0x0080:  a883 7425 a808 7421 8b46 0850 e837 9fff  ..t%..t!.F.P.7..
          …
        0x0480:  4973 5072 6f63 6573 736f 7246 6561 7475  IsProcessorFeatu
        0x0490:  7265 5072 6573 656e 7400 0000 4b45 524e  rePresent...KERN
        0x04a0:  454c 3332 0000 0000 0000 0000 0000 0000  EL32............
        0x04b0:  652b 3030 3000 0000 7275 6e74 696d 6520  e+000...runtime.
        0x04c0:  6572 726f 7220 0000 0d0a 0000 544c 4f53  error.......TLOS
        0x04d0:  5320 6572 726f 720d 0a00 0000 5349 4e47  S.error.....SING
        0x04e0:  2065 7272 6f72 0d0a 0000 0000 444f 4d41  .error......DOMA
        0x04f0:  494e 2065 7272 6f72 0d0a 0000 5236 3032  IN.error....R602
        0x0500:  380d 0a2d 2075 6e61 626c 6520 746f 2069  8..-.unable.to.i
        0x0510:  6e69 7469 616c 697a 6520 6865 6170 0d0a  nitialize.heap..
        0x0520:  0000 0000 5236 3032 370d 0a2d 206e 6f74  ....R6027..-.not
        0x0530:  2065 6e6f 7567 6820 7370 6163 6520 666f  .enough.space.fo
        0x0540:  7220 6c6f 7769 6f20 696e 6974 6961 6c69  r.lowio.initiali
        0x0550:  7a61 7469 6f6e 0d0a 0000 0000 5236 3032  zation......R602
        0x0560:  360d 0a2d 206e 6f74 2065 6e6f 7567 6820  6..-.not.enough.
        0x0570:  7370 6163 6520 666f 7220 7374 6469 6f20  space.for.stdio.
        0x0580:  696e 6974 6961 6c69 7a61 7469 6f6e 0d0a  initialization..
        0x0590:  0000 0000 5236 3032 350d 0a2d 2070 7572  ....R6025..-.pur
        0x05a0:  6520 7669 7274 7561 6c20 6675 6e63 7469  e.virtual.functi
        0x05b0:  6f6e 2063 616c 6c0d 0a00 0000 5236 3032  on.call.....R602
        0x05c0:  340d 0a2d 206e 6f74 2065 6e6f 7567 6820  4..-.not.enough.
```

The remaining packets carry similar payloads. The repeat occurrence of 0x90's provides the NOOP instructions needed to execute this code, which turns out to be a variant of the W32.Klex worm, most likely the E@mm variant as indicated by the error messages presented and confirmed in a post to incidents.org. This particular variant contained a mass mailing engine that created system log entries matching those contained within the packet. Symantec's Security Response website indicates this worm/virus was first observed January 17, 2002 which makes this a likely threat given the time frames covered in these extracts.

The worm takes advantage of a vulnerability in Internet Explorer where incorrect MIME Headers can cause email attachments to be executed (MS01-020).

Reason this detect was selected
This detect was selected initially because of the confirmed existence of SHELLCODE. Given the loose access-lists applied, apparent lack of P2P controls, a virus or worm could be devastating. This particular threat can disable the antivirus software leaving a computer vulnerable to any number of outside threats that could potentially result in a security breach. Security awareness is a major concern and confirmed traffic to the internal network containing worm or virus code is going to find a victim sooner or later. Unfortunately, it only takes one user getting infected with a worm like this before an internal network is turned into a SPAM factory.

Detect was generated by
This detect was generated using Fedora Core 2 running Snort 2.2.0 with the latest rules dated August 11, 2004. Tcpdump version 3.8 is supported by libpcap version 0.8.3.

The default rule base triggered on this event:
```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 NOOP"; content:"|90 90 90 90 90 90 90 90 90
90 90 90 90 90|"; depth:128; reference:arachnids,181;
classtype:shellcode-detect; sid:648; rev:7;)
```

The $EXTERNAL_NET and $HOME_NET variables are set to any, which matches on any IP address. The $SHELLCODE_PORTS variable looks at every port except port 80. In this rule, snort is asked to fire an alert when an ip packet going to any address from any address, except those addresses where tcp port 80 is involved, contains 14 occurrences of '90' within the first 128 bytes. If a packet meets this criteria, an alert is generated for a "SHELLCODE x86 NOOP" that references arachnids:181 and snort id # 648. This is revision number 7 and this alert is classified as 'shellcode-detect', which is defined in the default classification file as a Priority 1 alert and corresponds to a description of "Executable code was detected" in the classification file.

Probability the source address was spoofed

The probability is low that this source address was spoofed given the manner in which most worms spread, ie probing, uploading of malicious code, and executing malicious code. The source email addresses are randomly generated.

Attack mechanism
The vulnerability exploited by this particular worm is such that the only user interaction required is previewing or opening an infected email in MS Outlook or Outlook Express. This ability to almost guarantee an infection on vulnerable computers through social engineering is a major attraction for worm/virus writers. This variant of the W32.Klez worm introduces a mass mailing spam agent. Unfortunately the worm also delivers another virus known to cause infrequent system crashes: W32.ElKern.3587. The mass mailing portion of this worm sends emails with spoofed sender email addresses. Returned email may be the first indication of an infection.

The worm's payload is triggered on the 6th of each odd-numbered month. This is when antivirus is disabled, massive spamming occurs, and various files are filled with zeros .If the tag-along virus is present, system crashes will begin on the 13th of March and September.

Correlations
John Sage appears to have been one of the early recipients of this virus. John corresponded with Patrick Nolan who posted the details to incidents.org mailing list April 9, 2002, just over a month before this traffic was collected. The post is located here: http://cert.uni-stuttgart.de/archive/intrusions/2002/04/msg00124.html

Symantec's Security Response:
http://securityresponse.symantec.com/avcenter/venc/data/w32.klez.e@mm.html. This also references Microsoft Security Bulletin MS01-020 and CVE-2001-0154.

Evidence of Active Targeting
This worm is mailed to a large number of email addresses obtained from an infected user's email program. As long as the worm is active, it will continue to target email addresses. It is possible for a computer to receive two copies of the worm as a result of this mass mailing.

Severity
severity = (criticality + lethality) (system countermeasures + network countermeasures)

Criticality=3. Even though this targets end-user systems, the worm is capable of overwriting files including network attached files and shutting down anti-virus protection. In a university setting a worm capable of spreading by file shares can be devastating to the network-attached file servers. The original distribution rating for this malware was a high because of its potential to spread easily.

Lethality=3. If the attack were successful, wide-spread system file corruption would be likely. Additionally, because this worm can disable antivirus protection any infected PC is capable of becoming infected with additional worms. The spam capabilities of this worm are also a concern if a large number of computers on a university network began spamming. The original rating for this malware was a category 3: moderate threat with High Damage.

System Countermeasures=2. There is no information to suggest virus updates are not applied regularly; however even the best antivirus software takes time to update when signatures must be replicated across a large network. Additionally, it seems likely from the other activities on the network that end-users may not fully understand the risks associated with executing applications and email attachments.

Network Countermeasures=2. Although the network appears to have some level of filtering and NATing in place, worms that spread through email are primarily defended against by antivirus software and educating users to not open attachments they are not expecting.

Severity = (3 + 3) – (2+2) = 2. This virus represents a minor threat if controls are in place to ensure antivirus software is updated; however, a crisis could only be one missed antivirus update away.

Defensive Recommendations

- Review the SANS Top 20 and block as many inbound and outbound ports as possible. Remember it isn't just important to filter inbound, filtering outbound data ensures less information leakage.
- Filter on known malicious traffic: block ports known to be exploited or used by malware where not needed. An example would be the port 31337 traffic
- Create a central repository of alarm, alert, firewall, network element, and IDS sensor data. There is now way to step back into time and determine the extent of an incident if logs are not maintained and available.
- Disable the P2P traffic, MSN Messenger traffic, and remove the Spyware (hitbox and gator).
- Consider having the Internet Service Provider provide some layer of traffic filtering in order to lessen the load on internal systems. There were several retransmitted packets contained in the logs.
- Develop a formal security awareness program and educate faculty and students alike. University networks are home to some very intelligent people, there is no reason to not hold them to a higher Information Security Standard.
- Periodically test your security policies in approved vulnerability scans and network traffic audits.

Part III: Analysis Process

The analysis workstation consisted of a Fedora Core 2 Linux OS with kernel version 2.6 with the following applications:

- Tcpdump version 3.8 using libpcap version 0.8.3
- Snort version 2.2.0 compiled with the default settings. The only changes made were to enable the snort.conf file to uncomment each rule
- SnortSnarf-021111.1 was used to back up the manual text manipulations performed to generate alert summaries and alert extracts
- Sguil-0.5.2 was compiled for testing, but log parsing proved difficult when performing the xscript requests
- p0f version 2.0.5 was used to attempt finger printing of the three selected detect sources
- ethereal (gnome) and tethereal version 0.10.7 were used to aid in statistical verification of capture files
- tcpdump2dot version 0.9.2 was used to create various link graphs while performing data analysis; Appendix B covers my patch for tcpdump2dot and additional jpegs from the supplied capture files
- tcpflow 0.21 was used early on while attempting to gather any available binary data
- mergecap was used to consolidate the 7 capture files into a single capture file, Appendix A provides a brief how to

A Windows XP laptop was used to create this report using the following applications:

- MS Excel XP for plotting statistical data including mapping ip id, source port, and ttl for specific tcpdump filters and in an attempt to correlate the number of hosts residing behind 78.37.212.28
- MS Word XP utilized for report preparation

Appendix A

mergecap

The mergecap command can be used to create a single capture file for manipulation, if desired.

> mergecap –w mergecapfile-4.14.2002-4.20.2002 ./2002.4.1* ./2002.4.20

The resulting file should be the same size as the sum of the original files

Misc UNIX text commands

This section presents some basic UNIX commands used in this report. For the following example:

```
> tcpdump -r mergecap-4.14.2002-4.20.2002 -tqen| cut -d ' ' -f 1-
3|sort| uniq| tr , ' '
reading from file mergecap-4.14.2002-4.20.2002, link-type EN10MB (Ethernet)
00:00:0c:04:b2:33 > 00:03:e3:d9:26:c0
00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33
```

The *tcpdump* options have been combined and 4 UNIX text manipulation commands have been used to filter the information output to the screen as follows:

`cut`[11] `–d ‘ ‘ –f 1-3` defines columns using the delimiting character specified by the `–d` switch and extracts only the fields specified by the `–f` switch. In this case only the first three columns or fields are extracted. These fields correspond to the source and destination MAC Addresses as indicated by the arrow.

`sort`[12] performs, as the name would imply, a sorting or collecting of data. Here the sort is used to gather the full data set from the cut operation before being passed onto the next command.

`uniq`[13] removes duplicate lines of text from a sorted file. There are several options available under this command. In the next section, the `–c` switch will be used to count the number of occurrences of duplication for each unique element.

`tr`[14] translates or deletes specified characters. This is used here to remove the trailing comma on the destination MAC address in the *tcpdump* output.

[11] cut manual page. URL: http://www.rt.com/man/cut.1.html (September 28, 2004)
[12] sort manual page. URL: http://www.rt.com/man/sort1.html (September 28, 2004)
[13] uniq manual page. URL: http://www.rt.com/man/uniq.1.html (September 28, 2004)
[14] tr manual page. URL: http://www.rt.com/man/tr.1.html (September 28, 2004)

Appendix B

tcpdump2dot[15] is a perl script written to convert tcpdump data into “nodes” and “hosts” associated with those nodes that can be fed into a parser for rendering by GraphViz[16]. GraphViz is a tool originally developed by AT&T Research Laboratories.

Tcpdump has some interesting uses. The URL provided in the footnotes provides a method to obtain near real-time topology maps using tcpdump collection data where tcpdump2dot basically updates the node/host matrix for Graphiz; allowing a refresh of the network topology representation at periodic intervals.

The regex expressions used to evaluate the MAC addresses and IP Addresses needed a few tweaks to work with the current version of tcpdump (v3.8 is current):

```
 # fix, could do some sanity checking...
        $lines++;
-       if ($line =~ /\s+([0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-
fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+)\s+([0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-
fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+)/){

+       if ($line =~ /\s+([0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-
fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+).+?([0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-
fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+)/){
```

This tweak allows perl’s regex parser to match one or more times on a non-greedy character as would be the case when the ‘>’ was picked up between the two MAC addresses. This same fix is needed for the IP Address section in order to let tcpdump2dot map out the MAC addresses and IP Addresses:

```
      if ($line =~
/\s+(\d+\.\d+\.\d+\.\d+)(.\S+)*\s*(>)\s*(\d+\.\d+\.\d+\.\d+)(\.\S+)*[*0
-9]/){
```

The syntax to create a jpeg from a specific tcpdump file named 2002.4.20 where ‘port 25’ is used to filter the capture file:

```
> tcpdump –r 2002.4.14 –qenn port 25 | ./tcpdump2dot.pl | dot –
Tjpeg –Goverlap=false –o mynew.jpeg
```

This will create the file mynew.jpeg in the current directory that should look similar to the figure below:

[15] tcpdump2dot Website. URL: http://www.grotto-group.com/~gulfie/projects/monitoring/tcpdump2dot/
[16] GraphViz Website. URL: http://www.grpahviz.org (November 10, 2004)

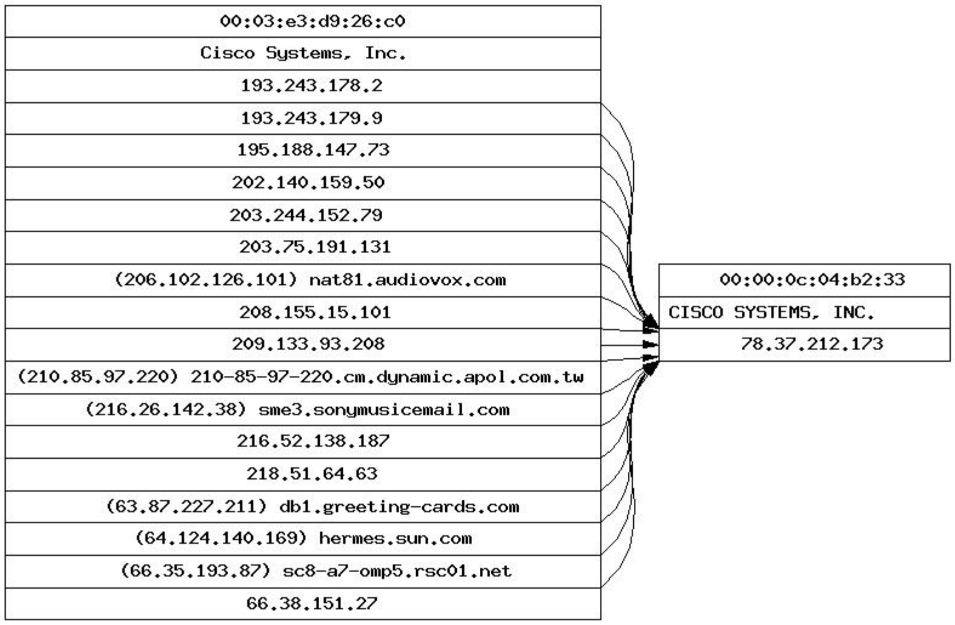


Figure 5: SMTP Destination Port tcpdump filter

References

The following is a complete listing of all references and footnotes contained with this report.

Tcpdump MAN page. URL: http://www.tcpdump.org/tcpdump_man.html (October 1, 2004)
IEEE OUI Search Website. URL: http://standards.ieee.org/regauth/oui/index.shtml (October 17, 2004)
AmigaForever Website. URL: http://www.amigaforever.com/kb/4-109.html (October 11, 2004)
Cisco NAT Setup URL: http://www.cisco.com/warp/public/556/12.html#topic3 (October 5, 2004)
Kurt Seifried Website. URL: http://www.seifried.org/security/ports/1000/1080.html (October 1, 2004)
CA Spyware Encyclopedia Website. URL: http://www3.ca.com/securityadvisor/pest/pest.aspx?id=453076483 (September 25, 2004)
RedHat Errate Website. URL: http://rhn.redhat.com/errata/RHSA-2003-101.html (August 14, 2004)
PHP Website. URL: http://bugs.php.net/bug.php?id=16128 (August 14, 2004)
Incident Mailinglist. URL: http://lists.jammed.com/incidents/2001/05/0038.html (September 14, 2004)
Mike Wyman GCIA Paper. URL: http://www.dshield.org/pipermail/intrusions/2003-January/006748.php (October 18, 2004)
Danny Walker GCIA Paper. URL: http://www.dshield.org/pipermail/intrusions/2002-November/005972.php (October 17, 2004)
Les Gordon, Intrusion Detection – Director's Cut. URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc (October 20, 2004)
Nessus ID: 10028. URL: http://cgi.nessus.org/plugins/dump.php3?id=10028 (November 1, 2004)
Whitehats ID: 278. URL: http://www.whitehats.com/info/IDS278 (October 31, 2004)
Mike Ellis GCIA post. URL: http://www.dshield.org/pipermail/intrusions/2003-April/007411.php (August 12, 2004)
Ian Marks GCIA post. URL: http://lists.sans.org/pipermail/intrusions/2004-July/008200.html (August 12, 2004)
Patrick Nolan CERT Post. URL: http://cert.uni-stuttgart.de/archive/intrusions/2002/04/msg00124.html (November 1, 2004)
Symantec Security Response: W32.KLEZ. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.klez.e@mm.html (November 15, 2004)
cut manual page. URL: http://www.rt.com/man/cut.1.html (September 28, 2004)

sort manual page. URL: http://www.rt.com/man/sort1.html (September 28, 2004)
uniq manual page. URL: http://www.rt.com/man/uniq.1.html (September 28, 2004)
tr manual page. URL: http://www.rt.com/man/tr.1.html (September 28, 2004)
tcpdump2dot Website. URL: http://www.grotto-group.com/~gulfie/projects/monitoring/tcpdump2dot/ (November 14, 2004)
GraphViz Website. URL: http://www.grpahviz.org (November 10, 2004)