Global Information Assurance Certification Paper

Intrusion Detection In-Depth

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 3.4

Ryan Rathe

December 15th, 2003

Table of Contents

1. 802.11b Wireless Security and Intrusion Detection

1.1.1 - Executive Summary

This paper is designed to quickly bring a technical person up to speed on the IEEE 802.11b Wireless LAN (Local Area Network) by giving you a basic overview of the technology, its security shortfalls, and common methods used to better secure it. The goal is to lead you to the conclusion that with all of the current security shortfalls, we need WLAN Intrusion Detection Capabilities. I'll then get you started with a couple of open-source WIDS (Wireless Intrusion Detection System) systems that you can experiment with in a test environment. Please note that these systems are not ready for enterprise deployment. If you're looking for such, you should consider a commercial product such as Airdefense. Lastly, we'll talk about the future of IDS, where it's going and what we can expect to see.

1.1.2 - Enter the Wireless LAN

In recent years, the Wireless LAN has emerged, adding many new capabilities to the once seemingly landlocked network infrastructure. With the wireless industry standardizing on IEEE 802.11b, users can now roam from their office to the conference room or to their home patio with little configuration required and without worry of inter-brand compatibility. Warehouse inventory quantities can be kept in real-time with the integration of wireless cards into barcode scanners. For the Network Administrator, all the headaches of running cables, drilling holes, and installing jacks can be alleviated with just one cable run, a simple web based AP (Access Point) configuration, and plug-and-play wireless network cards.

1.1.3 - Built-In Insecurities

Like many other computer technologies, convenience and ease of use doesn't come without a price. Wireless adds an addition layer of security concern in that wireless is physically a broadcast medium whereas anyone within range (across the street, down the block, or miles away) could potentially pick up the signal and gain access to your network. Moreover, many out-of-the-box APs come preconfigured without WEP (Wireless Equivalent Protocol) encryption enabled, with default SSIDs (Service Set Identifier), and with a built in DHCP (Dynamic Host Configuration Protocol) server, which if used "as-is", will allow anyone with an 802.11b card to connect instantly. Realizing this danger, Network Administrators often configure a unique SSID, turn broadcast SSID off, and disable the built in DHCP server to secure a wireless network. It's a good start; however, without encryption the values are being sent in clear text. As a result, such popular wardriving (actively sniffing for vulnerable wireless networks) tools as Kismet or NetStumbler can easily sniff out these SSID and channel values in a matter of seconds. Last but not least, WEP encryption itself is insecure. Using the RC4 encryption algorithm, WEP was designed to protect an 802.11b network from eavesdropping. However, using such tools as AirSnort or WEPCrack, an attacker can discover enough weak keys to crack a 128-bit shared key in a

matter of days or possibly hours. More information on this topic can be found at http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf and http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html .

1.1.4 - The Back Door

Simple oversights often become the downfall to even the most highly secured networks. One of the biggest oversights in traditional LANs are dial-up modems commonly installed on department PCs so that the user can access his or her work from home. The problem is that this gives an attacker a channel of access where they could island-hop their way through the entire corporate network. Just like the LAN, the WLAN also has its share of backdoors. One such wireless backdoor is an AP that was either covertly installed by an attacker or more likely one that was purchased by a user for personal or departmental use without the guidance of the Network Administrator. Without the Network Administrator's guidance it's likely that the user will install the AP with out-of-box default configurations and henceforth the AP can be wide open to attack. This gaping security hole can be largely prevented by educating users about these security concerns; however, this doesn't prepare for the possibility that a rogue AP may be installed with malicious intent. Also, if an attacker can physically access a legitimate AP (i.e. AP under someone's desk) it's possible that the attacker could plug in directly to the AP and discover the configuration or simply reset and reconfigure it. This attack can be largely thwarted by configuring an administrator password and ensuring that it is physically hidden and out of reach. Other backdoors involves wireless clients, generally not recognized as targets. If permitted, it's possible for clients to connect up directly to other clients (Ad-Hoc) without the use of an AP. Also, it's possible for the attacker to attempt to connect to other wireless clients using the AP as a hub. Attacks like these can expose a wireless client to a large array of IP based attacks. Once the client is compromised the attacker can use that machine as a springboard to launch other attacks.

1.1.5 - Denial of Service Attacks

Gaining access to a network may not be the motivation at all. Other wireless attacks aim at causing network congestion or intercepting traffic in a man-in-the-middle style of attack. Well known wireless Denial of Service attacks include Auth Floods, Deauth Floods, General Volume Floods, FakeAP, WLAN-jack and Fatajack. Some of these attacks simply chew up bandwidth and increase latency while others, such as Fatajack, disconnect wireless clients by sending 802.11b Authentication Failed Packets. For example, Fatajack spoofs the source MAC address to appear as if it came from the legitimate AP. This in effect tears down the client session and can cause the client driver to crash, requiring a reboot. Many APs and wireless clients are powerless to this sort of attack and with virtually no logging/alerting capabilities available on APs, this becomes an even greater challenge to overcome. The biggest challenge is that many APs and clients are vulnerable, and current technologies do little to prevent these attacks. The worst part is that because these attacks occur at layer 1-2 of the 802.11,

protocol they can't be detected by a traditional Network IDS. More information on these DoS attacks can be found at http://packetstormsecurity.nl/wireless/ .

1.1.6 - Basic Ways of Securing an 802.11b Network

Several different approaches can be followed in attempting to secure a wireless network and depending on the type of environment (i.e. Government vs. Saw Mill) realistic threat assessments vary. At the very least, organizations should consider enabling 128-bit WEP and should have a unique SSID. After all, it's free and is standard on most APs and wireless cards. It is true that WEP can be compromised, but it's not an easy task. Not only does it take time to crack the key, but AirSnort and WEPCrack are far from plug and play. For the most part they need to be built on Linux, have a short list of supported wireless cards, and have many dependencies. For the novice wardriver this can be a daunting task, resulting in moving onto an easier target that doesn't use WEP encryption. Even the high security environment (i.e. Government) that has other wireless security systems in place should consider enabling 128-bit WEP because it constitutes another layer of security and will keep the kiddies at bay. Granted it's not very secure, yet it's another hurdle for the attacker to overcome. Different security technologies can compliment each other just as the moat compliments the castle wall.

1.1.7 - Advanced Ways of Securing an 802.11b Network

Moving beyond the basics let's look at some other ways we can better secure a wireless network. WEP isn't the only advanced security feature built into APs. Many APs have standardized features such as 802.1x Authentication, Dynamic Key Exchange (EAP-TLS), and MAC address based access lists. 802.1x Authentication is a protocol, whereas a client and a Radius authentication server mutually verify each other's identity before a connection can be established. Also, APs can be required to authenticate their identity which helps to prevent Rogue AP man-in-the-middle and MAC-spoofing based attacks. In addition, 802.1x specifies a method of dynamic per-user/per-session key exchange, thus making WEP a much more secure encryption protocol to deploy. In this environment the keys are constantly changing which makes tools like AirSnort and WEPCrack all but useless as they function by algorithmically finding similarities in weak IV keys that were all generated by the same base WEP key. Many APs also have the capability to block access based on MAC address. This can be useful; however, an attacker can easily subvert this by spoofing the MAC identity of a legitimate client. MAC addresses can be often times be easily changed as most wireless cards support this capability. Furthermore, obtaining a legitimate MAC can be a trivial task by using such tools as a wireless sniffer or with a simple brute-force script. Independent of the AP, a commonplace security mechanism includes establishing a VPN across the wireless link. In today's environment, this is perhaps the best way to secure sensitive data on wireless network. Using IPSEC with WEP is likely to deter even the most persistent of attackers as it is very difficult to compromise. More information on this topic can be found at

http://www.cisco.com/warp/public/cc/pd/witc/ao350ap/prodlit/a350w_ov.htm .

1.1.8 - Wireless LAN (802.11b) Summary
In summary there are several ways an attacker can compromise or disrupt an 802.11b wireless network. We've discussed some of the most common vulnerabilities and what can be done using current technologies. Employing unique SSIDs, WEP, MAC tables, 801.1x, and VPN tunnels will go along ways in securing a WLAN. However with all the security mechanisms discussed one security facet is evidently missing, Intrusion Detection. Most of the wireless attacks discussed occur at layer 1 and 2 of the 802.11 protocol and can not be detected by most Network IDS systems and that's what well discuss next.

1.2.1 - Wireless IDS Introduction
Network Intrusion Detection in its raw form (before Session-sniping and Inline IDS) doesn't stop attackers but rather just alerts you to their presence. Wireless IDSs have the same purpose and goal as the traditional NIDS - to generate alerts for the proactive Security Administrator to interpret and act upon. However, a Wireless IDS differs from a Network IDS in how and what it analyzes. Traditional Network IDSs, such as Snort, focus on analyzing mostly layers 3-7 of the OSI model. Snort simply wasn't designed for 802.11. Many wireless IDSs attempt to fill the gap by focusing specifically on 802.11 analysis while leaving layer 3-7 analysis to a separate IDS (possibly Snort) still others such as Snort-Wireless, can potentially perform analysis at all layers.

Wireless IDS capabilities can be built on just about any hardware platform. The only requirement is that the machine be capable of housing a standard 802.11b wireless card. It's even possible to do this on the AP itself. Deploying these capabilities on the AP has many obvious advantages, such as utilizing what you already have and avoiding what could potentially be one Wireless IDS machine per AP due to wireless signal range limitations. There is work being done on an open source AP WIDS. Following is an excerpt taken from www.snort.org.

Snorting from wireless access points? **Brian @ Thu Aug 28 12:55:09 2003 GMT**

Jim Buzbee has been working on getting Snort up and running on the Linksys WRT54G access point. He's got snort and libpcap cross compiled for the MIPS platform and its up and running in sniffing mode.

This is Jim's first time snorting, so he needs some help building a config for IDS mode. The WRT54G has a 125 Mhz MIPS processor, 16 meg of ram, and runs a slightly modified 2.4.5 kernel. Since its a small box, recent features in snort such as *lowmem* will really helpful.

Check out Jim's experience and grab his Snort binary if you have one of these access points.

Wireless IDSs are still in their early stages and, for the most part, are unrefined and clumsy. In the past year a number of commercial WIDS systems have been introduced into the market, many of which would be well suited to the task. However, in the spirit of open source, I'd like to discuss what is available under a GNU license. I'll introduce you to WIDZ and Snort-Wireless, two open source WIDS systems that can run on an Intel platform machine.

1.3.1 - Enter WIDZ

WIDZ was one of the first open source Wireless IDS systems available to the public. WIDZ focuses on three major risks: unauthorized APs, SSID probing (wardriving), and various DoS attacks. The latest release (WIDZv1.8) shores up a lot of bugs in version 1.5 and adds some new functionality; however, WIDZ is still "proof of concept" and under continuous development by its creator Mark Osborne (www.loud-fat-bloke.co.uk). The future release of 2.0 is expected to bring WIDZ out of the "proof-of-concept" phase and into use. At this humorous website you'll find the download, articles, and some basic documentation.

1.3.2 - Why WIDZ

Although WIDZ is far from perfect and may require a lot of customization, WIDZ does have some features that can enhance an organization's IDS strategies. WIDZ is capable of alerting to the presence of rogue/unauthorized APs and unauthorized clients, and alerting to SSID probing (i.e. Net Stumbler, Kismet).

1.3.3 - WIDZ HOWTO

Like most open source, WIDZ can be built on different platforms; however, documentation is scarce. This HOWTO will guide you through building WIDZ on Red Hat 9.0 with an Orinoco Gold Card. The creator of WIDZ built it on Mandrake Linux. At the moment there is very little documentation for WIDZ. The first hurdle to overcome is getting a wireless card to operate in monitor mode. Some cards come with drivers that don't support promiscuous sniffing. Fortunately, having experience with AirSnort, Kismet, and the like, I've already become familiar with the related driver issues. The second minor hurdle is to get his GUI (Graphical User Interface) to run on your GUI (Mandrake vs. Red Hat). This issue is easily fixed by substituting his gmessage entries with xmessage. Upon getting through these two issues, I found that WIDZ was fairly easy to use and configure. Ok, so let's start.

1.3.4 - How I Built WIDZ

1.3.4.1 - WIDZ: Hardware Requirements

a. Pentium Based PC (I used a Dell Inspiron 1100 laptop)
b. Orinoco Gold (Agere Systems)
   -Other cards may work as well. AirSnort and WIDZ have the same monitor mode requirements so an excellent resource for getting various cards to work in monitor mode (promiscuous) mode can be found at http://airsnort.shmoo.com

1.3.4.2 - WIDZ: Need to Download to /root/

a. David Gibson's Orinoco Drivers v13e (latest drivers)
   http://ozlabs.org/people/dgibson/dldwd/orinoco-0.13e.tar.gz
b. Patch for v13e Drivers (to support monitor mode)
   http://airsnort.shmoo.com/orinoco-0.13e-patch.diff
   -from your browser do a save as to download it
c. WIDZ v1.8
   http://www.loud-fat-bloke.co.uk/tools/widzv1.8.zip
d. libpcap
   http://www.tcpdump.org/daily/libpcap-current.tar.gz

1.3.4.3 - Red Hat 9.0

Let's assume that you already have Red Hat 9.0 Installed. If you don't, there is plenty of install documentation that can be found at www.redhat.com . You can run WIDZ on both server and client installation types. If you're going to run WIDZ in a GUI, you'll need X Windows and Gnome Desktop. In "Add-Remove-Packages" you might want to have the following installed: X Windows, Gnome Desktop, Editors, Graphical Internet, and Development Tools. To add items simply check the checkboxes and click update when done.

1.3.4.4 - Install Orinoco Drivers

If you haven't gotten your Orinoco card working with the factory drivers, be sure to do that first. Kudzu should detect it upon a reboot and install the factory drivers automatically.

Once this is done, we'll need to download the drivers and the patch. Important: Download both of these to the /root/ directory. The patch will show up in your browser window. To download it, go to "File" and select "Save As", then specify the /root directory:
Drivers: http://ozlabs.org/people/dgibson/dldwd/orinoco-0.13e.tar.gz
Patch: http://airsnort.shmoo.com/orinoco-0.13e-patch.diff

Install the Orinoco drivers AS ROOT:

```
# cd  /root
# tar –zxf orinoco-0.13e.tar.gz
# patch –p0 < orinoco-0.13e-patch.diff
```

This is what you should see:

```
patching file orinoco-0.13e/hermes.c
patching file orinoco-0.13e/hermes.h
patching file orinoco-0.13e/orinoco.c
patching file orinoco-0.13e/orinoco.h
```

```
# cd orinoco-0.13e
```

make
make install

Now well need to overwrite the old driver files with the new ones, but before we do that, let's back up the old drivers. I backed them up to the /root directory because backing them up in the /lib/modules could cause the old drivers to be loaded. Between and not including the brackets is where you'd insert your kernel version (i.e. mine is 2.4.20-19.9):
cd /lib/modules/[kernel version]/kernel/drivers/net/
mv wireless /root
mkdir wireless
cp –fv /root/orinoco-0.13e/* wireless
reboot

Now once you've rebooted, let's test the drivers:
dmesg | more
In the output, about halfway down, you should see some lines which look similar to this:
orinoco.c 0.13e (David Gibson hermes@gibson.dropbear.id.au> and others)
orinoco_cs.c (David Gibson hermes@gibson.dropbear.id.au> and others)

Also a few lines down you should be able to see you firmware version. Note- you firmware version doesn't have to be 6.16, most versions should work fine:
eth0: Looks like a Lucent/Agere firmware version 6.16

Next, we'll want to test for the presence of monitor mode in ioctl, In the output you should see a line for monitor:
iwpriv eth0 (note: substitute eth0 with what your wireless card is)

1.3.4.5 - Install libpcap (if you don't have it already)

Go to http://www.tcpdump.org/daily/libpcap-current.tar.gz and download it to /root/ then:
cd /root
tar –zxf libpcap-current.tar.gz
ls (jot down what the tar archive was extracted as)
cd [substitute name of extracted directory]
./configure
make
make install

1.3.4.6 - Install WIDZ v1.8

Go to http://www.loud-fat-bloke.co.uk/tools/widzv1.8.zip and save it to /root/. Next, uncompress and extract it like this:
cd /root
unzip widzv1.8.zip
mkdir widz

```
# cd widz
# tar –zxf /root/widzv1.8.tgz
```

1.3.4.7 - Configuring WIDZ with the Correct Interface
Now let's configure WIDS. To start, the scripts all reference wlan0 as the interface to sniff on; however, this can vary (i.e. mine is eth0). Use whatever your wireless interface is (use the ifconfig command to find out). The proper way to fix this problem would be to use a variable for this; however, I simply did a search and replace for every occurrence of wlan0 to be replaced with eth0. To do this you'll need to edit the startprobemon and startapmon file using vi to give this vi command:
:g/wlan0/s//eth0/g

1.3.4.8 - Gmessage vs. Xmessage
After installing and configuring WIDZ, I launched WIDZ with ./startprobemon and was greeted by Mark's smirky face in the GUI and two buttons: one for probemon and one for the other module apmon. Upon clicking anywhere on the screen, it went directly into probemon logging mode. The Probemon GUI doesn't really have much purpose anyways, so I didn't go to a great deal of effort to try to fix the issue. To disable the GUI, I hashed out the "xwud –in lfb" entry in the startprobemon file (command prompts rule). However, there is yet another GUI element that may be useful, xmessage pop-up alerts. In the apmon directory there are two files, Alert and startapmon where you'll find a reference to a program called gmessage. You'll also find gmessage in the startprobemon file located in the probemon directory. Unfortunately, my Red Hat 9.0 doesn't have gmessage. Perhaps this is something standard on Mandrake. I don't know. To solve this, I simply changed all of the gmessage entries to xmessage and Voila. It works beautifully. To fix it, you'll need to replace the g with an x in the three files mentioned above.

1.3.4.9 - The > Bug
Another bug I discovered was the use of > as opposed to < in the output statements found in the Alert files. The Alert files for both probemon and apmon need to be fixed to echo output to the tty and /dev/console. The redirects should point to the left. As shown below:

```
echo $mess  <  'tty'
echo $mess  <  /dev/console
```

1.3.4.10 - The Second Argument Bug
This largely depends on the drivers, but in my case to put the card in monitor mode (promiscuous) requires not one but two private-parameters following the monitor private-command. The problem is that in startprobemon and startprobemon, the wireless card is brought into monitor mode with only one parameter. The solution is to simply add the second parameter to each of these

startup files. It should look something like this whereas the second parameter equals the wireless channel you're looking to monitor on:

iwpriv eth0 monitor 2 11

1.3.4.11 - Apmon Configuration
Starting with apmon, you'll need to edit widz-ap.config (a whitelist of authorized APs). In this file you'll want to add all of your authorized APs, one AP per line, giving the ESSID and MAC address for each AP. With this completed, apmon will alert you to any AP MAC address it discovers that isn't in that file. Next we'll need to set up alerts. Within the Alert file you'll find canned entries for echoing output to the console, syslog, sendmail, or snmp. Simply remove the hash for whatever method you choose for alerting. For example, I wanted to consolidate this along with my Swatch alerts so I removed the hash for logger and modified my /etc/ syslog.conf file by adding this line:

Security.notice @loghost

Then I added this to my /etc/hosts file:
1.1.1.1 syslogmaster.fakedomain.com loghost

1.3.4.12 - Probemon Configuration
To configure probemon, you'll have to decide what features you want to use. Probemon.conf has four optional features: usesbadmacs, usebadssids, usegoodmacs, and usescripts. Each of these can be toggled on and off with a y or n value. Usebadmacs allows you to specify bad MAC addresses. With this feature enabled, you will receive alerts if the specified bad MAC addresses are discovered. For example, this could be useful in monitoring the activities of a suspicious employee. With userbadssids you could put together a list of SSIDs that will generate an alert if the SSID value is discovered in a frame. For example you could put default SSIDs in this list, thereby alerting to unconfigured clients. However, this may generate a lot of false positives. Then there's usescripts which will allow you to deploy your own scripts (placed in the scripts directory) within WIDZ. Lastly, and perhaps the most useful, is usegoodmacs. Here we're creating a whitelist of all authorized wireless card MAC addresses. An Alert will be generated if WIDZ discovers a MAC address on the wireless medium that doesn't match one of the entries in this file.

1.3.4.13 - Running WIDZ
WIDZ has two modules and unfortunately they don't work well together. I was forced to run either one or the other. Running both at the same time would cause them to function abnormally. The creator of WIDZ has reported the same kind of problem. To get them running normally again, you should first kill apmon and probemon. Then at the command prompt enter the following command to bring the wireless card out of monitor mode. (Substitute the interface and channel that applies):

iwpriv eth0 monitor 0 11

1.3.4.14 - Orinoco Firmware Links
Should you have any problem with getting into monitor mode (iwpriv eth0), here are a couple of links for downgrading and upgrading the Orinoco firmware. Note: These are windows executables and will need to be installed from a Windows laptop. Also, one thing I discovered is that in order to downgrade firmware you may need to go all the way back down to 6.06 and then upgrade to the higher version.

6.06__ftp://ftp.orinocowireless.com/pub/software/ORiNOCO/PC_Card/Firmware/WSU_606.exe
6.16__ftp://ftp.orinocowireless.com/pub/software/ORiNOCO/PC_Card/Firmware/R6.4winter2001/WSU_616.exe
7.28__ftp://ftp.orinocowireless.com/pub/software/ORiNOCO/PC_Card/Firmware/R7.0spring2001/WSU_728.exe
7.52__ftp://ftp.orinocowireless.com/pub/software/ORiNOCO/PC_Card/Firmware/R7.1summer2001/WSU_752.exe
8.10__ftp://ftp.orinocowireless.com/pub/software/ORiNOCO/PC_Card/Firmware/R7.4winter2002/WSU_810.exe

1.4.1 - Enter Snort-Wireless
Introduced publicly just in the past year or so, Snort-Wireless for the most part is still on the drawing board. Andrew Lockhart has hosted a site at www.snort-wireless.org that contains full downloads and patches which can be applied to Snort 2.0.x. There is also some good documentation on creating rules with the WiFi protocol and information about what is currently under development. The README.wireless document will give you a good indication of the general status of the development effort.

1.4.2 - Why Snort-Wireless
Snort-Wireless is still in the alpha stages of development and isn't ready for large-scale deployments. However, Snort-Wireless has the advantage of being started on a proven foundation. Snort-Wireless is really just Snort 2.0 with some preprocessors for the 802.11 protocol and some added rule making capabilities that allow you to make your own rules based on 802.11 frame analysis. Snort has a preprocessor for identifying and alerting to Net Stumbler type scans and a preprocessor for identifying rogue APs and Adhoc networks. Snort-Wireless is also capable of performing the traditional layer 3-7 analysis in addition to 802.11b giving you a comprehensive and scalable IDS solution for a wireless LAN.

1.4.3 - Building Snort-Wireless
To install Snort-Wireless, you need a few things: a Red Hat capable PC, a wireless card and a cursory knowledge of Snort. In this installation I already had WIDZ installed and configured (as you may have by now) so throughout this section I'll often refer to the WIDZ HOWTO for the sake of brevity.

1.4.4 - How I built Snort-Wireless
The official test configurations from the Snort-Wireless website reference Red Hat 7.3 and 9.0 with the AirJack and HostAP drivers. I built it with AirJack v0.6.6balpha on Red Hat 9.0 with an Orinoco Gold Card (Hermes drivers). Note everything is downloaded, compiled, installed, and run as root. Here is the download site http://802.11ninja.net/ .

1.4.4.1 - Snort-Wireless Hardware Requirements

a. Pentium Based PC (I used a Dell Inspiron 1100 laptop)
b. Orinoco Gold (Agere Systems)
   -Other cards may work as well. AirSnort and Snort-Wireless have the same monitor mode requirements so an excellent resource for getting various cards to work in monitor mode (promiscuous) mode can be found at: http://airsnort.shmoo.com .

1.4.4.2 - Snort-Wireless: Need to Download to /root/

a. AirJack drivers
   http://prdownloads.sourceforge.net/airjack/airjack-v0.6.6b-alpha.tar.bz2?download
b. PCMCIA Card Services Package
   http://prdownloads.sourceforge.net/pcmcia-cs/pcmcia-cs-3.2.4.tar.gz?download
c. Snort-Wireless 2.0.1-current
   http://snort-wireless.org/files/snort-wireless-2.0.1-current.tar.gz
d. Bzip2 Compression
   http://sources.redhat.com/bzip2/
e. libpcap
   http://www.tcpdump.org/daily/libpcap-current.tar.gz

1.4.4.3 - Red Hat 9.0
Once again, let's assume you have this installed. If you don't, excellent documentation can be found at www.redhat.com .

1.4.4.4 - Installing Libpcap
If you did the WIDZ HOWTO, then you can skip this section. Otherwise, please refer to 1.3.5.5 for this task.

1.4.4.5 - Install AirJack Drivers
First we need to download to /root and install bzip2 (compression):
http://sources.redhat.com/bzip2/

Now extract and install bzip2
tar –zxf bzip2-1.0.2.tar.gz

cd bzip2-1.0.2
make
make install

Next, download the 3.2.4 pcmcia-cs package to /root:
http://prdownloads.sourceforge.net/pcmcia-cs/pcmcia-cs-3.2.4.tar.gz?download
tar –zxf pcmcia-cs-3.2.4.tar.gz
cd pcmcia-cs-3.2.4
./Configure
Now you'll be prompted for a few things. The first thing you'll see is:
Linux kernel source directory [usr/src/linux]
Obviously in Red Hat this default is wrong. In Red Hat, this directory will be the kernel version # that you're running. For example, my kernel version was 2.4.20-19.9, so I responded with:
/usr/src/linux-2.4.20-19.9
Now you'll be prompted with 3 (y/n) questions. Answer y to all of these.
The last prompt is for the install directory. If the default doesn't show up as /lib/modules/2.4.20-19.9, then specify it as such.

Now that we've configured the pcmcia-cs source we're done. Note that we don't need to do not need to build it because pcmcia is already built into the kernel. We're just configuring it so that AirJack can access and use these files when it's being built. Ok, so now let's build AirJack.

First we need to download to /root and install AirJack:
http://prdownloads.sourceforge.net/airjack/airjack-v0.6.6b-alpha.tar.bz2?download
Now extract and install AirJack
bzip2 –d airjack-v0.6b-alpha.tar.bz2
tar –xf airjack-v0.6b-alpha.tar

cd /root/airjack-v0.6.6b-alpha
vi Makefile
now edit this as such: PCMCIADIR=/root/pcmcia-cs-3.2.4/include
make
make install
cp airjack_cs.o /lib/modules/2.4.20-19.9/kernel/drivers/net/wireless

edit /etc/pcmcia/config
and add a section that looks like this:
device "airjack_cs"
class "network" module "airjack_cs"

Now we want to edit the card to driver bindings which can be found in various files located in /etc/pcmcia. For me this was hermes.conf and if you've followed

my HOWTO exactly so will yours. You may want to back up the files before doing this. It is up to you. Here's what we want to edit in the /etc/pcmcia/hermes.conf . It should look like this when done. Note: to restore normal functionality you'll simply need to unhash the original "bind orinoco_cs" and hash out the "bind airjack_cs" entry. If you don't have a hermes.conf, look in the other conf or opt files for something like this that represents the wireless card:

card "Lucent Technologies WaveLAN/IEEE Adapter"
version "Lucent Technologies", "WaveLAN/IEEE"
bind "orinoco_cs"
bind "airjack_cs"

When done editing save what you just did and reboot:
reboot
Now let's see if it works. Type the following and if you see an aj0 interface you're good:
#/sbin/ifconfig –a
Now we need to bring up the aj0 interface (Red Hat doesn't do that for us)
/sbin/ifconfig aj0 up

1.4.4.6 - Installing Snort-Wireless
Go to http://snort-wireless.org/files/snort-wireless-2.0.1-current.tar.gz and download the latest full version to /root as root. Then open up a shell and as root and type the following:
cd /root
tar –zxf snort-wireless-2.0.1-current.tar.gz
cd snort-wireless-2.0.1-current.tar.gz
./configure
make
make install

1.4.4.7 - Configuring Snort-Wireless
For those of you familiar with Snort you'll find that Installing and configuring Snort-Wireless is not much different. For those not familiar with Snort, the best way to get up-to-speed with the how and what is to read up on the Snort User's Manual found at http://www.snort.org/docs/writing_rules/ .

Here's a list of the Snort-Wireless components that we've added to the Snort foundation. All of these are activated in snort.conf:
1. include $RULE_PATH/wifi.rules
   -the $SNORT_PATH/rules/wifi.rules is where built in 802.11 frame based rules can activated (unhashed) and custom rules can be written
2. var ACCESS_POINTS [Authorized AP MAC Address (comma separated)]
   -this is where you create the list of Authorized APs; example:
   var ACCESS_POINTS 1D:EC:AF:C0:FF:EE, 04:DE:AD:C0:DE:00

3. var CHANNELS [Authorized Channels (comma separated)
   -this is where you specify the channels that correspond with the above AP entries; example: var CHANNELS 11, 6
4. RogueAP Preprocessor
   -alerts to AP MAC addresses that aren't stated in "var ACCESS_POINTS"
5. AntiStumbler Preprocessor
   -alerts to 802.11 probe requests (possible wardriving) that don't have the SSID field set
6. 802.11 Rule Engine (not fully functional)
   -allows you to create Snort rules using source and destination MAC addresses and various 802.11 frame fields and flag keywords

1.4.4.8 - Running Snort-Wireless

Snort-Wireless is a bit buggy and still needs a lot of work. I unhashed all of the WiFi rules present in the wifi.rules just to see how ready they were and upon executing snort from the src directory (so as not to load my normal Snort install) with these commands, I received the following output:

```
# cd /root/snort-wireless-2.0.1-current
#./src/snort –c ./etc/snort.conf –I aj0

Warning: /etc/../rules/wifi.rules(69) => Unknown keyword 'seq_num' in rule!
Warning: /etc/../rules/wifi.rules(70) => Unknown keyword 'seq_num' in rule!
Warning: /etc/../rules/wifi.rules(71) => Unknown keyword 'frag_num' in rule!
Warning: /etc/../rules/wifi.rules(72) => Unknown keyword 'frag_num' in rule!
```

Apparently, all of the WiFi rules in the wifi.rules file are hashed out for a reason and that's because they're just not ready yet.

1.5 - WIDZ vs. Snort-Wireless – Current Features Comparison

Given these are both under development, the yes and no are based on what currently works without intervention. The "Future Plans" vs. "No" classification is based on what has been formally documented as a future feature under development.

Feature	WIDZ 1.8	Snort-Wireless 2.0.1
Rogue AP Detection (Unauthorized)	Yes	Yes
Probing Detection (Null SSID field)	Yes	Yes
Adhoc Network Detection	Yes	Yes
Syslog and Console Alerting	Yes	Yes
Unauthorized Client Detection (MAC)	Yes	See Customizable Rules (WiFi Protocol)
DoS Attack Detection (Auth Fail, Assoc, and Deauth Flood, Fatajack, Wireless Client Attack)	Future Plans	Future Plans
Customizable Rules (WiFi Protocol)	No	Future Plans
Seq # based MAC Spoof Detection	No	Future Plans
Channel Scanning (Multiple Channels)	No	Future Plans
WEP Preprocessor (for layer 3 analysis)	No	Future Plans
Flexible Response (Session Sniping)	No	Future Plans
ACID Support	No	Future Plans

1.6 - Conclusion and the Future of Wireless Intrusion Detection

In this paper we discussed current 802.11b technologies, insecurities, and several methods to better secure a wireless network. We discussed the most common 802.11b vulnerabilities and some methods to better secure a wireless network. Moreover, the emphasis was on the need for Wireless Intrusion Detection and we introduced a couple of different open source WIDS systems that contribute to the progress of Wireless Intrusion Detection as a whole. Current technologies such as WIDZ and Snort-Wireless are rough around the edges, yet can still prove to have some value from learning, testing, and development perspectives. Deploying Wireless IDS systems such as WIDZ or Snort-Wireless in their current form can be very complex and costly for an organization because each AP will likely require its own Wireless IDS. This is because a WIDS uses the same 802.11b wireless card with the same range limitations as that of the AP. A possible solution to this would be for the AP vendors to include flexible WIDS functionality within the AP itself. Adding IDS functionality may require a little extra memory and processing power; however, it could give those vendors a competitive advantage to enterprise customers. Regardless of what AP features come about, I would expect that a stable open-source standard will emerge in the coming years and Snort-Wireless is likely to be that standard. After all, it's really just Snort with a couple of added preprocessors and a new "WiFi" rules protocol. The foundation of Snort is proven and with the Snort community behind it, Snort-Wireless will likely have the development contributions necessary to make it yet another, great IDS.

1.7 - References

WIDZ:
[1]Mark Osbourne. Author of WIDZ
URL: www.loud-fat-bloke.co.uk (Dec 6, 2003)

Snort Wireless:
[2]Snort.wireless.org. Author Andrew Lockart, Nov 16 2003
URL: www.snort-wireless.org (Dec 6, 2003)

Snort:
[3]Snort.org T.M. of Sourcefire, Inc. Authors, Brian Caswell and Marty Roesch 2002, 2003
URL: www.snort.org (Dec 6, 2003)

Airsnort:
[4]Airsnort.shmoo.org, Page hosted by SOurcefire, Website host, snax@shmoo.com
URL: http://airsnort.shmoo.com (Dec 6, 2003)

Airjack Drivers:
[5]Aijack Drivers. Developed by Robert Baird (aka xx25) and Michael Lynn (aka Abaddon)
URL: http://802.11ninja.net/ (Dec 6, 2003)

Advanced AP Security Features:
[6]Cisco Systems, Inc. Copyright © 1992--2002
URL: http://www.cisco.com/warp/public/cc/pd/witc/ao350ap/prodlit/a350w_ov.htm (Dec 6, 2003)

WEP and WEP Algorithm Weaknesses:
[7]Cisco Systems, Inc. Authors retain rights, Scott Fluhrer, Itsik Mantin, Adi Shamir
URL: http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf (Dec 6, 2003)

WEP and Advanced Attacks:
[8]University of California, Berkley. Authors, Nikita Borisov, Ian Goldberg, and David Wagner
URL: http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html (Dec 6, 2003)

Various Wireless Tools:
[9]Packet Storm. 2003
URL: http://packetstormsecurity.nl/wireless/ (Dec 6, 2003)

2.0 - Three Network Detects

The three detects that I analyzed for this assignment were 2002.9.9, 2002.8.23, and 2002.9.15. All three attacks were taken from the incidents.org raw log files located at http://www/incidents.org/logs/Raw[1].

2.1 - Detect 1, IIS _vti_inf.html and _vti_rpc Access

2.1.1 - Source of Trace:

This trace comes from the raw logs at http://www.incidents.org/logs/Raw/[1] . download file: 2002.9.9

2.1.2 - Detect Generated By: Snort v.1.8.6

The signatures that generated the relevant alerts follow:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS _vti_inf
access";flow:to_server,established; uricontent:"_vti_inf.html"; nocase; classtype:web-application-activity;
sid:990; rev:5;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-FRONTPAGE
_vti_rpc access"; flow:to_server,established; uricontent:"/_vti_rpc"; nocase; reference:bugtraq,2144;
classtype:web-application-activity; sid:937; rev:6;)
```

The relevant Snort alerts follow:

```
[**] [1:990:2] WEB-IIS _vti_inf access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-07:07:31.926507 67.96.81.242:1135 -> 32.245.166.119:80
TCP TTL:118 TOS:0x0 ID:45381 IpLen:20 DgmLen:304 DF
***AP*** Seq: 0xD8FB6E79  Ack: 0x721A4106  Win: 0x2238  TcpLen: 20

[**] [1:937:3] WEB-FRONTPAGE _vti_rpc access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-07:07:33.126507 67.96.81.242:1135 -> 32.245.166.119:80
```

TCP TTL:118 TOS:0x0 ID:2118 IpLen:20 DgmLen:438 DF
AP Seq: 0xD8FB6F81 Ack: 0x721A5168 Win: 0x1D3E TcpLen: 20
[Xref => http://www.securityfocus.com/bid/2144]

[**] [1:990:2] WEB-IIS _vti_inf access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-07:10:31.976507 67.96.81.242:1558 -> 32.245.166.119:80
TCP TTL:118 TOS:0x0 ID:52321 IpLen:20 DgmLen:304 DF
AP Seq: 0xDD97B728 Ack: 0x7D4775F4 Win: 0x2238 TcpLen: 20

[**] [1:937:3] WEB-FRONTPAGE _vti_rpc access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-07:10:33.306507 67.96.81.242:1558 -> 32.245.166.119:80
TCP TTL:118 TOS:0x0 ID:62817 IpLen:20 DgmLen:438 DF
AP Seq: 0xDD97B830 Ack: 0x7D478656 Win: 0x1D3E TcpLen: 20
[Xref => http://www.securityfocus.com/bid/2144]

[**] [1:990:2] WEB-IIS _vti_inf access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-10:35:28.226507 64.174.39.98:1492 -> 32.245.166.119:80
TCP TTL:110 TOS:0x0 ID:5034 IpLen:20 DgmLen:305 DF
AP Seq: 0x12C25AC8 Ack: 0x83C1600F Win: 0x4470 TcpLen: 20

[**] [1:937:3] WEB-FRONTPAGE _vti_rpc access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-10:35:36.686507 64.174.39.98:1502 -> 32.245.166.119:80
TCP TTL:110 TOS:0x0 ID:5085 IpLen:20 DgmLen:430 DF
AP Seq: 0x12F7527D Ack: 0x84869255 Win: 0x4470 TcpLen: 20
[Xref => http://www.securityfocus.com/bid/2144]

[**] [1:990:2] WEB-IIS _vti_inf access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-10:38:41.806507 64.174.39.98:1824 -> 32.245.166.119:80
TCP TTL:110 TOS:0x0 ID:9018 IpLen:20 DgmLen:69 DF
AP Seq: 0x17230B01 Ack: 0x8FC4BA6E Win: 0x4470 TcpLen: 20

[**] [1:937:3] WEB-FRONTPAGE _vti_rpc access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-10:38:45.706507 64.174.39.98:1826 -> 32.245.166.119:80
TCP TTL:110 TOS:0x0 ID:9046 IpLen:20 DgmLen:84 DF
AP Seq: 0x1732E74E Ack: 0x903D1AA1 Win: 0x4470 TcpLen: 20
[Xref => http://www.securityfocus.com/bid/2144]

[**] [1:990:2] WEB-IIS _vti_inf access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-10:39:51.396507 64.174.39.98:1900 -> 32.245.166.119:80
TCP TTL:110 TOS:0x0 ID:10050 IpLen:20 DgmLen:305 DF
AP Seq: 0x189B0463 Ack: 0x938F193D Win: 0x4470 TcpLen: 20

[**] [1:937:3] WEB-FRONTPAGE _vti_rpc access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-10:39:52.016507 64.174.39.98:1901 -> 32.245.166.119:80
TCP TTL:110 TOS:0x0 ID:10060 IpLen:20 DgmLen:430 DF
AP Seq: 0x189E1A48 Ack: 0x93B010EF Win: 0x4470 TcpLen: 20
[Xref => http://www.securityfocus.com/bid/2144]

[**] [1:990:2] WEB-IIS _vti_inf access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-15:26:46.656507 202.18.172.35:25755 -> 32.245.166.119:80
TCP TTL:41 TOS:0x0 ID:15871 IpLen:20 DgmLen:332
AP Seq: 0xEC81D53E Ack: 0xCF5F2EE8 Win: 0x2238 TcpLen: 20

[**] [1:937:3] WEB-FRONTPAGE _vti_rpc access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-15:26:47.776507 202.18.172.35:25769 -> 32.245.166.119:80
TCP TTL:41 TOS:0x0 ID:16400 IpLen:20 DgmLen:431
AP Seq: 0xEC95E3D1 Ack: 0xCF722D10 Win: 0x2238 TcpLen: 20
[Xref => http://www.securityfocus.com/bid/2144]

[**] [1:990:2] WEB-IIS _vti_inf access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-16:20:02.546507 218.17.203.54:38225 -> 32.245.166.119:80
TCP TTL:44 TOS:0x0 ID:47906 IpLen:20 DgmLen:298 DF
AP Seq: 0xFA3E5D2D Ack: 0x9830C2F1 Win: 0x4197 TcpLen: 20

[**] [1:937:3] WEB-FRONTPAGE _vti_rpc access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-16:20:03.386507 218.17.203.54:38226 -> 32.245.166.119:80
TCP TTL:44 TOS:0x0 ID:47916 IpLen:20 DgmLen:423 DF
AP Seq: 0xFA45DDC5 Ack: 0x98EEC58C Win: 0x4410 TcpLen: 20
[Xref => http://www.securityfocus.com/bid/2144]

For the purpose of brevity only one set of detailed logs were included. Note that all of the source hosts that generated these alerts have logs entries similar to this:

```
[**] WEB-IIS _vti_inf access [**]
10/09-07:07:31.926507 67.96.81.242:1135 -> 32.245.166.119:80
TCP TTL:118 TOS:0x0 ID:45381 IpLen:20 DgmLen:304 DF
***AP*** Seq: 0xD8FB6E79  Ack: 0x721A4106  Win: 0x2238  TcpLen: 20
47 45 54 20 2F 5F 76 74 69 5F 69 6E 66 2E 68 74  GET /_vti_inf.ht
6D 6C 20 48 54 54 50 2F 31 2E 30 0D 0A 56 69 61  ml HTTP/1.0..Via
3A 20 31 2E 30 20 6D 61 69 6E 0D 0A 43 6F 6E 6E  : 1.0 main..Conn
65 63 74 69 6F 6E 3A 20 4B 65 65 70 2D 41 6C 69  ection: Keep-Ali
76 65 0D 0A 43 6F 6E 74 65 6E 74 2D 4C 65 6E 67  ve..Content-Leng
74 68 3A 20 30 0D 0A 55 73 65 72 2D 41 67 65 6E  th: 0..User-Agen
74 3A 20 4D 6F 7A 69 6C 6C 61 2F 32 2E 30 20 28  t: Mozilla/2.0 (
63 6F 6D 70 61 74 69 62 6C 65 3B 20 4D 53 20 46  compatible; MS F
72 6F 6E 74 50 61 67 65 20 34 2E 30 29 0D 0A 48  rontPage 4.0)..H
6F 73 74 3A 20 77 77 77 2E 58 58 58 58 58 58 58  ost: www.XXXXXXX
58 0D 0A 41 63 63 65 70 74 3A 20 2A 2F 2A 2C 61  X..Accept: */*,a
75 74 68 2F 73 69 63 69 6C 79 0D 0A 50 72 61 67  uth/sicily..Prag
6D 61 3A 20 6E 6F 2D 63 61 63 68 65 0D 0A 44 61  ma: no-cache..Da
74 65 3A 20 57 65 64 2C 20 30 39 20 4F 63 74 20  te: Wed, 09 Oct
32 30 30 32 20 31 37 3A 30 35 3A 30 39 20 47 4D  2002 17:05:09 GM
54 0D 0A 4D 49 4D 45 2D 56 65 72 73 69 6F 6E 3A  T..MIME-Version:
20 31 2E 30 0D 0A 0D 0A                           1.0....

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

[**] WEB-FRONTPAGE _vti_rpc access [**]
10/09-07:07:33.126507 67.96.81.242:1135 -> 32.245.166.119:80
```

```
TCP TTL:118 TOS:0x0 ID:2118 IpLen:20 DgmLen:438 DF
***AP*** Seq: 0xD8FB6F81  Ack: 0x721A5168  Win: 0x1D3E  TcpLen: 20
50 4F 53 54 20 2F 5F 76 74 69 5F 62 69 6E 2F 73  POST /_vti_bin/s
68 74 6D 6C 2E 64 6C 6C 2F 5F 76 74 69 5F 72 70  html.dll/_vti_rp
63 20 48 54 54 50 2F 31 2E 30 0D 0A 56 69 61 3A  c HTTP/1.0..Via:
20 31 2E 30 20 6D 61 69 6E 0D 0A 43 6F 6E 6E 65   1.0 main..Conne
63 74 69 6F 6E 3A 20 4B 65 65 70 2D 41 6C 69 76  ction: Keep-Aliv
65 0D 0A 43 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74  e..Content-Lengt
68 3A 20 34 31 0D 0A 55 73 65 72 2D 41 67 65 6E  h: 41..User-Agen
74 3A 20 4D 53 46 72 6F 6E 74 50 61 67 65 2F 34  t: MSFrontPage/4
2E 30 0D 0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65  .0..Content-Type
3A 20 61 70 70 6C 69 63 61 74 69 6F 6E 2F 78 2D  : application/x-
77 77 77 2D 66 6F 72 6D 2D 75 72 6C 65 6E 63 6F  www-form-urlenco
64 65 64 0D 0A 48 6F 73 74 3A 20 77 77 77 2E 58  ded..Host: www.X
58 58 58 58 58 58 58 0D 0A 41 63 63 65 70 74 3A  XXXXXXX..Accept:
20 61 75 74 68 2F 73 69 63 69 6C 79 0D 0A 50 72   auth/sicily..Pr
61 67 6D 61 3A 20 6E 6F 2D 63 61 63 68 65 0D 0A  agma: no-cache..
44 61 74 65 3A 20 57 65 64 2C 20 30 39 20 4F 63  Date: Wed, 09 Oc
74 20 32 30 30 32 20 31 37 3A 30 35 3A 31 31 20  t 2002 17:05:11
47 4D 54 0D 0A 4D 49 4D 45 2D 56 65 72 73 69 6F  GMT..MIME-Versio
6E 3A 20 31 2E 30 0D 0A 58 2D 56 65 72 6D 65 65  n: 1.0..X-Vermee
72 2D 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20  r-Content-Type:
61 70 70 6C 69 63 61 74 69 6F 6E 2F 78 2D 77 77  application/x-ww
77 2D 66 6F 72 6D 2D 75 72 6C 65 6E 63 6F 64 65  w-form-urlencode
64 0D 0A 0D 0A 6D 65 74 68 6F 64 3D 73 65 72 76  d....method=serv
65 72 2B 76 65 72 73 69 6F 6E 25 33 61 34 25 32  er+version%3a4%2
65 30 25 32 65 32 25 32 65 32 36 31 31 0A        e0%2e2%2e2611.

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

2.1.3 - Probability the Source Address Was Spoofed

It's unlikely that the source address was spoofed given that this is an attempt to gather information from the web server and in order to be successful, the response has to be routed back to the source host. It is entirely possible that an attacker could source route the packet; however, in these traces, the IP length is 20 bytes which means that it doesn't have any IP options. Thus, no source routing was performed.

2.1.4 - Description of Attack

Microsoft IIS ships with FrontPage Server Extensions (FPSE) which gives the administrator the capability of remotely uploading web content to the IIS server. At first glance it appears that someone could be attempting an information gathering attempt on the IIS server. However, this activity could be quite normal as it is the default behavior of FPSE client to server communication. In the process of uploading new content the FrontPage client will try to retrieve the FPSE version and path information from the server (e.g. GET http://www.fakedomain.com/_vti_inf.html) and upon receiving the _vti_inf.html file the FrontPage client will upload the new content to (e.g. POST http://www.fakedomain.com/_vti_bin/shtml.exe/_vti_rpc) the server.

With a default installation of FrontPage, it's quite easy for an attacker to retrieve

this information just as the FrontPage client does. The attacker would simply enter the URL into a browser and if no security mechanisms are in place, the attacker would have attained valuable information including the version of the FPSE and path information as to where they're located. Knowing this information allows an attacker to focus his/her efforts in the next phase of attack, attempting to exploit specific vulnerabilities (i.e. worms, buffer overflows) in old and unpatched versions of FrontPage.

2.1.5 - Attack Mechanism
In a browser or within code the attacker crafts a URL that requests the _vti_inf.html file which is typically located in c:\inetpub\wwwroot\ on the target host. The URL would look something like this:

GET http://www.fakedomain.com/_vti_inf.html

This file contains the version of the FrontPage extensions and the paths to which they're located.

2.1.6 - Correlations
http://www.securityfocus.com/bid/2144/info/[2]

2.1.7 - Evidence of Active Targeting:
It does appear that this web server is being actively targeted as no other target hosts were sent these packets (according to the log data). Moreover, four different source hosts (67.96.81.242, 64.174.39.98, 202.18.172.35, and 218.17.203.54) from different subnets attempted to access _vti_inf.html and the _vti_rpc binary. It is entirely possible that these were legitimate administrators trying to publish web content from remote locations. However, if this was legitimate FrontPage client/server communication you would typically expect to see a response from the server before the client attempted to publish data. The following is a chronology of normal FrontPage client server communication:

1. FrontPage client requests (GET) http://www.fakedomain.com/_vti_inf.html
2. FrontPage server responds with the contents of the _vti_inf.html
3. FrontPage client uploads (POST) to http://www.fakedomain.com/_vti_bin/shtml.exe/_vti_rpc

---

The following 4 lines were amended after incidents.org mailing list posting to make my point clear that:
I searched the log files for using this command (please see posting discussion below):

```
> > > # tcpdump -r 2002.9.9 src host 32.245.166.119
> > > #
```

As you can see, 32.245.166.119 isn't the src host in ANY of the logs, thus it's likely that the suspected target host doesn't exist.

The log files seem to be inconsistent as we are missing step 2.

All we see are the two requests (step 1 and 3). This doesn't fit the pattern of a legitimate connection nor does it completely fit the profile of a well known attack. Bugtraq 2144 (http://www.securityfocus.com/bid/2144/info/[2]) does note a failure to handle exceptional conditions (DoS attack vulnerable), however there are no known exploits (at the time of writing). Searching for _vti_rpc exploits with Google and the like leads me to the same conclusion. For the record, there are a plethora of FPSE vulnerabilities but none can be found referencing _vti_rpc specifically. Perhaps the attacker wrote an exploit; however, you would expect to see a lot more traffic than this. A more accurate analysis could be made with information regarding the topology of the network and more complete logs.

2.1.8 - Severity
Severity=2
The formula to derive this number: (Criticality + Lethality) – (System Countermeasures + Network Countermeasures).

Criticality=2
Based on the logs the targeted host could be a web server and as such would be fall under the critical server's category. However, if this were a vital web server you would expect to see legitimate traffic to port 80 and there isn't any indication of that in the logs, maybe the server doesn't even exist. Given that the logs do not show a lot of traffic at all to the target host I can't rank it highly and therefore give it a 2.

Lethality=3
If an intended attack succeeded against the _vti_inf.html, the damage would be moderate as the attacker would have simply gained knowledge of the FrontPage server extensions. If an intended attack succeeded against the _vti_rpc, it would mean that a new exploit was introduced and possible consequences could be root compromise, or a DoS attack although these have not been defined on Bugtraq or other various security websites. Without knowing more about the target host, it's difficult to ascertain the vulnerability of the target host. However, if in a worst-case-scenario these potential exploits could lead to full-compromise of a critical web server. I have to give it a 3.

System Countermeasures=3
According to the logs it doesn't appear as if the target host responded to this attempt to retrieve the _vti_inf.html file. Therefore, I would venture to guess that the target host either isn't running a web-server, isn't vulnerable, doesn't exist, or that the logs aren't complete.

Network Countermeasures=1
It is possible that a firewall with application intelligence could filter this particular traffic signature; however, this doesn't appear to be the case because the incoming _vti_inf.html GET request to the target host wasn't filtered before reaching what is assumed to be an internal IDS.

2.1.9 Defensive Recommendations

Unfortunately, a traditional stateful firewall wouldn't do much good at blocking this type of traffic because you typically can not block incoming connections to port 80 on your public web server for obvious reasons. The best defense is to simply uninstall the FrontPage server extensions. Moving new content to the server via sneaker net (i.e. a Burned CD) or other means would be the most secure method of moving new content to the IIS server. A second and more realistic defense is to keep the server patched with the latest IIS and FPSE patches located at Microsoft's website. A third possible defense is to implement an application intelligent Firewall (i.e. CheckPoint NG with Application Intelligence) which is capable of signature based content filtering (it has its limitations) at the application layer. Here are links to the patches.

Microsoft IIS 4.0:
Microsoft Patch Q280322
http://download.microsoft.com/download/winntsrv40/Patch/q280322/NT4/EN-US/Q280322i.EXE[3]

Microsoft IIS 5.0:
Microsoft Patch Q280322
http://download.microsoft.com/download/win2000platform/Patch/q280322/NT5/EN-US/Q280322_W2K_SP2_x86_en.EXE[3]

2.1.10 - Multiple Choice Test Question

The _vti_inf.html file contains:

a. The version of the FrontPage extensions
b. The path to the FrontPage extensions
c. The actual web content
d. a and b

2.1.11 – Posting and Feedback:

My Detect Analysis was posted to intrusions@incidents.org on Dec. 4th, 2003
Here's a response to my post:

```
Hello Brian,
Indeed, it has been very helpful.  I realize that I could have
clarified my conclusions a little bit better and will be adding an
addendum to the analysis with some of the points we've discussed.  I
appreciate your feedback and your time.  It's always good to receive
feedback from your peers.  Take care.
Kind Regards,
Ryan Rathe

Thank you for the discussion. I believe that you have clarified your
response
very adequately.
I hope that this has been helpful to you.
Good luck with you submission, I am sure you will do well.
Thank You,
Brian A Kee

On Thursday 04 December 2003 02:12 pm, you wrote:
```

> Hello Brian,
> Please see my comments inline.
> Kind Regards,
> Ryan
>
> From: Brian A Kee <bkee@lurhq.com>
>
> >To: "Ryan Rathe" <ryanrathe777@hotmail.com>
> >Subject: Re: LOGS: GIAC GCIA Version 3.4 Practical Detect Ryan Rathe
> >Date: Thu, 4 Dec 2003 12:24:03 -0600
> >
Brian: It does not look like you are mistaken. I am curious about a few things though.
> >
Brian: 1. The packets have the Ack and Push flags set. This usually indicates that an 3-way handshake has been completed. In which case we should see a "normal" response from the server.
>
Ryan: I know this. But the point is that there is no normal response, no response at all. That's the main point I was trying to make, that this doesn't fit the pattern of normal nor malicious intent. If anything this potential FPSE server simply doesn't exist (hence no traffic from 32.245.166.119). Maybe I should be more persistent about that point in my analysis.
>
Brian: 2. If they were crafted packets, then we should expect to see a TCP Reset from the server to the client.
>
Ryan: Once again, I realize this, the point is that there is NO traffic from 32.245.166.119, no SYN/ACKS, no RESETS. When I said no response in my analysis I should have been clearer. I will be sure to revise the wording.
>
Brian: 3. Another option, would really be that the packets were filtered before they got to the specified host. In this case, we may not see any return traffic.
>
Ryan: I'm not sure I understand your 3rd point. But I have considered many of the filtering possibilities. Let me explain my theories on why this is unprobable. And it does relate to the 3way handshake, this is a point I could add to my analysis, but then it all gets back to the point that there is no response from 32.245.166.119. First of all these are ACK/PUSH packets, the 3rd step in the 3-way handshake. It is possible that the SYN from the remote host got filtered, but if that happened no SYN/ACK would be sent from 32.244.166.199 (and it wasn't) and no ACK/PSH would be sent by the remote host. How would step 3 occur without 1 and 2? It could be some sort of an ACK scan (wanting a RESET) but what good would that do the attacker? also only one host was targeted (mentioned in analysis).
>
Brian: One point I am trying to make, is that there are some things that you did not address in the analysis. Although, we may never know really what happened without more information from the environment, I think that it is prudent to discuss (or at least comment on) the options in these cases.
> >

Brian: Another point is that the analysis did not cover what we might expect to
> >see
Brian: if these were "real" attacks/attempts against a vulnerable system. This information could be useful to you in the future if you see these attempts again. To state this in a questions:
Brian: How can you expect to understand this as an attack if you do not know the posible response patterns?
>
Ryan: Oh but the analysis did cover what one might expect to see in a "real" attack/attempt against a vulnerable system. I clearly stated that this it is a simple recon where the attacker does a GET on the _vti_inf file and in step 2. as outlined in the FPSE client/server communication example the server send the client the content of the _vti_inf file, hence the version and path information to the front page extensions. There are a plethora of attacks that can be implemented with that information. This is not about those attacks, this is about a simple recon attack. I do understand this attack and the response patterns and what I stated..."This doesn't fit the pattern of a legitimate connection nor does it completely fit the profile
> of a well known attack." says it all.
>
> >or
> >
Brian: If you saw this traffic pattern again, how would you determine if it was successful?
>
Ryan: If I saw this traffic pattern again I would most likely have more complete logs and/or topology information. But If I saw this exact same information I would conclude the same things.
> >--
> >Thank You,
> >
> >Brian A. Kee
> >
> >On Wednesday 03 December 2003 09:39 pm, you wrote:
> > > Brian,
Ryan: Thanks for the feedback. But if I'm not mistaken (and maybe I am) the potential web server 32.245.166.119 didn't respond as there are no logs in 2002.9.9 with 32.245.166.119 as a source, none at all (strange). I used tcpdump to check this. Here following is the output of both src and dst filters: Thanks again for your response.
> > > Kind Regards,
> > > Ryan
> > >
> > > # tcpdump -r 2002.9.9 src host 32.245.166.119
> > > #
> > > # tcpdump -r 2002.9.9 dst host 32.245.166.119
> > > 05:49:08.606507 203.135.23.100.44240 > 32.245.166.119.http: P
> > > 1233193047:1233194099(1052) ack 3637481141 win 31856 [tos
> > > 0x10]05:51:26.926507 p508639DC.dip.t-dialin.net.63350 >
> > > 32.245.166.119.http: P 34507112:34507950(838) ack 1383143189 win 8472
> > > (DF)07:07:31.926507 67.96.81.242.1135 > 32.245.166.119.http: P
> > > 3640356473:3640356737(264) ack 1914323206 win 8760 (DF)
> > > 07:07:33.126507 67.96.81.242.1135 > 32.245.166.119.http: P 264:662(398)ack

```
> >
> > > 4195 win 7486 (DF)
> > > 07:10:31.976507 67.96.81.242.1558 > 32.245.166.119.http: P
> > > 3717707560:3717707824(264) ack 2101835252 win 8760 (DF)
> > > 07:10:33.306507 67.96.81.242.1558 > 32.245.166.119.http: P
264:662(398)ack
> >
> > > 4195 win 7486 (DF)
> > > 08:17:26.256507 203.252.131.140.1569 > 32.245.166.119.http: P
> > > 2053849798:2053851262(1464) ack 468932805 win 32120 [tos 0x10]
> > > 08:48:45.936507 142.176.39.74.1653 > 32.245.166.119.http: P
> > > 170831026:170831282(256) ack 4038834395 win 64860 (DF)
> > > 10:35:28.226507 64.174.39.98.1492 > 32.245.166.119.http: P
> > > 314727112:314727377(265) ack 2210488335 win 17520 (DF)
> > > 10:35:36.686507 64.174.39.98.1502 > 32.245.166.119.http: P
> > > 318198397:318198787(390) ack 2223411797 win 17520 (DF)
> > > 10:38:41.806507 64.174.39.98.1824 > 32.245.166.119.http: P
> > > 388172545:388172574(29) ack 2412034670 win 17520 (DF)
> > > 10:38:45.706507 64.174.39.98.1826 > 32.245.166.119.http: P
> > > 389211982:389212026(44) ack 2419923617 win 17520 (DF)
> > > 10:39:51.396507 64.174.39.98.1900 > 32.245.166.119.http: P
> > > 412812387:412812652(265) ack 2475628861 win 17520 (DF)
> > > 10:39:52.016507 64.174.39.98.1901 > 32.245.166.119.http: P
> > > 413014600:413014990(390) ack 2477789423 win 17520 (DF)
> > > 11:38:29.926507 200.30.148.202.4355 > 32.245.166.119.http: P
> > > 179152360:179152643(283) ack 1904849999 win 8760 (DF)
> > > 11:38:29.956507 200.30.148.202.4356 > 32.245.166.119.http: P
> > > 179152377:179152660(283) ack 1906844228 win 8760 (DF)
> > > 11:38:29.966507 200.30.148.202.4357 > 32.245.166.119.http: P
> > > 179152394:179152675(281) ack 1910808764 win 8760 (DF)
> > > 11:38:29.976507 200.30.148.202.4358 > 32.245.166.119.http: P
> > > 179152405:179152686(281) ack 1898948717 win 8760 (DF)
> > > 13:07:30.326507 64.158.230.19.3740 > 32.245.166.119.http: P
> > > 2722792405:2722792718(313) ack 3250247911 win 8760 (DF)
> > > 15:26:46.656507 202.18.172.35.25755 > 32.245.166.119.http: P
> > > 3967931710:3967932002(292) ack 3479121640 win 8760
> > > 15:26:47.776507 202.18.172.35.25769 > 32.245.166.119.http: P
> > > 3969246161:3969246552(391) ack 3480366352 win 8760
> > > 16:20:02.546507 218.17.203.54.38225 > 32.245.166.119.http: P
> > > 4198391085:4198391343(258) ack 2553332465 win 16791 (DF)
> > > 16:20:03.386507 218.17.203.54.38226 > 32.245.166.119.http: P
> > > 4198882757:4198883140(383) ack 2565784972 win 17424 (DF)
> > > 17:58:31.156507 24.92.90.200.3324 > 32.245.166.119.http: P
> > > 230457770:230457859(89) ack 214922242 win 8760 (DF)
> > > 18:14:27.706507 4.60.2.72.3246 > 32.245.166.119.http: P
> > > 65641141:65641397(256) ack 1205281352 win 64240 (DF)
--
Thank You,

Brian A. Kee
Regional Secure Operations Manager
LURHQ Corporation
(630) 371-4700
```

2.2 - Detect 2, Nmap Xmas Scan

2.2.1 - Source of Trace:
This trace comes from the raw logs at
http://www.incidents.org/logs/Raw/2002.8.23[1]

2.2.2 - Detect Generated By: Snort v.1.8.6

The relevant Snort alerts follow:
[**] [100:1:1] spp_portscan: PORTSCAN DETECTED to port 601 from 115.74.249.65 (STEALTH) [**]
11/17-22:41:44.599940

[**] [1:1228:1] SCAN NMAP XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/23-14:38:46.316507 115.74.249.65:61621 -> 198.61.16.19:601
TCP TTL:50 TOS:0x0 ID:55961 IpLen:20 DgmLen:60
U*PF Seq: 0x417A1598 Ack: 0x0 Win: 0x800 TcpLen: 40 UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
[Xref => http://www.whitehats.com/info/IDS30]

[**] [1:1228:1] SCAN NMAP XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/23-14:38:49.836507 115.74.249.65:61621 -> 198.61.16.19:601
TCP TTL:50 TOS:0x0 ID:31462 IpLen:20 DgmLen:60
U*PF Seq: 0x417A1598 Ack: 0x0 Win: 0x800 TcpLen: 40 UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
[Xref => http://www.whitehats.com/info/IDS30]

[**] [100:2:1] spp_portscan: portscan status from 115.74.249.65: 1 connections across 1 hosts: TCP(1), UDP(0) STEALTH [**]
11/17-22:41:44.601354

[**] [1:1228:1] SCAN NMAP XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/23-14:38:54.356507 115.74.249.65:61621 -> 198.61.16.19:601
TCP TTL:50 TOS:0x0 ID:6909 IpLen:20 DgmLen:60
U*PF Seq: 0x4AC38CDD Ack: 0x0 Win: 0x800 TcpLen: 40 UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
[Xref => http://www.whitehats.com/info/IDS30]

[**] [1:1228:1] SCAN NMAP XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/23-14:38:56.776507 115.74.249.65:61621 -> 198.61.16.19:601
TCP TTL:50 TOS:0x0 ID:65279 IpLen:20 DgmLen:60
U*PF Seq: 0xD53E5D5B Ack: 0x0 Win: 0x800 TcpLen: 40 UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
[Xref => http://www.whitehats.com/info/IDS30]

[**] [100:2:1] spp_portscan: portscan status from 115.74.249.65: 1 connections across 1 hosts: TCP(1), UDP(0) STEALTH [**]
11/17-22:41:44.602546

[**] [1:1228:1] SCAN NMAP XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]

```
09/23-14:39:00.136507 115.74.249.65:61621 -> 198.61.16.19:601
TCP TTL:50 TOS:0x0 ID:27625 IpLen:20 DgmLen:60
**U*P**F Seq: 0xD53E5D5B  Ack: 0x0  Win: 0x800  TcpLen: 40  UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
[Xref => http://www.whitehats.com/info/IDS30]
```

For the purpose of brevity only one detailed log was included. Note- all of the source hosts that generated these alerts have logs entries similar to this:

```
[**] SCAN NMAP XMAS [**]
09/23-14:38:46.316507 115.74.249.65:61621 -> 198.61.16.19:601
TCP TTL:50 TOS:0x0 ID:55961 IpLen:20 DgmLen:60
**U*P**F Seq: 0x417A1598  Ack: 0x0  Win: 0x800  TcpLen: 40  UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL

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

# tcpdump -r 2002.8.23 src 198.61.16.19 and dst 115.74.249.65
```

Note- This was done to verify whether or not any reset packets were sent from the target host (198.61.16.19) back to the attacker (115.74.249.65).

2.2.3 - Probability the Source Address Was Spoofed

It's unlikely that the source address was spoofed given that these packets aim at soliciting a response from the target host for the purposes of OS (Operation System) fingerprinting and inversely identifying closed ports. Unless the packets were source routed, the responses would not be routed back to the attacker. Thus, the attacker would achieve nothing by spoofing the source. Lastly, the source address was not spoofed as there are no IP options (IpLen=20bytes).

2.2.4 - Description of Attack

The Xmas scan falls into the information gathering phase of an organized attack plan and functions by sending irregular packets to a target host in the hopes of receiving unique responses that can help to identify the target host's listening ports and OS platform. According to RFC specifications, a closed port should respond with a RESET while a listening port should not respond at all. Receiving RESETS in response to Xmas packets allows an attacker to inversely map listening ports. Here they're assuming that if it doesn't respond, it must be listening. In the case of OS fingerprinting the Xmas packet, typically is accompanied by other packets (i.e. NULL to open port, ACK to open Port) in an attempt to gather OS specific information about the target host(s) for the purpose of narrowing down the scope of vulnerabilities that he/she will scan for in the next phase. Knowing the OS of the target host allows the attacker to focus his/her efforts in scanning for vulnerabilities based on OS (i.e. IIS Directory Traversal Exploit on a Microsoft Server). This saves the attacker time, effort, and reduces the overall traffic sent to target host(s), thereby reducing the risk of being discovered.

2.2.5 - Attack Mechanism

The Xmas scan gets its name from the code bits resemblance to that of lights on a Christmas tree (13th TCP byte offset = |0010|1001|). Looking at the code bits in

the 13th byte offset of the TCP header you'll see that the URG, PSH, and FIN bits are set. An Xmas packet violates TCP specifications by sending packets with code bits that aren't expected at the beginning of a connection (TCP 3-way handshake) in an attempt to elicit a RESET response from the target host. Xmas packets are usually generated by a scanning tool such as Nmap and used in conjunction with other scanning techniques. Various OS TCP stacks will respond differently to these out-of-spec packets and it's those inconsistencies that help to identify the OS of the target host. It's worth noting that Microsoft 9x, NT, 2K, XP based systems are not vulnerable to this scan as their TCP stack doesn't follow RFC specifications regarding when to send RESETs.

2.2.6 - Correlations
This attack has been well published and my own knowledge and experience were used in this analysis.

2.2.7 - Evidence of Active Targeting
It appears as if the attacker is actively targeting this particular host given that all of the Xmas alerts generated reflect a destination IP of 198.61.16.19. If the attacker were performing an across the board reconnaissance effort, you would likely see Alerts being generated for the entire subnet (198.61.16.0/24) and not just 198.61.16.19.

2.2.8 - Severity
Severity=1
The formula to derive this number: (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality=2
Based on the logs the host targeted by this attack doesn't appear to provide any critical services.

Lethality=1
If the intended attack succeeded, the damage would be minimal as the attacker simply gained knowledge that port 601 was closed and possibly some insight into the target hosts operating system.

System Countermeasures=3
Either the target host was a Microsoft based OS which doesn't respond to Xmas scans or port 601 was in fact listening. Given that the target didn't respond, I give it a 3 here.

Network Countermeasures=1
A stateful firewall would have prevented these Xmas packets from reaching the target host. However, it can't be determined if a firewall was in place.

2.2.9 - Defensive Recommendations
A stateful firewall such as Cisco PIX or CheckPoint FW1 would provide an excellent defense against such an attack. A firewall would block this unsolicited traffic as it would not show an established connection in its state table and henceforth these scans would be dropped without reaching the intended target host.

2.2.10 - Multiple Choice Test Question
According to RFC specification what is the expected behavior of a target host receiving an Xmas packet on a closed port with the URG, PSH, and FIN code bits set?

a. To not respond and send nothing back to the source host.
b. To send a SYN, ACK packet
c. To send a FIN packet
d. To send a RESET packet

2.3 - Detect 3, SMB C$ Share

2.3.1 - Source of Trace:
This trace comes from the raw logs at
http://www.incidents.org/logs/Raw/2002.9.15[1]

2.3.2 - Detect Generated By: Snort v.1.8.6

Rule that Generated the Alerts:
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 relevant Snort alerts follow:
[**] [1:533:1] NETBIOS SMB C access [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/14-20:18:50.406507 211.245.119.235:2095 -> 32.245.166.132:139
TCP TTL:109 TOS:0x0 ID:556 IpLen:20 DgmLen:99 DF
AP Seq: 0x4492E798 Ack: 0xD2082290 Win: 0x446C TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS339]

[**] [1:533:1] NETBIOS SMB C access [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/14-20:37:27.756507 211.99.60.135:2111 -> 32.245.166.132:139
TCP TTL:107 TOS:0x0 ID:316 IpLen:20 DgmLen:99 DF
AP Seq: 0xFFC917 Ack: 0xE293CAE7 Win: 0x221C TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS339]

[**] [1:533:1] NETBIOS SMB C access [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/14-20:49:39.696507 218.155.89.152:1041 -> 32.245.166.132:139
TCP TTL:109 TOS:0x0 ID:678 IpLen:20 DgmLen:99 DF

```
***AP*** Seq: 0x9F934CCD  Ack: 0xED6EEE34  Win: 0x431E  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS339]

[**] [1:533:1] NETBIOS SMB C access [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/15-04:38:03.826507 200.155.66.172:1143 -> 32.245.166.132:139
TCP TTL:109 TOS:0x0 ID:25531 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x7E1C10  Ack: 0x8DC8C162  Win: 0x217C  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS339]
```

For the purpose of brevity only one detailed log was included. Note- all of the source hosts that generated these alerts have logs entries similar to this:

```
 [**] NETBIOS SMB C access [**]
10/15-04:38:03.826507 200.155.66.172:1143 -> 32.245.166.132:139
TCP TTL:109 TOS:0x0 ID:25531 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x7E1C10  Ack: 0x8DC8C162  Win: 0x217C  TcpLen: 20
00 00 00 37 FF 53 4D 42 75 00 00 00 00 00 00 00  ...7.SMBu.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 04 FF 00 00 00 00 00 01 00 0C 00 21  ...............!
5C 5C 42 32 42 5C 43 00 41 3A 00                 \\B2B\C.A:.

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

2.3.3 - Probability the Source Address Was Spoofed
It is unlikely that an attacker would spoof the source address due to the nature of this attack. The attacker is attempting to gain SMB access to the default C$ administrative share on a Windows based system. If the source address were spoofed, the packet would have to be source routed in order for them to receive the return traffic. Also, the IP Length is 20 bytes which doesn't indicate any IP options such as source routing.

2.3.4 - Description of Attack
By default Windows systems have default administrative shares using the format %DRIVE_LETTER% + $. The $ allows the share to be hidden from network browsing, albeit this doesn't do much good as the default C$ is commonly known. For example, an attacker simply has to open up a browser and in the address bar type \\1.2.3.4\C$ where 1.2.3.4 represents the IP address. From there the attacker will be prompted for a password and in many cases the password is either extremely easy to guess or there is no password. With administrative access to the entire C drive an attacker for the most part "owns" the machine as they could copy over trojans, root kits or do whatever they like.

2.3.5 - Attack Mechanism
The attacker simply needs a system with a samba (SMB) client and a network connection. This attack can be performed from the internal network as well as from the outside depending on the presence of a firewall or filtering router and its ruleset (block external traffic to internal port 239).

2.3.6 - Correlations
This attack has been well published and my own knowledge and experience

were used in this analysis.

2.3.7 - Evidence of Active Targeting
If indeed this isn't a false positive it does appear that the target host has been actively targeted by four different source hosts (211.245.119.235, 211.99.60.135, 218.155.89.152, and 200.155.66.172). No other target hosts received external traffic destined for port 139.

2.3.8 - Severity
Severity=2
The formula to derive this number: (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality:=2
Based on the logs the host targeted by this attack doesn't appear to provide an critical services Sorting through the logs it appears that these four traces were the only communication to or from this target host.

Lethality=4
If the attack were successful it would give the attacker administrative access to the entire C drive of the target host. This is extremely dangerous as the attacker could then take additional steps to control the system (i.e. trojan, back doors, and sniffers). With complete control of this machine the attacker could sniff the network segment and launch further attacks against other systems thereby expanding his/her influence into the network.

System Countermeasures=3
Given that the target host (32.245.166.132) didn't respond in any fashion, I would venture to say that this was either an ill led attack or a false positive.

Network Countermeasures=1
A firewall or Filtering Router that blocks external hosts from attempting to communicate with internal hosts at port 139 would prevent this type of attack/communication from taking place. Being that traffic showed up on what is assumed to be an internal IDS It would appear that a Firewall or Filtering Router wasn't in place.

2.3.9 - Defensive Recommendations
Two things can be done to easily prevent this type of attack. C$ administrative shares can be disabled. However, this will require a registry edit as Windows machines will automatically recreate the C$ share upon reboot. For NT.40,W2K, and W2K3 Server the you would invoke regedit and change this value as shown:

Hive: HKEY_LOCAL_MACHINE
Key: SYSTEM\CurrentControlSet\Services|LanManServer\Paramters
Name: AutoShareServer (Note: for workstations- Name: AutoShareWks)

Data Type: REG_DWORD
Value: 0

Secondly, the best way to prevent C$ access is to have a Firewall or Filtering Router that blocks all external network traffic destined for internal systems at port 139. It can be extremely dangerous to have Samba shares open to the outside world. If samba share access is needed by remote offices, then something such as a VPN or dedicated connection should be considered.

2.3.10 - Multiple Choice Test Question
Which of the default administrative shares listed could potentially give instant Samba (SMB) access to an entire drive on Windows systems?

a. FAX$
b. IPC$
c. D$
d. PRINT$

2.4 References

Incidents.org Raw Log Files Analysis:

[1]Incidents.org
URL: http://www/incidents.org/logs/Raw (Dec 6, 2003)

Detect 1, IIS _vti_inf.html and _vti_rpc Access:

[2]SecurityFocus. Bugtraq Vulnerability Database
URL: www.securityfocus.com (Dec 6, 2003)

[3]Microsoft Corp. (2003)
URLs:http://download.microsoft.com/download/winntsrv40/Patch/q280322/NT4/EN-US/Q280322i.EXE (Dec 6, 2003)
http://download.microsoft.com/download/win2000platform/Patch/q280322/NT5/EN-US/Q280322_W2K_SP2_x86_en.EXE (Dec 6, 2003)

3.0 - Analyze This

3.1.1 - Introduction
This section is an analysis of five days of Snort generated alert files from “The University network”. The Snort system in place was running in IDS mode and collected alerts in fast mode. Scan and OOS (Out-of-Spec) alerts were also collected and used in the analysis process. Unfortunately, no logging of any kind was in place and subsequently many of the conclusions made in this document are based on certain assumptions about the network topology. In this document assumptions are made as to normal vs. malicious activity based on knowledge of TCP/UDP ports commonly used by Trojans (and the like) and what is largely considered to be ‘normal’ activity. Many of these alerts could very well be false

positives. An inventory and assessment of the University's applications and processes (i.e. custom database application that uses out of the ordinary port numbers) and their associated port numbers would help in checking the validity of the Alerts.

First, we'll discuss the scan files, activities that typically indicate that an external user may be attempting to perform some reconnaissance (information gathering) work on the University's internal systems. I'll explain what is actually happening, cite examples, break down the majority of activity and give possible explanations based on what we see going on in the scan files. Secondly, well discuss the out-of-spec (OOS) files. Out-of-spec files are malformed packets that are in violation of RFC specifications. These packets are typically sent in an information-gathering attempt to discover live hosts and ports, and to identify the OS platform that is target host is running. I'll give a brief description of each type of OOS packet that we received in the five-day period. Lastly, we'll discuss the alerts, which are the most useful of the three alert types as they generally are more specific to the actual event that is taking place and provide the most insight. We'll start with a high level overview with a lot of charts and graphs. Then we'll discuss the most frequent alerts, the most active hosts, and the most dangerous ports seen within the alerts. Next, we'll go over the most severe internally generated alerts. This isn't meant to shadow the severity of the externally generated alerts, which are also important; however, alerts generated by an internal machine typically are more serious as it commonly means that the internal machine has already been compromised and can spread the infection internally at a much greater rate than machines on the outside. Finally, we'll finish up with some general defensive recommendations and details about the interesting hosts that were party to some of the more severe alerts.

3.1.2 - Executive Summary

The University has security issues of epidemic proportions. The University network is in dire need of an in-depth vulnerability assessment, systems audit, security policies, and security systems deployment. One of the biggest problems is that of users running P2P and gaming applications. This consumes network bandwidth and is a liability for the University. Also, it's evident that several viruses, worms, and Trojan horses have manifested themselves upon the internal network and are trying to propagate further. Much work needs to be done to clean up the infections, shore up the vulnerabilities, and prevent misuse of network resources. All systems will require a full audit and vulnerability assessment. Virus Scanners will need to be installed on all systems, configured appropriately and maintained to receive daily updates. Stateful Firewalls will need to be deployed with policies that block all but what is necessary. IDS systems will also need some re-evaluation as they should be upgraded to the most current stable release with current rules. Also, more in depth logging will be required and a sensor on each network segment will also be required. A computer usage policy will need to be enforced limiting users use of P2P software and games. Lastly, network baselining, integrity checkers, Snort binary

logging, and other security tools should be utilized so that the University is better prepared to handle future security incidents.

3.1.3 - List of Analyzed Files from HTTP://www.incidents.org/logs/

Log Date	Alerts	Out-Of-Spec	Scans
01/06/02	alert.030106	OOS_Report_2003_01_07_31845	scans.030106
01/07/02	alert.030107	OOS_Report_2003_01_08_8856	scans.030107
01/08/02	alert.030108	OOS_Report_2003_01_09_12713	scans.030108
01/09/02	alert.030109	OOS_Report_2003_01_10_4480	scans.030109
01/10/02	alert.030110	OOS_Report_2003_01_11_4183	scans.030110

To make it easier for me to analyze these files, I consolidated all of the alerts, oos, and scans into one file of type (e.g. cat alert.030106 >> alerts.csv). I then converted the files to a comma-separated format (.csv) so they could be loaded into a mysql database. Many thanks to Todd Beardsley, GCIA for his published csv.pl script[1]. After creating the csv files, I proceeded to load them into a mysql database using the following command:

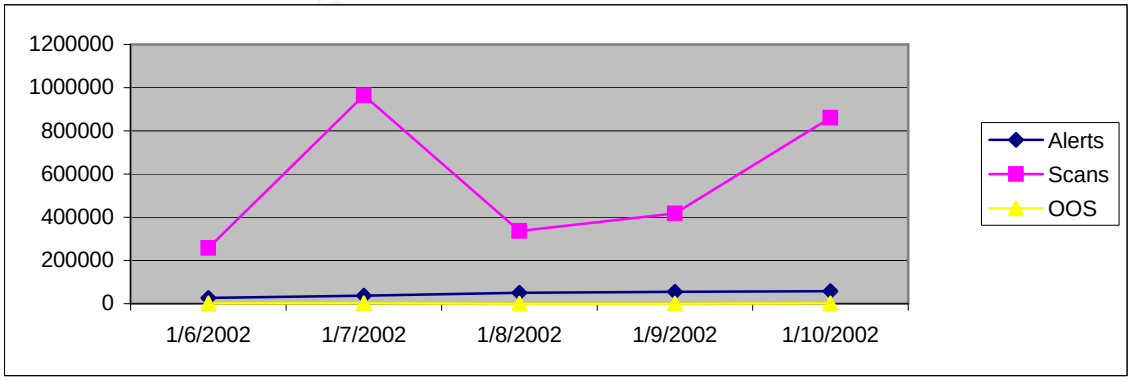
```
mysql> LOAD DATA LOCAL INFILE '/root/practical/alerts.csv' INTO TABLE analysis FIELDS
TERMINATED BY ',' LINES TERMINATED BY '/n';
```

3.2 - Scans Analysis

3.2.1 - Summary

There were a total of 2,837,600 scan alerts. Of the 2,638,970 alerts, roughly 93% were triggered by the UDP scan signature and 6.9% were classified as SYN scans. Most of these scan alerts were triggered by P2P and gaming activities such as WinMX, Kazaa, Morpheus, and Blubster. Some scanning did take place. This was not a major event, but certainly warrants concern. The following is a breakdown of the activity.

3.2.2 - 5-day Trend of Scans, Alerts, and OOS



3.2.3 Most Frequent Scan Alerts

Scan Message	Code Bits	Total
UDP scan (Externally based)	NA	2638978
SYN scan (Externally based)	Syn Flag Set	197604
NULL scan (Externally based)	No Flags Set	391

INVALIDACK scan (Externally based)	Abnormal Ack, not SPAU or FULLXMAS	230
UNKNOWN scan (Externally based)	Abnormal Combination of Flags	182
NOACK scan (Externally based)	Ack Flag is Missing	131
VECNA scan (Externally based)	One of These: P, U, PU, FP, or FU	48
FIN scan (Externally based)	Fin Flag Set	14
FULLXMAS scan (Externally based)	SFRPAU: All Flags Set	7
XMAS scan (Externally based)	FPU Flags Set	4
NMAPID scan (Externally based)	SFPU Flags Set	4
SPAU scan (Externally based)	SPAU Flags Set	4
SYNFIN scan (Externally based)	Syn and Fin Flags Set	3
TOTAL SCANS		2,837,600

3.2.4 -Top Source Ports in Scan Alerts

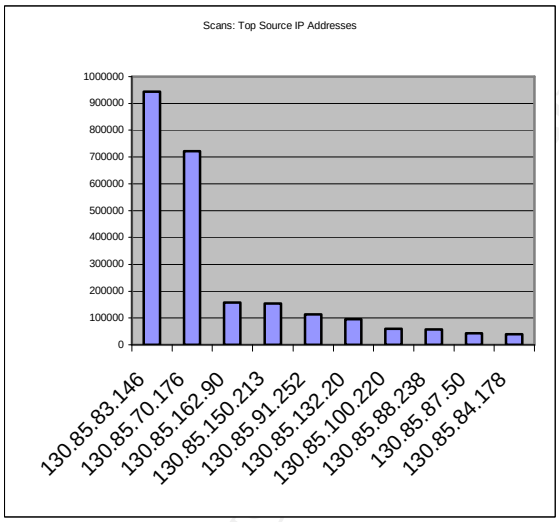
Source TCP/UDP port	Possible Port Activity	Number of Associated Scan Alerts
6257	**WinMX P2P**	1714067
1637	CableNet Admin Protocol	157104
2502	**possible W32.Blaster Worm**	141206
1237	tdos390	112157
4848	**Playlink Gaming**	92122
2095	**Webmail**	59367
1974	Data-Link Switching Remote Access Protocol	55475
999	**Deep Throat, Foreplay, WinSatan**	35534
2320	**ICQ or IMAP**	30633
3466	workflow	29846
2045	cdfunc	28414
137	**NetBIOS**	24444
2228	netml	19632
12203	**Medal of Honor Gaming**	18870
12300	**Medal of Honor Gaming**	14016
3339	Omf-data 1	12928
1846	tunstall pnc	12356
1141	fnone	12158
1066	fpo-fns	9055
2416	rmt-server	8188
888	accesbuilder, cd database protocol	7287
1037		5034
7001	**Freak88 Trojan**	2895
2600	**Digital Root Beer Trojan**	2805
1214	**Kazaa/Morpheus P2P**	2454

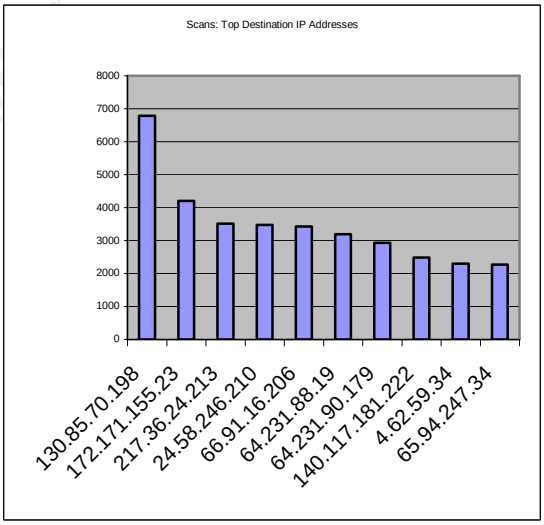
3.2.5 -Top Destination Ports in Scan Alerts

Destination TCP/UDP port	Possible Port Activity	Number of Associated Scan Alerts
6257	**WinMX P2P**	1663087
41170	**Blubster P2P**	66384
80	HTTP	48613
445	microsoft-ds	42634
137	NetBIOS-ns	38413
27005	Flex-lm	26764

1214	**KAZAA/Morpheus or Grokster**	22036
443	HTTPS	20817
135	**Epmap, Blaster**	12013
21	FTP	10639
6346	**Gnutella**	8865
1433	**Microsoft SQL server, SQL Snake**	7438
16257		6529
139	Netbios-ssn	6122
1186		4468
8888	**Napster or ddi-tcp1**	4097
6970	**QuickTime streaming video**	3843
1851	ctcd	3733
1367	dcs	3634
1320	Panja-axbnet	3062
1465	pipes-platform	2577
1024	**IRC or H.323 (NetMeeting), Jade or possible NetSpy Trojan**	2514
1327	ultrex	2477
3159	**Eclipse 2000, Sanctuary Trojan**	2470
1498	Sybase SQL AnyWhere	2455

3.2.6 - Scan Alerts: Top Talkers



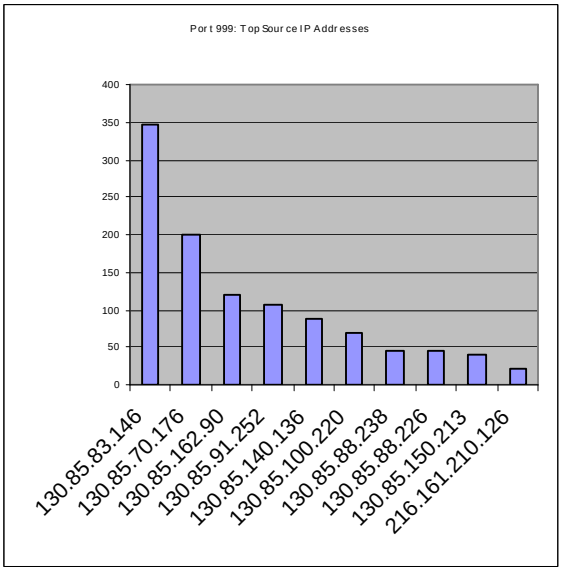


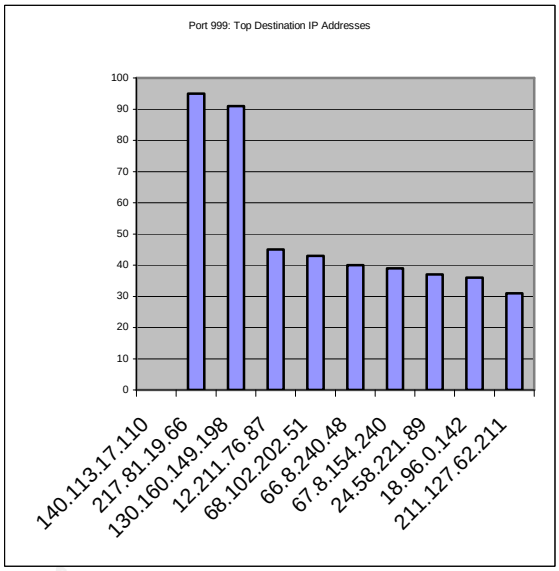
3.2.7 - Breakdown of Suspected Scan Activity

3.2.7.1 - Possible Trojans and Worms

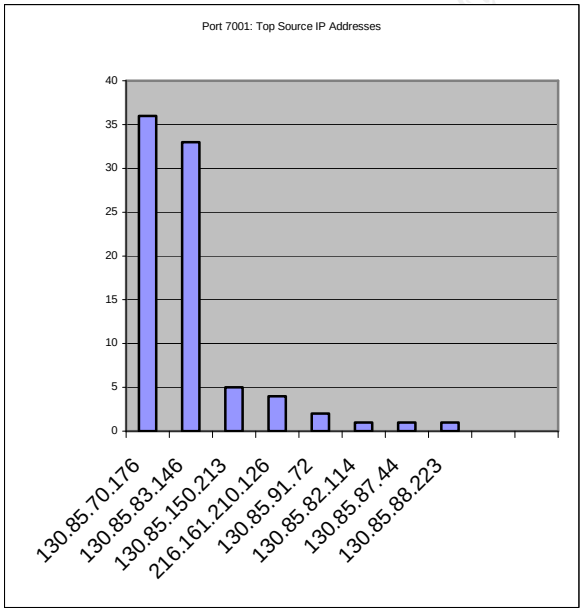
Several port numbers found in the scans are commonly associated with Trojan horses and worms. It is entirely possible that many of these alerts are false positives and are subject to the possibility that the random ephemeral port just happened to be that of the Trojan. However some ports listed below don't fall under the ephemeral range and could show compromise. The keyword I want to emphasize here is “possible”. Detailed logs would be required to bring about definitive conclusions. Here is a breakdown of the top 10 hosts per port number found in the top scan alerts. All hosts found in these charts should be given a thorough inspection.

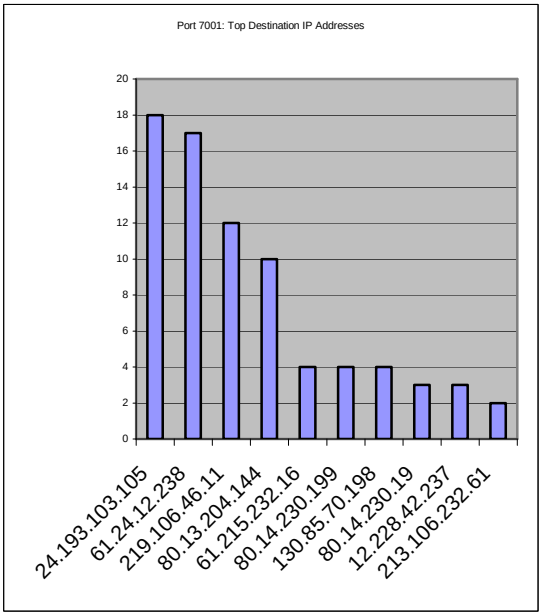
Port	Possible Activity	Occurrences
999	Deep Throat, Foreplay or WinSatan Trojan	35534



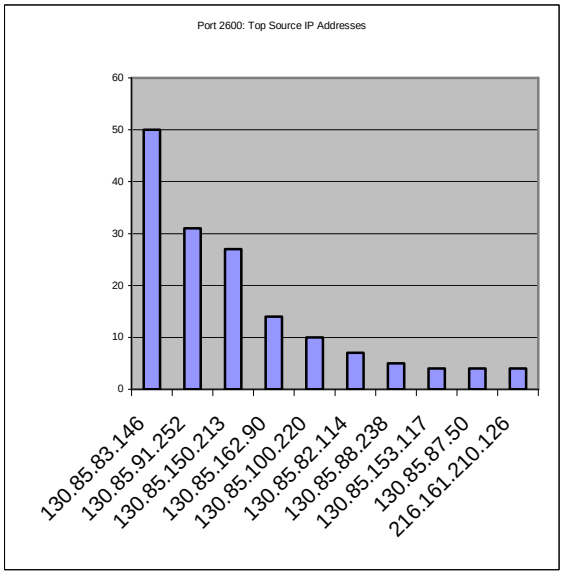


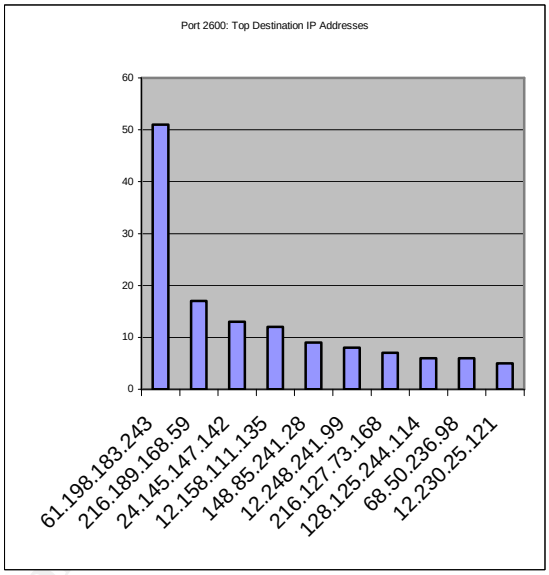
Port	Possible Activity	Occurrences
7001	Freak88 Trojan	2895



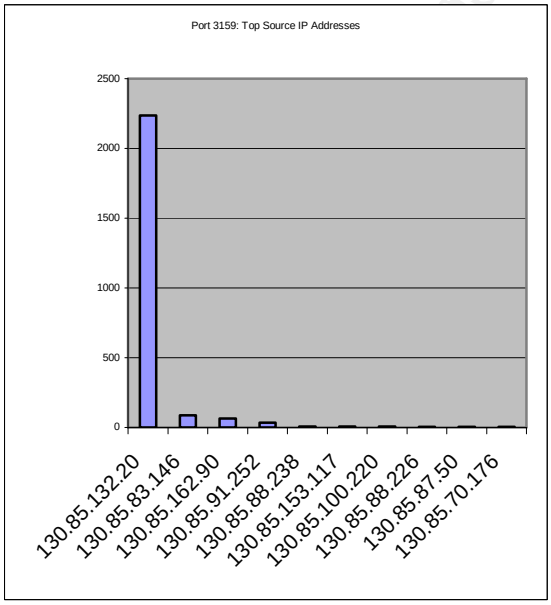


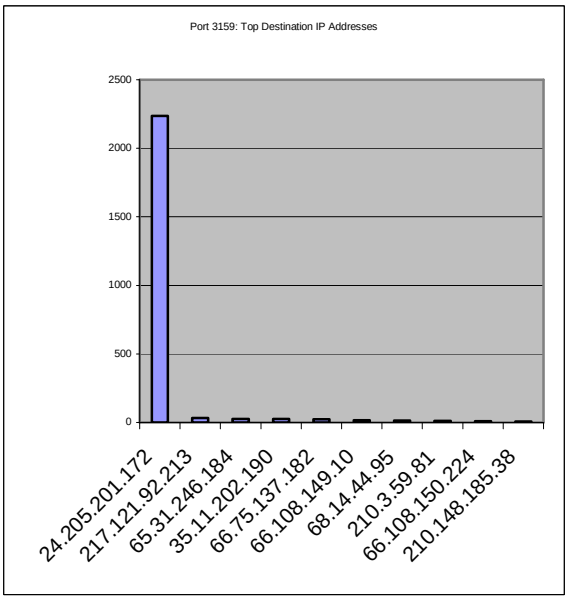
Port	Possible Activity	Occurrences
2600	Digital Root Beer Trojan	2805





Port	Possible Activity	Occurrences
3159	Eclipse 2000 or Sanctuary Trojan	2470

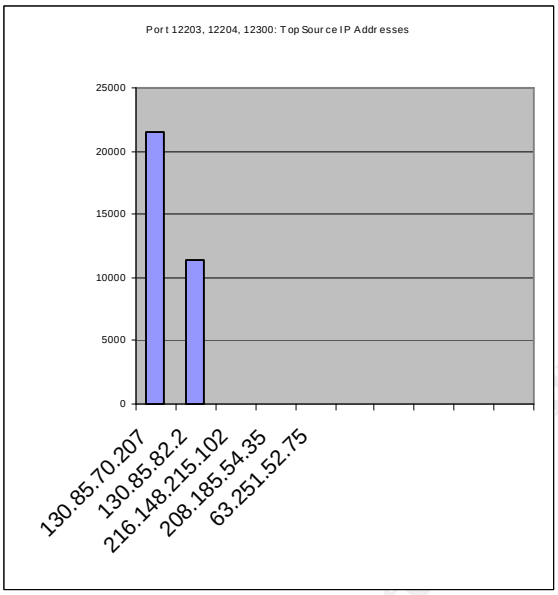


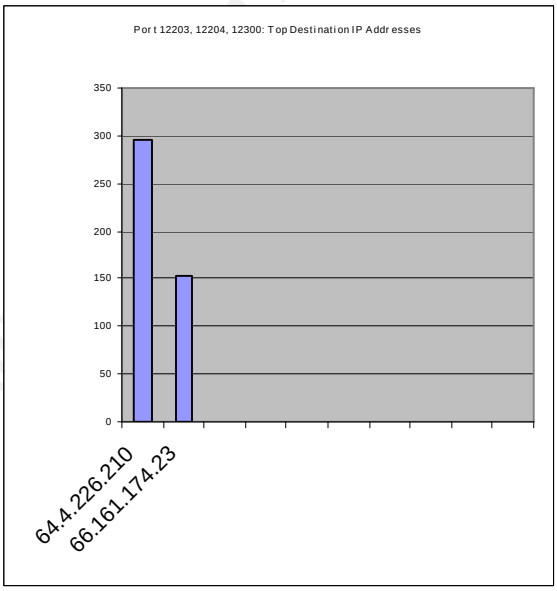


3.2.7.2 - Gaming

Gaming may not be malicious by nature but can still do plenty of damage to a network by simple right of bandwidth usage. Gaming is the 2nd most popular activity behind P2P on the University network and can likely account for just as much bandwidth utilization. Common gaming ports should be blocked at a firewall and it should be considered that PCs have a system policy that prevents users from installing applications without an administrative password.

Ports	Possible Activity	Occurrences
12203, 12204, 12300	Medal of Honor Gaming	32886

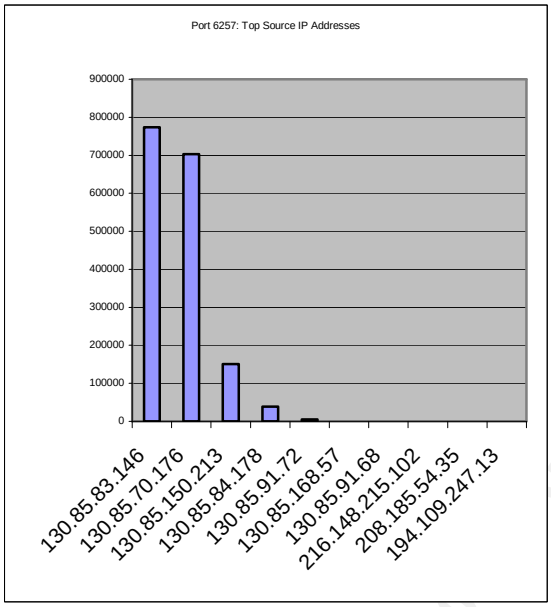


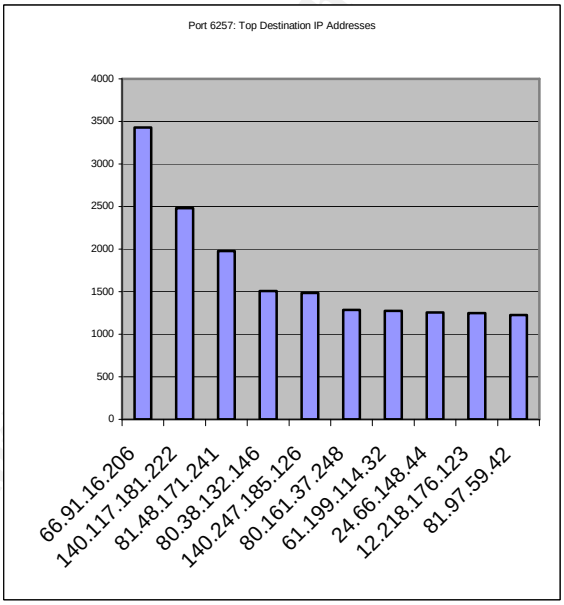


3.2.7.3 - Peer-to-Peer (P2P) File Sharing

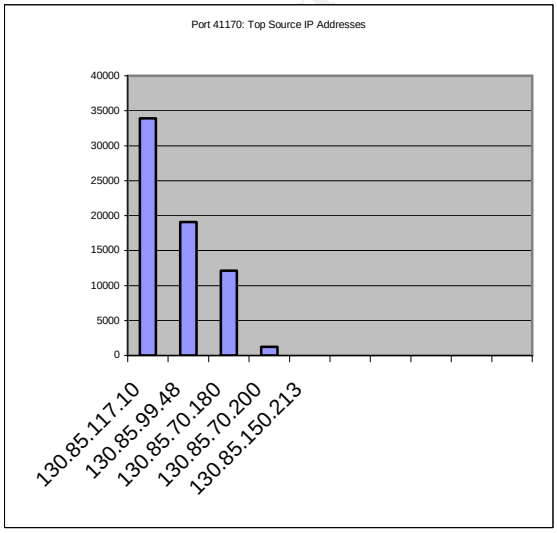
Most of the scan detects have to do with the abundance of Peer-to-Peer (P2P) file, music, and movie file sharing that is taking place on the network. This activity should be blocked at the firewall and a computer usage policy needs to be brought into force.

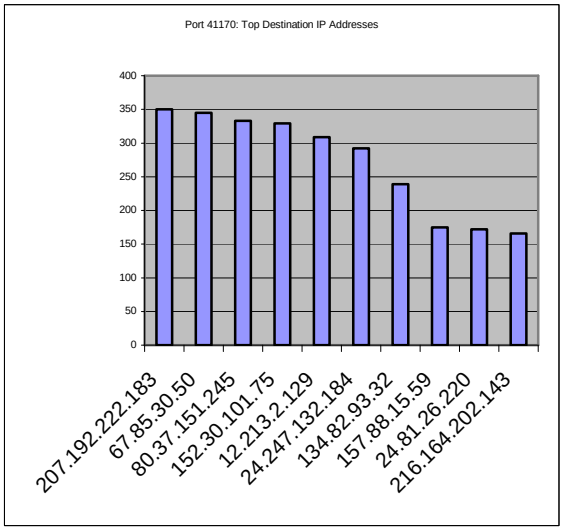
Port	Possible Activity	Occurrences
6257	WinMX P2P	1663083



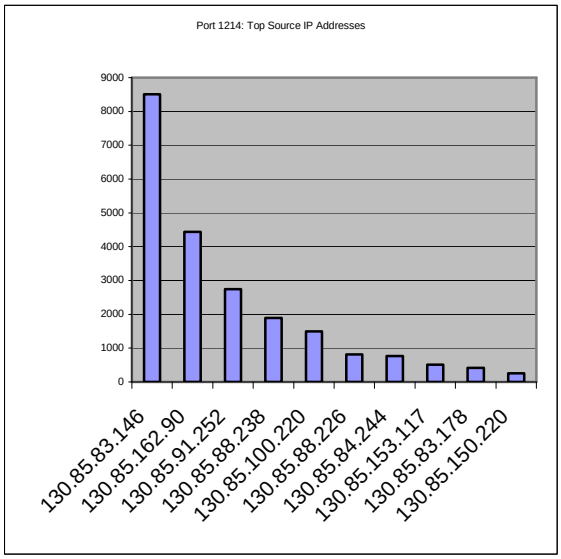


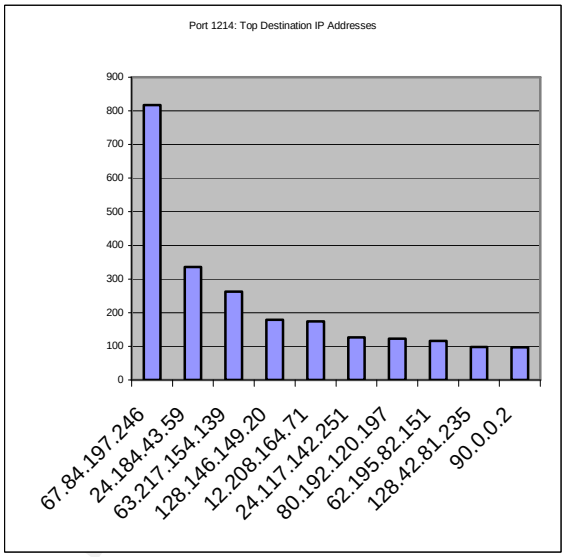
Port	Possible Activity	Occurrences
41170	Blubster P2P	66384





Port	Possible Activity	Occurrences
1214	Kazaa or Morpheus	22036





3.2.8 - Recommendations Based on Scan Analysis

All of the University hosts involved in source or destination alerts relating to the list of alerts shown above should be brought offline and inspected thoroughly. They should be scanned for viruses, Trojans, and worms, and cleaned of all unnecessary gaming and P2P applications. The lists above are brief and in actuality, every host machine within the University network will require a thorough investigation. Beyond the infections, the University should create a computer usage policy for all students and employees. Moreover, stateful firewalls should be put into place with a rule base that allows only the necessary services with an implicit “deny all to all“ at the bottom of the list. Critical system computers associated with the Trojan based ports should be brought offline immediately and inspected and cleaned thoroughly by an experienced security incident handler. For many Trojan associated systems, it may make more sense to simply reinstall as the time to reinstall may outweigh the time in cleaning such systems. After all, even with the most thorough virus and Trojan cleanup, you're still not 100% sure that something didn't linger as an attacker can leave untraceable backdoors at a kernel level that can be activated with magic packets.

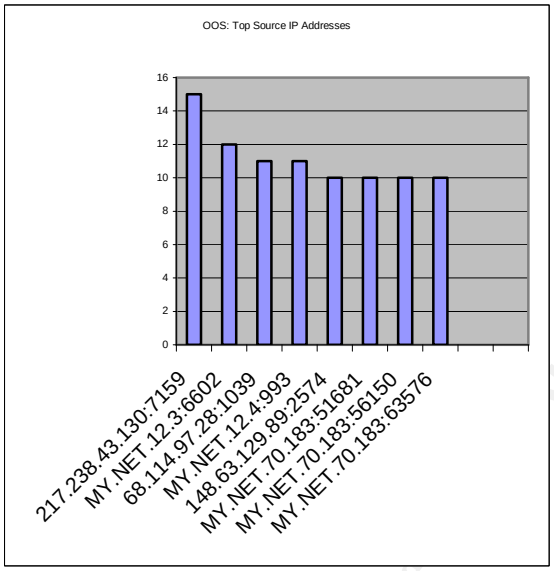
3.3 - OOS Analysis

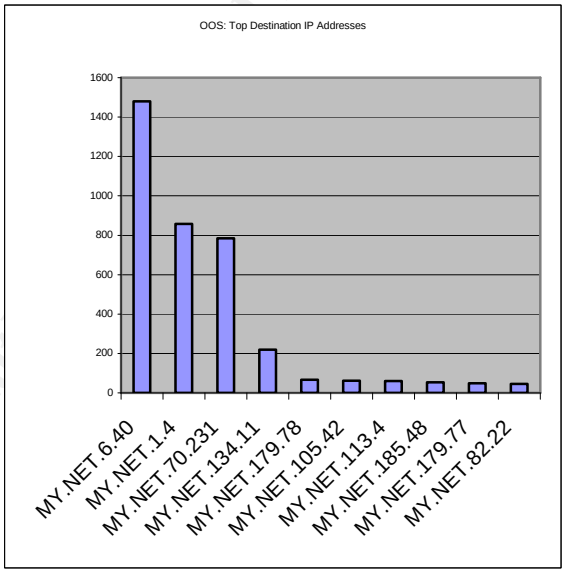
There were a total of 4,182 Out-of-Spec alerts within this 5-day period. Many of the OOS triggered alerts have to do with the ECN (Explicit Congestion Notification) bits and are valid as per RFC3168. These are probably false alerts, yet another reason to upgrade the Snort sensors on the University network. Many of these other packets are simply ‘crud’; however, some do indicate that real scanning is taking place both to and from the University network, largely the byproduct of P2P and some of the worm activity that is taking place.

3.3.1 - OOS: Flags

OOS flags	Description
12****S*	Syn Scan with reserved bits
********	No TCP header flags set
12UAPRSF	All bits set, common OS fingerprinting technique, also faulty router's can cause this
URSF	Urg, Reset, Syn, and Fin flags set
12****SF	Syn-Fin flags set
12U*PR**	Urg, Push, and Rest flags set
1**A*RSF	Ack, Reset, Syn and Fin flags set
12***R**	Reset with Reserved Bits
****P***	Push Flag Set

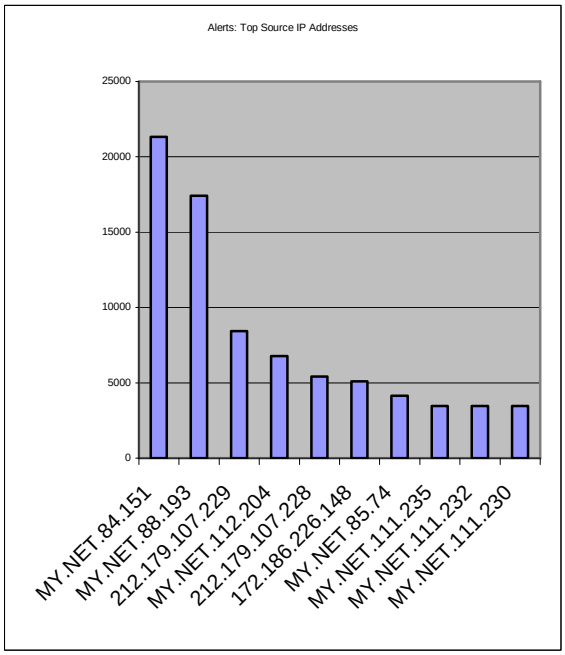
3.3.2 - OOS: Top Ten Talkers

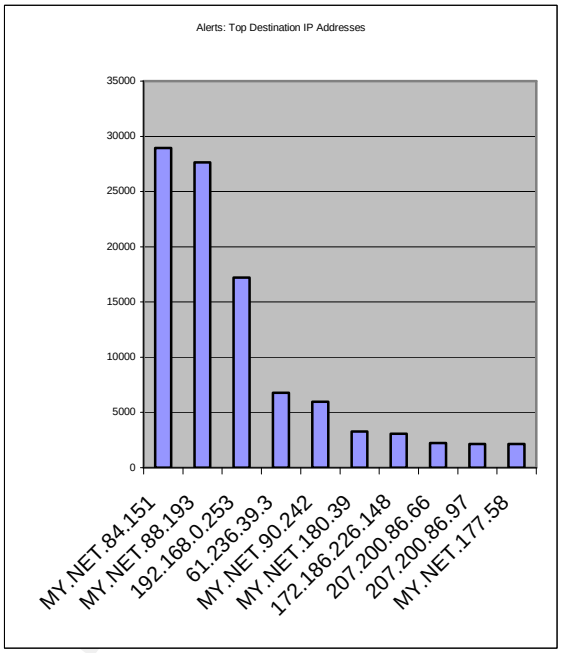




3.4 – Alerts

3.4.1 - Alerts: Top Ten Talkers





Rank	Source IP Address	Possible Activity	Occurrences
1	MY.NET.84.151	High port 65535 tcp - possible Red Worm - traffic	21336
2	MY.NET.88.193	High port 65535 tcp - possible Red Worm - traffic	17407
3	212.179.107.229	Watchlist 000220 IL-ISDNNET-990517	8427
4	MY.NET.112.204	spp_HTTP_decode: IIS Unicode attack detected	6785
5	212.179.107.228	Watchlist 000220 IL-ISDNNET-990517	5426
6	172.186.226.148	High port 65535 tcp - possible Red Worm - traffic	5102
7	MY.NET.85.74	spp_HTTP_decode: IIS Unicode attack detected	4135
8	MY.NET.111.235	TFTP - External UDP connection to internal TFTP server	3470
9	MY.NET.111.232	TFTP - External UDP connection to internal TFTP server	3465
10	MY.NET.111.230	TFTP - External UDP connection to internal TFTP server	3452

Rank	Destination IP Address	Possible Activity	Occurrences
1	MY.NET.84.151	High port 65535 tcp - possible Red Worm - traffic	28959
2	MY.NET.88.193	High port 65535 tcp - possible Red Worm - traffic	27635
3	192.168.0.253	TFTP - External UDP connection to internal TFTP server	17216

4	61.236.39.3	spp_HTTP_decode: IIS Unicode attack detected	6785
5	MY.NET.90.242	Watchlist 000220 IL-ISDNNET-990517	5968
6	MY.NET.180.39	Watchlist 000220 IL-ISDNNET-990517 EXPLOIT x86 NOOP	3286 1
7	172.186.226.148	High port 65535 tcp - possible Red Worm - traffic	3078
8	207.200.86.66	spp_HTTP_decode: IIS Unicode attack detected	2227
9	207.200.86.97	spp_HTTP_decode: IIS Unicode attack detected	2154
10	MY.NET.177.58	Watchlist 000220 IL-ISDNNET-990517	2134

3.4.2 - Alert Frequency Statistics

Most Frequent Alerts	Alert Rank	Occurrences
High port 65535 tcp - possible Red Worm - traffic	6	95469
SMB Name Wildcard	10	39730
spp_HTTP_decode: IIS Unicode attack detected	7	31958
Watchlist 000220 IL-ISDNNET-990517		26086
TFTP - External UDP connection to internal TFTP server	3	17287
High port 65535 udp - possible Red Worm - traffic	6	4391
spp_HTTP_decode: CGI Null Byte attack detected	4	2198
Watchlist 000222 NET-NCFC		1817
Possible Trojan server activity	5	1560
Port 55850 tcp - Possible myserver activity - ref. 01	8	1335
Queso fingerprint		1253
IDS552/web-iis_IIS ISAPI Overflow ida nosize		1185
Null scan!		601
EXPLOIT x86 NOOP		558
Incomplete Packet Fragments Discarded		431
TFTP - Internal TCP connection to external TFTP server	3	424
SUNRPC highport access!		368
IRC evil - running XDCC	9	197
SMB C access		152
TCP SRC and DST outside network		143
NMAP TCP ping!		132
scan (Externally-based)		113
EXPLOIT x86 setuid 0		84
ICMP SRC and DST outside network		74
TFTP - Internal UDP connection to external TFTP server	2	56
EXPLOIT x86 setgid 0		42
TFTP - External TCP connection to internal TFTP server	2	7

EXPLOIT NTPDX buffer overflow		6
DDOS shaft client to handler		5
HelpDesk MY.NET.83.197 to External FTP		5
External FTP to HelpDesk MY.NET.70.49		4
NIMDA - Attempt to execute cmd from campus host	1	4
External FTP to HelpDesk MY.NET.70.50		4
MY.NET.30.4 activity		1
connect to 515 from inside		1
SITE EXEC - Possible wu-ftpd exploit – GIAC000623		1
MY.NET.30.3 activity		1
Probable NMAP fingerprint attempt		1
Bugbear@MM virus in SMTP		1

Most Frequent Alerts Generated by an Internal Source IP Address	Count	Most Frequent Alerts Generated by an External Source IP Address	Count
High port 65535 tcp - possible Red Worm - traffic	38794	High port 65535 tcp - possible Red Worm - traffic	56675
spp_HTTP_decode: IIS Unicode attack detected	27842	SMB Name Wildcard	39730
TFTP - External UDP connection to internal TFTP server	17220	Watchlist 000220 IL-ISDNNET-990517	26086
High port 65535 udp - possible Red Worm - traffic	2262	spp_HTTP_decode: IIS Unicode attack detected	4116
spp_HTTP_decode: CGI Null Byte attack detected	2067	High port 65535 udp - possible Red Worm - traffic	2129
Possible Trojan server activity	987	Watchlist 000222 NET-NCFC	1817
Port 55850 tcp - Possible myserver activity - ref. 01	778	Queso fingerprint	1253
TFTP - Internal TCP connection to external TFTP server	206	IDS552/web-iis_IIS ISAPI Overflow ida nosize	1185
IRC evil - running XDCC	197	Null scan!	601
TFTP - Internal UDP connection to external TFTP server	49	Possible Trojan server activity	573
Port 55850 udp - Possible myserver activity - ref. 01	12	EXPLOIT x86 NOOP	558
RFB - Possible WinVNC - 010708-1	11	Port 55850 tcp - Possible myserver activity - ref. 01	557
HelpDesk MY.NET.83.197 to External FTP	5	Incomplete Packet Fragments Discarded	431
NIMDA - Attempt to execute cmd from campus host	4	SUNRPC highport access!	368
connect to 515 from inside	1	TFTP - Internal TCP connection to external TFTP server	218
		SMB C access	152
		TCP SRC and DST outside network	143
		NMAP TCP ping!	132
		spp_HTTP_decode: CGI Null Byte attack detected	131
		scan (Externally-based)	113
		EXPLOIT x86 setuid 0	84
		ICMP SRC and DST outside network	74
		TFTP - External UDP connection to internal TFTP server	67
		EXPLOIT x86 setgid 0	42
		Attempted Sun RPC high port access	14
		EXPLOIT x86 stealth noop	11
		Tiny Fragments - Possible Hostile Activity	11
		TFTP - Internal UDP connection to external TFTP server	7
		TFTP - External TCP connection to internal TFTP server	7
		EXPLOIT NTPDX buffer overflow	6
		DDOS shaft client to handler	5

	External FTP to HelpDesk MY.NET.70.50	4
	External FTP to HelpDesk MY.NET.70.49	4
	Port 55850 udp - Possible myserver activity - ref. 01	3
	RFB - Possible WinVNC - 010708-1	2
	MY.NET.30.3 activity	1
	SITE EXEC - Possible wu-ftpd exploit - GIAC000623	1
	Bugbear@MM virus in SMTP	1
	Probable NMAP fingerprint attempt	1
	MY.NET.30.4 activity	1

3.4.3 - Alert Port Frequency Statistics

Top 25 Source Ports	Possible Activity	Count	Top 25 Source Ports	Possible Activity	Count
65535	RCI Trojan or Adore Worm	40951	65535	RCI Trojan	58890
80	HTTP	20450	137	NetBIOS-ns	39725
69	TFTP	17441	80	HTTP	36302
1025	Maverick's Matrix 1.2 – 2.0 Trojan	6246	2130		2946
2130		4904	4168		2034
1026	Popadstop Trojan, Nterm or PalTalk V.5	4795	6257		1954
1027	PopadStop Trojan,ICQ or PalTalk V.5	4534	1214	Kazaa, Morpheus or Grokster	1805
1028	Popadstop Trojan, or PalTalk V.5	3650	1100		1739
1029	Popadstop Trojan, or PalTalk V.5	2878	6699		1418
4168		2510	4083	Abacast P2P	1368
6257		2059	1254		1353
1030	iad1	1895	2095		1327
4083		1516	1237		1242
3115		1312	None		1216
1214	Kazaa, Morpheus or Grokster	1267	4089		1150
4089	possible Abacast P2P	1252	3115		1007
35168		1174	27374	Bad Blood, SubSeven , SubSeven 2.1 Gold, SubSeven 2.1.4, or DefCon 8 Trojans	984
None		1172	4180		925
26499	possible Quake Gaming	1160	3037		882
3169		1142	35168		877
4180		1134	2887		864
3037		1118	26499		858
3545		1019	25	SMTP; Ajan, Antigen, Email Password Sender - EPS, EPS II, Gip, Gris, Happy99, Hpteam mail, I love you, Kuang2, Magic Horse, MBT (Mail Bombing Trojan), Moscow Email Trojan, Naebi, NewApt worm, ProMail Trojan, Shtirlitz, Stealth, Tapiras, Terminator, WinPC, or WinSpy Trojans	802
1086	NetSpy Trojan	1005	3169		782
1033	RCI Trojan or Adore Worm	967	55850	MyServer Trojan	778

3.4.4 - Most Severe Alerts Criterion

The criterion is based on these factors:

a. Internal Source. Alerts with an internal source are far more dangerous than those with external sources. If the internal host(s) is the source of bad traffic this can mean that the host has already been compromised which can lead to an attacker or infection spreading itself further into the network. Bad traffic from an external host can be dangerous as well, but doesn't necessarily mean that an internal host has been compromised.

b. Lethality. This ties in with Severity and Criticality; however, Lethality is a far better indicator of severity because it is based on the impact to the host and the network. Assuming that the target host(s) in question are alive and have critical importance, the relevant consideration is how damaging the attack would be on the target host(s) if it succeeded.

c. Number of Alerts. This is a secondary consideration to lethality, as an alert with a high total count doesn't necessarily mean it's the most dangerous. For example, which would be more dangerous? 10,000 successful OS fingerprinting scans or 1 successful root exploit. My answer to that would definitely be the root exploit.

3.4.5 - Most Server Alerts

3.4.5.1 - Most Severe Alert #1

NIMDA - Attempt to execute cmd from campus host	High	4

3.4.5.1.1 - Alerts Sample:

```
01/06-15:30:27.350895  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.130.187:2546 -> 207.68.132.9:80
01/07-14:21:09.133358  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.153.158:1106 -> 207.68.132.9:80
01/08-19:49:20.827443  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.83.183:1062 -> 65.54.250.120:80
01/09-18:52:28.211248  [**] NIMDA - Attempt to execute cmd from campus host [**]
MY.NET.109.59:1077 -> 65.54.250.120:80
```

3.4.5.1.2 - Summary

Nimda is a worm that can propagate itself in many ways such as through email, web browsers, NetBIOS shares, and web servers. An excellent attack description can be found in the SANS Reading Room at HTTP://www.sans.org/rr/papers/index.php?id=95. These Nimda alerts are very concerning given that they're coming from Internal Hosts. These internal hosts are most likely infected with Nimda and what we see here is Nimda attacking a couple of Microsoft servers at port 80. As you can see in the message, Nimda is attempting to remotely execute Microsoft's cmd.exe (Microsoft's shell) on the target host. Nimda takes advantage of certain “superfluous decoding” and Directory Traversal vulnerabilities found in unpatched version of Microsoft’s IIS server. Nimda will attempt to execute cmd.exe in several server directory structures. Nimda can attempt execution directly (e.g. GET /c/winnt/system32/cmd.exe?/c+dir) or by obfuscating characters utilizing Unicode (e.g. GET

HTTP://127.0.0.1/scripts/..%c1%1c..?winnt/system32/cmd.exe?/c+dir+c:\). If executing cmd.exe on the IIS server is successful, it will then command the victim IIS server to download a copy of itself from the attacking Nimda's own TFTP service. According to the alerts, there weren't any TFTP connections that were made from these two Microsoft servers to these Nimda infected hosts, thus it appears that Nimda failed in propagating itself from the University network to the Microsoft servers.

I've included Registration Information regarding the two target hosts. It can be found in the Registration Section near the end of this document.

3.4.5.1.3 - Tables

Source IP	Occurrences
MY.NET.153.158	1
MY.NET.83.183	1
MY.NET.109.59	1
MY.NET.130.187	1

Destination IP	Ocurrences
65.54.250.120	2
207.68.132.9	2

3.4.5.1.4 - Recommendations

User Systems: These four internal hosts will need to be cleaned up with a virus scanner using the latest definitions (DAT files). Virus definitions should be downloaded daily and all email and JavaScript scanning should be enabled. In addition, a proxy and/or web content filter such as squid or Surf Control should be considered to limit and protect from web surfing activities.

Mail Servers: As an additional layer of virus scanning, the University should consider deploying virus scanning on the email server itself, whereas every incoming and outgoing email is scanned before it can leave the mail server. For example, Mimedefang on a Sendmail server is capable of scanning email attachments using multiple virus scanners simultaneously. At a commercial level, products like Surf Control can also help filter malicious emails and web content.

Web Servers: All Microsoft IIS web servers should be updated with the latest service packs and patches. Also, it's important that all Microsoft servers are scheduled to receive automatic updates and to install them automatically, especially for critical updates. Lastly, a thorough virus scan and clean up will be in order for all servers running IIS.

File Servers: Running and maintaining (getting virus definitions daily) a virus scanner on a file server can help to prevent viruses from propagating and can help to protect the important files being stored.

Network: Various ingress and egress technologies can be used to filter content entering and leaving the network. Technologies such as Cisco's Network-Based Application Recognition (NBAR) or CheckPoint Firewall with Application Intelligence (FP4) should be considered.

3.4.5.1.5 - Correlations

www.cisco.com/warp/public/63/nimda.shtml [2]

HTTP://www.sans.org/rr/papers/index.php?id=95 [3]
HTTP://www.cert.org/advisories/CA-2001-26.html [4]

3.4.5.2 - Most Severe Alert #2

TFTP - External TCP connection to internal TFTP server	High	7
TFTP - External UDP connection to internal TFTP server	High	17220

3.4.5.2.1 - Alerts Sample

01/06-12:41:58.491443 [**] TFTP - External UDP connection to internal TFTP server [**] 192.168.0.253:1539 -> MY.NET.5.74:69
01/06-12:52:44.586196 [**] TFTP - External UDP connection to internal TFTP server [**] MY.NET.111.231:69 -> 192.168.0.253:5524

3.4.5.2.2 - Summary

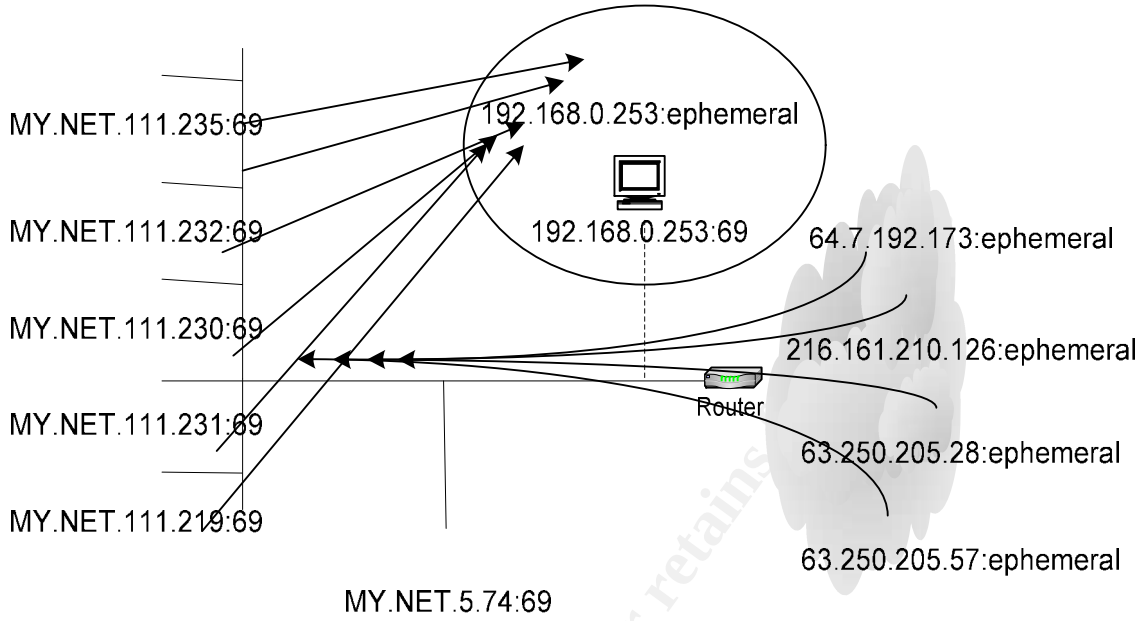
TFTP (Trivial File Transfer Protocol) is commonly used by diskless workstations allowing then to boot up to the network, connect up to a BOOTP server, retrieve it's boot files, and operate without a hard drive. TFTP is also used by routers (such as Cisco) in a process of upgrading the IOS (Operating System) of the router. TFTP has many legitimate uses, too many to list here, but very few of these need to cross the network boundary (internal to external hosts).

Here is a basic chronology of how a TFP connection is initiated:

1. The host requesting the TFTP connection will send a connection request (WRQ) to the target host's port 69.
2. The target host selects an ephemeral port for communication and respond with an ACK to the target host at port 69.

Looking closer at the alerts, I found a lot of abnormal activity. All 64 packets from 192.168.0.253 were destined for MY.NET.5.74:69. Obviously, this is non-routable and we don't see any response from MY.NET.5.74:69 (which should show up in the alerts given it's an internal host at port 69). What is even more peculiar is that the rest of the alerts appear to be from various MY.NET.0.0:69 hosts destined to 192.168.0.253:ephemeral with no response here either. Nowhere do we see a full-connection taking place. This doesn't indicate scanning activity as you'd typically see a routable host probing an entire subnet at port 69 to find out if the port is open or closed, source routing is possible but then with all of the MY.NET.0.0:69 traffic to 192.168.0.253:ephemeral, we do not see an initial packet to MY.NET.0.0:69 with the exception of MY.NET.5.74. Spoofing is probable and the most likely explanation. Just looking at the pattern of things (without logs mind you), it looks like some sort of Distributed DoS attack, but then we should see a lot more traffic than this. Simply put, we can't come to any conclusion without log files and an accurate analysis can't be made without further investigation; however, regardless of what is happening, we need to take a serious look at each internal host to assess possible worms and/or Trojans. It's better to be safe than sorry, and for the sake of security, we must assume the worst with these hosts.

3.4.5.2.3 - Data Link Graph: TFTP - External UDP Connection to Internal TFTP Server



3.4.5.2.4 - Tables

Source	Occurences
MY.NET.111.235	3470
MY.NET.111.232	3465
MY.NET.111.230	3452
MY.NET.111.231	3417
MY.NET.111.219	3412
192.168.0.253	64
216.161.210.126	7
63.250.205.28	1
63.250.205.57	1
64.7.192.173	1

Destination IP Address	Occurrences
192.168.0.253	17216
MY.NET.5.74	64
MY.NET.70.198	7
MY.NET.153.201	1
MY.NET.177.61	1
MY.NET.88.164	1

3.4.5.2.5 - Recommendations

Internal Machines have no business running a TFTP server. An internal vulnerability assessment (i.e.Nessus) should be performed to ascertain which machines are running TFTP servers. An action plan should be deployed, closing unused ports, upgrading and patching vulnerable applications and installing a stateful firewall that blocks port 69 from entering and leaving the network.

3.4.5.2.6 - Correlations

HTTP://www.faqs.org/rfcs/rfc1350.html [5]

3.4.5.3 - Most Severe Alert #3

TFTP - Internal UDP connection to external TFTP server	High	49
TFTP - Internal TCP connection to external TFTP server	High	206

3.4.5.3.1 - Alerts Sample

01/06-04:07:30.110151 [**] TFTP - Internal UDP connection to external TFTP server [**] MY.NET.130.187:2195 -> 192.168.1.1:69
01/06-07:12:21.862963 [**] TFTP - Internal UDP connection to external TFTP server [**] MY.NET.130.187:2276 -> 62.217.98.2:69

3.4.5.3.2 - Summary

TFTP activity most likely indicates that there are numerous worm and Trojan infections and what we see here is our internal hosts being compromised as they're opening up a TFTP connection to an external TFTP server where it will then download the worm. Port 69 is used by TFTP and is what triggered these alerts. Here's a chronology of the activity between 192.168.1.1, MY.NET.130.187 and 62.217.98.2:

1. MY.NET.130.187:(several 2,1XX, 2,2XX ports) to 192.168.1.1:69 triggering several "TFTP-Internal UDP connection to external TFTP server" alerts. It doesn't appear that 192.168.1.1:69 ever responded.

2. MY.NET.130.187 at several ports between 2276-2291 then triggers several "TFTP-Internal UDP connection to external TFTP server" alerts to 62.217.98.2:69 (an external server).

Given that the next set of relevant alerts are IIS Unicode related (possible directory traversal) I thought these comments found on incidents.org were interesting, but they don't lead us to any definitive conclusions. Excerpt taken from Incidents.org website. Search on port 69: HTTP://isc.incidents.org/port_details.html?port=69&repax=1&tarax=2&srcax=2&percent=N&days=40 [6]

CVE ID	Protocol	Source Port	Targetport
Description			
CVE-1999-0183	udp	any	69
Linux implementations of TFTP would allow access to files outside the restricted directory.			
CAN-2002-1209	udp	any	69
Directory traversal vulnerability in SolarWinds TFTP Server 5.0.55, and possibly earlier, allows remote attackers to read arbitrary files via ".."" (dot-dot backslash) sequences in a GET request			

3. Four Seconds after the last alert in step 2 we see an "spp_HTTP_decode: IIS Unicode attack detected" alert from 62.217.98.2:69 to MY.NET.130.187:80.

It's quite obvious that 192.168.1.1:69 is not normal traffic as it is not globally routable. Many Viruses/Worms attack the 192.168.0.0 subnet as it is the most commonly used private network address in use per CIDR. Given the traffic pattern, I would classify this as malicious activity, which requires immediate

investigation. These other Internal Hosts will need to be investigated as well as it is doubtful that any TFTP activity is legitimate.

3.4.5.3.3 - Tables

Source IP Address	Occurences
MY.NET.130.187	**27**
MY.NET.114.45	7
MY.NET.83.171	6
MY.NET.88.238	4
63.250.214.139	2
64.7.192.173	2
MY.NET.178.42	1
208.153.50.192	1
195.92.252.254	1
MY.NET.177.62	1
64.152.216.85	1
MY.NET.84.22	1
MY.NET.198.34	1
| MY.NET.150.216	1

Destination IP Address	Occurrences
192.168.1.1	**16**
62.217.98.2	**11**
62.210.116.150	11
130.10.2.134	5
62.210.116.11	4
130.13.105.43	2
MY.NET.88.164	2
MY.NET.183.59	2
MY.NET.90.243	1
MY.NET.151.115	1
MY.NET.87.55	1

3.4.5.3.4 - Recommendations

Once again, there shouldn't be a legitimate reason for TFTP traffic leaving or entering the network. TFTP is a very insecure protocol that shouldn't be used by to a large extent. A firewall should be put in place that blocks both ingress and egress port 69.

3.4.5.3.5 - Correlations

Excerpt taken from Incidents.org website. Search on port 69:
HTTP://isc.incidents.org/port_details.html?port=69&repax=1&tarax=2&srcax=2&percent=N&days=40[6]

3.4.5.4 - Most Severe Alert #4

Spp_HTTP_decode: CGI Null Byte attack detected	Noise	2067

3.4.5.4.1 - Summary

This alert is generated whenever %00 (a null byte) is detected within a CGI form. This alert is generated by the HTTP decode preprocessor. According to the Official Snort.org FAQ, this signature can generate many false alerts "with sites that use cookies with url-encoded binary data, or if you're scanning port 443 and picking up SSL-encrypted traffic". For example, a commonly known false positive is triggered by Netscape cookies. It's also possible that the CGI form itself contains null bytes, a bad programming practice. Looking at the alerts, we see MY.NET.90.242 communicating with many sequential ephemeral ports to 209.185.162.149:80, www.ivillage.co.uk. Also, in another example, the alerts containing MY.NET.99.36:80 (possibly an internal web server) indicate CGI Null Byte activity closely associated with IIS Unicode alerts which is also subject to false positives with the usage of %00 in cookies and SSL encrypted traffic.

3.4.5.4.2 - Tables

Source IP Address	Occurrences	Dest IP Address	Occurrences
MY.NET.90.242	**228**	192.151.53.10	303
MY.NET.90.115	220	209.185.162.149	228
MY.NET.83.53	206	66.37.219.2	182
MY.NET.86.124	182	**MY.NET.99.36**	**131**
MY.NET.87.52	141	66.129.106.116	120
MY.NET.99.148	120	66.135.192.220	102
MY.NET.81.58	120	66.135.192.226	93
MY.NET.90.148	83	64.14.122.229	91
MY.NET.99.46	72	66.135.209.133	82
MY.NET.82.22	71	66.135.192.227	79
35.10.87.70	69	131.118.254.40	47
MY.NET.104.143	69	66.135.208.201	41
MY.NET.152.251	64	64.40.225.205	39
193.10.173.142	62	209.10.239.135	33

3.4.5.4.3 - Recommendations

If through further research these alerts are deemed to be false alerts and it's found that CGI forms aren't in use or aren't vulnerable, then it may be best to simply disable this in Snort (preprocessor HTTP_decode: 80 8080 -unicode –cginull), in the interest of reducing false alarms and avoiding the “boy who cried wolf” effect with the Security Administrator. Signatures such as 1002, 1256 will do a much better job at identifying worms than a blanket Unicode based alert. However, if CGI is heavily used on University web severs, I would recommend that full logging be performed to better ascertain this activity and that null byte usage in CGI forms be discouraged. An excellent article on securing CGI forms can be found at:
HTTP://www.sans.org/top20/oct02.php[7]

3.4.5.4.4 - Correlations

John Ellis mentions this in his practical located at
HTTP://www.giac.org/practical/Joe_Ellis_GCIA.doc [8]
Also, an excellent description of the CGI Null Byte detect can also be found at HTTP://www.snort.org/docs/FAQ.txt[9] section 4.12.

3.4.5.5 - Most Severe Alert #5

Possible Trojan server activity	High	987

3.4.5.5.1 - Alerts Sample

01/09-11:11:25.549091 [**] Possible Trojan server activity [**] 204.48.169.252:27374 -> MY.NET.179.77:80
01/09-11:11:25.549736 [**] Possible Trojan server activity [**] 204.48.169.252:27374 -> MY.NET.179.77:80
01/09-11:11:25.553715 [**] Possible Trojan server activity [**] MY.NET.179.77:80 -> 204.48.169.252:27374
01/09-11:11:25.554841 [**] Possible Trojan server activity [**] MY.NET.179.77:80 -> 204.48.169.252:27374
01/09-11:11:25.554859 [**] Possible Trojan server activity [**] MY.NET.179.77:80 -> 204.48.169.252:27374

01/09-14:47:39.523218 [**] Possible Trojan server activity [**] 63.79.101.3:27374 -> MY.NET.25.21:110
01/09-14:47:39.523321 [**] Possible Trojan server activity [**] MY.NET.25.21:110 -> 63.79.101.3:27374
01/09-14:47:39.523333 [**] Possible Trojan server activity [**] MY.NET.12.4:110 -> 63.79.101.3:27374
01/09-14:47:39.523477 [**] Possible Trojan server activity [**] MY.NET.25.21:110 -> 63.79.101.3:27374
01/09-14:47:39.523495 [**] Possible Trojan server activity [**] MY.NET.12.4:110 -> 63.79.101.3:27374
01/09-14:47:40.755333 [**] Possible Trojan server activity [**] MY.NET.25.21:110 -> 63.79.101.3:27374
01/09-14:47:40.755368 [**] Possible Trojan server activity [**] MY.NET.12.4:110 -> 63.79.101.3:27374
01/10-02:13:44.865343 [**] Possible Trojan server activity [**] 81.72.113.117:27374 -> MY.NET.137.18:6346
01/10-02:16:31.373522 [**] Possible Trojan server activity [**] MY.NET.91.104:1214 -> 68.18.228.205:27374
01/10-10:40:32.224172 [**] Possible Trojan server activity [**] 194.206.161.161:27374 -> MY.NET.163.107:1214

3.4.5.5.2 - Summary

Several well-known Trojans all use port 27374 to communicate with, most notably SubSeven. These alerts were triggered based on discovering either a src or dst port 27374 in the IP Header. This is the port that SubSeven typically listens on and can indicate that an attacker is either trying to discover hosts that are already compromised or to handle one that is (backdoor). If this were SubSeven activity, you would typically see scans on port 1243 or an entire subnet; however, there are other Trojans that use port 27374 as well. It is entirely possible that 27374 could be a randomly chosen ephemeral port in a legitimate (non-Trojan) connection. In these alerts a two way communication takes place between several internal to external hosts with sources ports 80, 110, 1214, and 6346 all communicating to port 27374. There are also external hosts communicating to internal hosts with a destination port of 27374. Allow me to break it out by port.

Port 1214 to 27374:

It is important to note that most of the activity occurred between these two ports. Port 1214 is associated with Kazaa and Morpheus file sharing programs. The Kazaa client will open local port 1214 for outbound requests to remote port 80 to find the distributed network. When the client finds an item of interest to download, it will select for itself an ephemeral port (>1023 and <=65535) and send a request to the remote host (which has the file) with a dest port of 1214. The client will also service requests made by remote hosts using port 1214. In these incidents it is likely that an external host was attempting to pull a file (possible a music file) from an internal host. In this case, the external host selected an ephemeral port of 27374 (which triggered these alerts). What we're seeing is an alert being triggered for each and every packet being sent between the local and external Kazaa hosts.

Ports 80, 110:
It's important to note that Kazaa Lite 2.0 allows the user to change the default port of 1214 to whatever they want. Perusing various discussion boards on this topic I found that it's quite common for Kazaa users to select port 80, 110 along with other ports that are typically not blocked by a firewall.

Port 6346:
This port is associated with another P2P program called Gnutella, which connects to peers on ports 6346 and 6347.

Here's an interesting OOS entry related to these port 6346:
01/06-23:18:12.868392 **148.64165.75:1445 -> MY.NET.184.37:6346**
TCP TTL:111 TOS:0x0 ID:4503 IpLen:20 DgmLen:365 DF
****P*** Seq: 0x6EE6920A Ack: 0x0 Win: 0x2000 TcpLen: 20
47 4E 55 54 45 4C 4C 41 20 43 4F 4E 4E 45 43 54 **GNUTELLA CONNECT**
2F 30 2E 36 0D 0A 55 73 65 72 2D 41 67 65 6E 74 /0.6..User-Agent
3A 20 4D 6F 72 70 68 65 75 73 20 32 2E 30 2E 31 : **Morpheus 2.0.1**
2E 38 0D 0A 58 2D 55 6C 74 72 61 70 65 65 72 3A .8..X-Ultrapeer:
20 46 61 6C 73 65 0D 0A 50 4F 4E 47 2D 43 41 43 False..PONG-CAC
48 49 4E 47 3A 20 30 2E 31 0D 0A 58 2D 4D 59 2D HING: 0.1..X-MY-
41 44 44 52 45 53 53 3A 20 31 34 38 2E 36 34 2E ADDRESS: 148.64.
31 36 35 2E 37 35 3A 37 36 38 39 0D 0A 58 2D 54 165.75:7689..X-T
72 79 3A 20 32 34 2E 32 32 39 2E 36 34 2E 32 33 ry: 24.229.64.23
31 3A 36 33 34 36 2C 31 36 39 2E 32 35 34 2E 32 1:6346,169.254.2

3.4.5.5.3 - Tables

Source IP Address	Occurrences	Destination IP Address	Occurrences
MY.NET.91.104	966	217.235.45.31	965
217.235.45.31	552	MY.NET.91.104	553
MY.NET.113.4	7	MY.NET.113.4	8
24.56.223.18	5	63.79.101.3	6
MY.NET.179.77	3	24.56.223.18	4
194.206.161.161	3	204.48.169.252	3
MY.NET.25.21	3	MY.NET.163.252	3
MY.NET.12.4	3	MY.NET.163.107	3
MY.NET.91.104	3	MY.NET.137.18	3
204.48.169.252	2	MY.NET.179.77	2

3.4.5.5.4 - Recommendations
I believe that the majority of these alerts triggered by port 27374 are the result of a randomly selected ephemeral port of 27374 by the external host. Special concern should be warranted if external hosts are initiating communication to an internal host with a destination port of 27374 for this likely indicates a compromised host. It may be in the University's interest to block port 1214 as Kazaa and Morpheus are associated with illegal copyright infringement and it's been documented that 5% of all Kazaa downloads contain viruses.

3.4.5.5.5 - Correlations
HTTP://www.sans.org/rr/papers/36/953.pdf[10]

HTTP://www.securityfocus.com/archive/75/194328[11]

3.4.5.6 - Most Severe Alert #6

High port 65535 tcp - possible Red Worm - traffic	High	38794
High port 65535 udp - possible Red Worm - traffic	High	2262

3.4.5.6.1 - Alerts Sample

01/06-00:33:41.761236 [**] High port 65535 tcp - possible Red Worm - traffic [**]
212.95.85.172:1540 -> MY.NET.84.151:65535
01/06-00:33:41.761994 [**] High port 65535 tcp - possible Red Worm - traffic [**]
MY.NET.84.151:65535 -> 212.95.85.172:1540
01/06-00:33:45.780939 [**] High port 65535 tcp - possible Red Worm - traffic [**]
80.15.129.69:3844 -> MY.NET.84.151:65535
01/06-00:33:45.781127 [**] High port 65535 tcp - possible Red Worm - traffic [**]
MY.NET.84.151:65535 -> 80.15.129.69:3844
01/06-00:27:21.251936 [**] High port 65535 tcp - possible Red Worm - traffic [**]
172.180.111.226:4037 -> MY.NET.84.151:65535

3.4.5.6.2 - Summary

With a count of 95,469 alerts generated, this signature tops out the alert charts as the most frequent alert. There are 2777 different source ports within these alerts. Given that all of the source port 65535 activity is from MY.NET.0.0 hosts, it appears that many University systems have been compromised, possibly by Adore. However, this could very well be a number of worms and/or Trojans that all use this port. The external source host, source port of 1540 (rds), shows up in numerous alerts all coming from 212.95.85.172. This is very suspicious activity given that 1540 isn't a commonly used port and there doesn't appear to be any reason our internal hosts should be communication with an external host at 1540. I suspect that we see here is external hosts communicating with our compromised systems.

The Adore worm scans for vulnerabilities in four common Linux daemons, LPRng, rpc-statd, wu-ftpd and BIND. More details about Red Worm (Adore Worm) can be found at www.sans.org/y2k/adore.htm. Once a machine has been compromised by this worm, it will then set up a backdoor listening on port 65535. This backdoor is activated by sending the host an ICMP echo request with 77 bytes of data (Magic Packet). Receiving this Magic Packet the target host will fork a command shell to local port 65535. I would think that a "large ICMP packet" alert would be triggered if such activity were taking place. The key problem with Adore is that it this backdoor listening port lingers even after Adore has done its thing. Full logs would be helpful in determining whether or not Magic Packets were sent before these two-way connections were made.

3.4.5.6.3 - Tables

Source IP Addresses	Occurrences
MY.NET.84.151	21366
MY.NET.88.193	17407
172.186.226.148	5102
67.69.224.186	2529
80.200.137.128	2461
193.252.60.115	2010
80.14.209.119	1722
80.14.126.37	1507
80.13.193.40	1498
80.14.113.219	1347
194.206.50.2	1166
MY.NET.83.146	1104
81.48.81.146	1091

Destination IP Address	Occurrences
MY.NET.84.151	28958
MY.NET.88.193	27632
172.186.226.148	3078
67.69.224.186	2047
80.200.137.128	1877
193.252.60.115	1534
80.14.126.37	1354
81.48.81.146	1117
80.13.193.40	1096
80.14.209.119	1048
80.14.113.219	1042

3.4.5.6.4 - Recommendations

A firewall needs to be in place to block port 65535. Also, using and maintaining Virus Scanners on all machines will help to prevent the spread of infections.

3.4.5.6.5 - Correlations

HTTP://www.sans.org/y2k/adore.htm[12]

3.4.5.7 - Most Severe Alert #7

spp_HTTP_decode: IIS Unicode attack detected	Moderate	27842

3.4.5.7.1 - Alerts Sample

01/06-07:12:21.862963 [**] TFTP - Internal UDP connection to external TFTP server [**]
MY.NET.130.187:2276 -> 62.217.98.2:69
01/06-07:12:21.896883 [**] TFTP - Internal UDP connection to external TFTP server [**]
MY.NET.130.187:2276 -> 62.217.98.2:69
01/06-07:12:27.862782 [**] TFTP - Internal UDP connection to external TFTP server [**]
MY.NET.130.187:2277 -> 62.217.98.2:69
01/06-07:12:27.881741 [**] TFTP - Internal UDP connection to external TFTP server [**]
MY.NET.130.187:2277 -> 62.217.98.2:69
01/06-07:12:31.816195 [**] TFTP - Internal UDP connection to external TFTP server [**]
MY.NET.130.187:2279 -> 62.217.98.2:69
01/06-07:12:31.849926 [**] TFTP - Internal UDP connection to external TFTP server [**]
MY.NET.130.187:2279 -> 62.217.98.2:69
01/06-07:12:35.850529 [**] spp_HTTP_decode: IIS Unicode attack detected [**]
62.217.98.2:4684 -> MY.NET.130.187:80
01/06-07:12:35.850529 [**] spp_HTTP_decode: IIS Unicode attack detected [**]
62.217.98.2:4684 -> MY.NET.130.187:80
01/06-07:12:35.850529 [**] spp_HTTP_decode: IIS Unicode attack detected [**]
62.217.98.2:4684 -> MY.NET.130.187:80
01/06-07:12:37.168514 [**] spp_HTTP_decode: IIS Unicode attack detected [**]
62.217.98.2:4704 -> MY.NET.130.187:80
01/06-07:12:38.486137 [**] spp_HTTP_decode: IIS Unicode attack detected [**]
62.217.98.2:4716 -> MY.NET.130.187:80

3.4.5.7.2 - Summary

Nimda, Code Red and others worms all exploit a well-known vulnerability in Microsoft's IIS that has to do with the substituting Unicode representations (e.g. %255c) in place of actual characters. The Unicode representations and the actual ASCII characters they represent are essentially the same, however can be interpreted differently in IIS URL decoding. Without writing an entire paper on this subject, suffice to say that it's this inconsistency that can allow give an attacker access to directories that would otherwise be secured from external access.

It's important to note that this alert is well known to generate a lot of false positives as Unicode can also be found in legitimate traffic. It would be useful to look at the full log to see exactly what the content was within these packets. It appears that what we're seeing is a variety of internal compromises and alerts being triggered by both internal and external sources.

In the Alert sample I included an interesting correlation between two hosts that triggered two alerts in the same time frame. The first alert was for an internal UDP connection to an external TFTP server.

3.4.5.7.3 - Tables

Source IP Address	Occurrences	Destination IP Addresses	Occurrences
MY.NET.112.204	6785	61.236.39.3	6785
MY.NET.85.74	4135	207.200.86.86	2227
MY.NET.84.133	1264	207.200.86.97	2157
MY.NET.85.87	1251	MY.NET.70.207	1147
MY.NET.183.59	926	207.200.89.193	1015
MY.NET.122.118	820	MY.NET.99.36	966
195.25.191.82	731	64.95.120.131	760
MY.NET.145.197	530	211.233.32.56	636
MY.NET.116.84	529	64.12.42.117	494
MY.NET.151.120	524	199.244.218.42	480

3.4.5.7.4 - Recommendations

Regardless of whether these are false positives or the byproduct of compromised machines, it's obvious that the University has a lot of work to do in cleaning up and preventing compromises. With a firewall in place, a web proxy, full snort logs and virus protection, the University can focus on using a better Snort rule base to get a better reading of the malicious activity. Specifically, I would recommend using the -Unicode option on the HTTP_decode line in your snort.conf file. Signatures such as 1002, 1256 will do a much better job at identifying worms than a blanket Unicode based alert.

3.4.5.7.5 - Correlations

HTTP://www.kb.cert.org/vuls/id/111677[13]

3.4.5.8 - Most Severe Alert: #8

Port 55850 tcp - Possible myserver activity - ref. 01	Moderate	778
Port 55850 udp - Possible myserver activity - ref. 01	Moderate	12

3.4.5.8.1 - Alerts Sample

01/06-00:23:36.617407 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**] MY.NET.188.24:55850 -> 10.0.1.1:192
01/06-11:04:52.894697 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**] 207.200.89.193:80 -> MY.NET.139.52:55850
01/06-11:04:53.794553 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**] 207.200.89.193:80 -> MY.NET.139.52:55850
01/06-11:04:53.794675 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**] 207.200.89.193:80 -> MY.NET.139.52:55850
01/06-18:23:11.968143 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**] 24.210.218.39:55850 -> MY.NET.83.146:2502
01/06-19:55:52.876557 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**] MY.NET.6.40:25 -> 205.158.62.146:55850

3.4.5.8.2 - Summary

These alerts are triggered when the source or destination port is 55850, a port typically associated with a DDoS tool known as Myserver. Once myserver connects up to UDP 55850, it installs a rootkitted versions of Unix ls and ps in order to hide itself from administrators; however, you will see the port with the netstat -ap | egrep -i 'LISTEN|upd' command. Both the rootkit and DDoS code are stored in /lib on the victims machine. Although some of these alerts could be indication of this Trojan, it is entirely possible that 55850 was chosen as an ephemeral port and that many of these alerts could be false positives. This is especially convincing given that port 55850 was communicating with a variety of commonly used ports such as 20 (ftp-data), 80(HTTP), 1214 (Kazaa), 25 (SMTP), uncharacteristic of the Myserver Trojan. Detailed information including the TCP state would help in determining what this is. It's important to note that MY.NET hosts rarely initiated communication with dest port 55850. I would venture to say that most of these alerts are false positives; however, there are some oddities. For example, the first alert sample (above) shows a source of MY.NET.188.24:55850 and a destination of 10.0.1.1:192. It's obvious that 10.0.1.1:192 isn't a local network and it isn't globally routable, this would indicate that MY.NET.188.24 is likely infected with a Trojan or Worm. This host should investigated for signs of a Trojan or worm compromise.

3.4.5.8.3 - Tables

Source IP Address	Occurrences
MY.NET.162.67	664
150.163.200.98	448
MY.NET.70.231	34
62.31.64.2	19
MY.NET.6.40	18
MY.NET.29.3	15
MY.NET.140.136	13
68.55.22.122	12
MY.NET.113.4	7
205.158.62.146	7

Destination IP Address	Occurrences
150.163.200.98	664
MY.NET.162.67	448
62.31.64.2	33
MY.NET.70.231	21
MY.NET.6.40	19
68.55.22.122	15
MY.NET.29.3	12
MY.NET.113.4	10
10.0.1.1	6

3.4.5.8.4 - Recommendations
Once again, I believe the majority of this activity to be the ephemeral port byproduct of all the P2P activity. However some further investigation is warranted given the lethality of this potential activity. Logs and a vulnerability assessment are in order.

3.4.5.8.5 - Correlations
HTTP://seclists.org/incidents/2000/Oct/0141.html[14]

3.4.5.9 - Most Sever Alert # 9

IRC evil - running XDCC	Moderate	197

3.4.5.9.1 - Alerts Sample:
01/10-21:47:23.913216 [**] IRC evil - running XDCC [**] MY.NET.88.168:4415 -> 216.55.223.121:6667
01/10-22:17:23.874899 [**] IRC evil - running XDCC [**] MY.NET.88.168:4415 -> 216.55.223.121:6667
01/10-22:37:23.906224 [**] IRC evil - running XDCC [**] MY.NET.88.168:4415 -> 216.55.223.121:6667

3.4.5.9.2 - Summary
XDCC is the mechanism by which IRC (Internet Relay Chat) performs Peer to Peer (P2P) file transfers. Information on what triggers this Snort is hard to find and it appears that port 6667 is what triggered these alerts as this is the port typically associated with an XDCC server. XDCC functions in a client/server relationship whereas an XDCC server allows remote IRC users to download music, applications, and movies through the Internet. Most of the music and movie files shared are illegal and break copyright law. The groups involved in hosting such XDCC file sharing often prey on unsuspecting corporate networks in order to avoid the personal liability that would be associated with hosting a legitimate XDCC server on the Internet. Attackers typically compromise systems that have high bandwidth to the Internet, systems that would do well as an XDCC server. Having compromised the server, attackers will typical “root kit” it in order

to cover their tracks and hide the presence of the XDCC server that is now chewing up bandwidth, memory, disk I/O and processor speed.

Fortunately, it doesn't appear that any XDCC servers are running on the University network. All of the relevant alerts give reference to an external server at port 6667. The traffic we're seeing here is likely triggered by users and/or employees using a IRC client to download movies and music.

3.4.5.9.3 - Tables

Source IP Address	Occurrences
MY.NET.88.168	119
MY.NET.105.48	29
MY.NET.150.101	15
MY.NET.150.5	11
MY.NET.114.14	11
MY.NET.112.199	9
MY.NET.88.163	2
MY.NET.84.160	1

Destination IP Address	Occurrences
132.74.40.10	35
24.215.6.241	33
65.116.90.178	25
198.163.214.2	15
193.163.220.3	12
63.151.165.236	12
138.121.129.12	11
206.167.75.78	9
63.151.165.172	8
128.242.65.30	7
216.55.223.121	6
209.126.200.130	5
66.250.50.3	4
209.133.9.47	4
217.8.149.200	4
138.121.51.51	2
194.154.163.23	1
66.216.84.224	1
146.20.20.20	1
65.116.89.130	1
140.186.123.133	1

3.4.5.9.4 - Recommendations

Music and movie industries are starting to crack down on illegal file sharing and liability is starting to be placed on the users who take part in such activity. Just a few users heavily involved in downloading music and movies can chew up a T1 line (1.54Mbps) and bring a network to a crawl. To remedy these liabilities, the University should enact a computer usage policy and educate students and users about the policy. An audit of all systems should be collected (especially those involved in the charts above) and all such P2P file sharing applications should be removed. Lastly, a stateful firewall should be put in place, allowing only the basic services required (port 80, 443, 25, 110, etc.).

3.4.5.9.5 - Correlations

HTTP://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00120.html[15]

3.4.5.10 - Most Severe Alert #10

SMB Name Wildcard	Moderate	39730

3.4.5.10.1 - Snort Signature and Alerts Sample

misc-lib:alert udp any any -> $HOME_NET 137 (msg:"SMB Name Wildcard";

content:"CKAAAAAAAAAAAAAAAAAAAAAAAAAAAA|0000|";)

01/08-05:37:30.575117 [**] SMB Name Wildcard [**] 210.214.110.73:1065 -> MY.NET.132.62:137
01/08-05:37:30.719827 [**] SMB Name Wildcard [**] 210.214.110.73:1065 -> MY.NET.132.63:137
01/08-05:37:30.814096 [**] SMB Name Wildcard [**] 210.214.110.73:1065 -> MY.NET.132.64:137

3.4.5.10.2 - Summary

Microsoft systems utilize a suite of protocols collectively referred to as NetBIOS /SMB that allow users to view and access resources (i.e. files, shared printers) being shared by other users. The SMB Wildcard alert is triggered when a machine is actively attempting to query all NetBIOS/SMB resource information (its NetBIOS table) from another NetBIOS/SMB enabled host through the use of this command: netstat –A IP Address. This command can be manually entered via a command prompt or can be included in code, such as the network.vbs worm does. This query is sent to port 137 (NetBIOS) on the target host. The signature that triggers this alert contains CKAAA…etc. (sum of 32 ASCII characters) followed by a binary 0000. Allow me to break down how CKAAA is equivalent to a wildcard (*).A NetBIOS name has 16 ASCII characters and in the stream assembly, each character in the NetBIOS name is represented by 2 hex characters (ASCII to Hex translation) giving us 32 hex characters. However, there is an additional step performed before it's put on the wire. Each hex character nibble is split and added to a 0x41 byte, thus resulting in a total of 64 hex or 32 ASCII characters (twice as long as before). A wildcard ASCII character of * translates to 2A in hex and any unused space is blank padded. We then need to add an ASCII A or hex 0x41 to each hex character. Thus 2 + 41 = 43, A + 41 = 4B, and 0 (the blank padding) + 41 = 41. So in hex we have 43 4B 41 41 41 41…etc. and translating that back to ASCII (per Snort translating back for our viewing) it represents CKAAAAA…(sum of 32 ASCII characters).

Most of these alerts involve an external host sending an SMB wildcard to port 137 on an internal host. This activity likely indicates that a worm such as network.vbs is attempting to propogate itself to internal hosts. This is an item of serious concern, because once the network.vbs has collected share information about the target hosts it will then attempt to copy itself to unprotected (lack of password) shares in order to propogate itself onto the network.

3.4.5.10.3 - Tables

Source IP Address	Occurrences
200.84.76.226	298
61.234.196.148	283
137.45.69.167	267
212.59.27.204	248
62.89.67.162	243
165.228.7.72	243
203.162.15.76	236
200.164.23.30	234
81.195.171.10	230
61.0.151.239	228

Destination IP Address	Occurrences
MY.NET.132.50	317
MY.NET.137.18	191
MY.NET.190.17	155
MY.NET.6.16	122
MY.NET.137.46	112
MY.NET.133.225	97
MY.NET.133.251	94
MY.NET.134.251	94
MY.NET.133.247	93
MY.NET.134.242	92

3.4.5.10.4 - Recommendations

The top 10 sources are External, and the top 10 destinations are internal, indicating that we have outsiders or worms attempting to access internal resources. All SMB/NetBIOS services (tcp/udp 137, 138, 139) should be blocked (both ingress and egress) at the firewall. A solution such as VPN with NATing (Network Address Translation) should be deployed if legitimate remote users (i.e. professors) need to access internal SMB shares through the Internet.

3.4.5.10.5 - Correlations

HTTP://seclists.org/lists/ids/2000/Mar/0065.html[16]

3.4.6 - Summary of Recommendations

Each detect in this document includes recommendations that are specific to the detect and activity being discussed. These recommendations are broad in scope and present a strategy for the University to follow in resolving the issues at hand as well as preventing future occurrences.

1. Complete Audit and Documentation

   Detailed systems documentation should be created and maintained for all systems, and a detailed network topology diagram should be constructed. The documentation should include information on a computer's OS platform, installed patches, installed applications, IP addresses, its routing table, listening ports, and SMB, NFS shares. It would also be helpful to know the main function of each computer/server and the groups that it is used by. This information is vital to a Security Team in correctly analyzing and handling security incidents in a timely manner. The logical network diagram should include all of the segments, and IP addresses of all hosts, grouped by function and location (Student Lab, vs Server Room). Also a physical network diagram of how key systems are connected to switches/hubs and switches to hubs would also be helpful. Lastly, The network diagram should include addendums of IDS, router and firewall ACL/Rules printouts and configurations.

2. In-Depth Investigation of All Suspected Compromises
   In this document, many theories have been presented as to what has taken place, but without detailed information on the network topology and the systems in place, nothing can be said with certainty. Utilizing an updated Snort with full alerts and binary logging would greatly assist in producing a more accurate analysis. Beyond IDS logs, each system and the network in general should be fully scanned for vulnerabilities using a tool such as Nessus. The output of a Nessus (or the like) as well as host and network documentation (per audit above) should be used along side this analysis to investigate possible compromised hosts and to confirm false positives. Once these security issues have been resolved it would be wise to perform an ongoing vulnerability assessment periodically. I would recommend every month for the first 6 months, and if there aren't any major security incidents in that time frame, then a frequency of every quarter should be sufficient.

3. Deploy a Stateful Firewall with a DMZ (3-Tiered)
   All Public Servers (i.e. University web server, name severs, mail server) should be placed in a firewall segregated DMZ network with a rule base allowing the required external access to the DMZ and severely limiting the DMZ from connecting to internal systems.
   The firewall policy should limit the activity of students and employees and helps to protect them from the outside. Student groups should be limited to just HTTP, HTTPS, SMTP, and POP3. TFTP, NetBIOS, FTP, and other such unnecessary egress services should be explicitly denied in the firewall rule base; however, it is more efficient to specify the allowed services in the rule base and to set an implicit "deny all "at the bottom of the rule base to take care of the unwanted traffic.

4. Upgrade Snort and Snort Rules Set
   Run the latest stable version of Snort with the up-to-date rules.
   Disable Snort HTTP Pre-Processor in the interest of reducing false alarms and avoiding the "boy who cried wolf" effect with the Security Administrator. Set "preprocessor HTTP_decode: 80 8080 -unicode –cginull" in the snort.conf file. Signatures such as 1002, 1256 will do a much better job at identifying worms than a blanket Unicode based alert.

5. Deploy File Integrity Checkers Critical Systems
   File Integrity checkers (i.e. Tripwire, LSOF) can alert the administrator to a possibly compromised system and give the Security Administrator a baseline of processes, files, and ports open, all of which can prove to be extremely useful in such situations where a host has been compromised and is suspected that backdoors and rootkits may be installed. Also, on some UNIX based systems it is possible to turn off the capability of executing code within the stack/buffer. Buffer overflow attacks function by writing a new return pointer over the original one to execute code that has been pushed into the stack. Disabling this in the initial kernel build can prevent buffer overflow and

DoS attacks all together. Lastly, certain Linux kernels are modular which can allow an attacker to build a kernel level rootkit module onto an existing kernel. The modular function can be disabled and depending on the OS platform such capabilities should be investigated.

6. Deploy IDS Sensors on Internal, External, and DMZ Networks
   One Snort sensor should be present per network segment. The University should have one sensor on the external subnet off the firewall, one on the DMZ and one on the internal network. Log should be maintained internally and an effort should be made to slim down the false positives and customize Snort to the Universities environment. The sensors themselves should have two interfaces (one for management and one for sniffing). The sniffing interface shouldn't be assigned an IP address, and the management interfaces should be segregated onto a protected network subnet. The University may also want to consider the implementation of Host-Based IDS and Personal Firewall systems for all systems. There are several open source and commercial Unix and Windows based systems that would help in adding this additional layer of security. An excellent article on this can be found at http://www.sans.org/resources/idfaq/host_based.php[17]

7. Antivirus Software Installation, Configuration, and Daily Updates
   Virus scanning software should be installed and configured on all systems. The virus scanner should be configured to check for updates on a daily basis. All incoming email should be scanned, All JavaScript and VBscript content should be scanned, and an entire system scan should occur at least on a weekly basis. Virus scanning should also occur on the mail server and/or firewall (Checkpoint AI has these features) before reaching the client as an added layer of protection.

8. Enforce Computer Usage Policies (P2P, Gaming, etc.)
   Based on the analysis, it's clear that Gaming and Peer-to-Peer file sharing have become epidemic problems for the University. Firewall rules should take P2P and gaming ports into account and such gaming and P2P applications will need to be removed. More importantly, users need to be educated about the University's policy of "no gaming and no P2P". These activities not only are unproductive use of the student's time, but a waste of bandwidth, and a possible source of liability.

9. Network Baselining and Utilization Tests
   As a last step, once the "new" network is in place with all of the recommendations implemented, network baselining and utilization tests need to be performed to record what normal/stable network traffic looks like.

3.4.7 - Registration Information of Interesting External Hosts

3.4.7.1 - Microsoft Corp
The following Registration relates to the Alerts section on:
Most Severe Alert #1

NIMDA - Attempt to execute cmd from campus host	High	4

Search results for: 65.54.250.120

OrgName: Microsoft Corp
OrgID: MSFT
Address: One Microsoft Way
City: Redmond
StateProv: WA
PostalCode: 98052
Country: US

NetRange: 65.52.0.0 - 65.55.255.255
CIDR: 65.52.0.0/14
NetName: MICROSOFT-1BLK
NetHandle: NET-65-52-0-0-1
Parent: NET-65-0-0-0-0
NetType: Direct Assignment
NameServer: DNS1.CP.MSFT.NET
NameServer: DNS2.CP.MSFT.NET
NameServer: DNS1.TK.MSFT.NET
NameServer: DNS1.DC.MSFT.NET
NameServer: DNS1.SJ.MSFT.NET
Comment:
RegDate: 2001-02-14
Updated: 2002-12-05

TechHandle: ZM23-ARIN
TechName: Microsoft Corporation
TechPhone: +1-425-882-8080
TechEmail: noc@microsoft.com

OrgAbuseHandle: ABUSE231-ARIN
OrgAbuseName: Abuse
OrgAbusePhone: +1-425-882-8080
OrgAbuseEmail: abuse@microsoft.com

OrgNOCHandle: ZM23-ARIN
OrgNOCName: Microsoft Corporation
OrgNOCPhone: +1-425-882-8080
OrgNOCEmail: noc@microsoft.com

OrgTechHandle: MSFTP-ARIN
OrgTechName: MSFT-POC
OrgTechPhone: +1-425-882-8080
OrgTechEmail: iprrms@microsoft.com

3.4.7.2 - Latin American and Caribbean IP Address Regional Registry
The following Registration relates to the Alerts section on:
Most Severe Alert #10

SMB Name Wildcard	Moderate	39730

Search results for: 200.84.76.226

OrgName: Latin American and Caribbean IP address Regional Registry
OrgID: LACNIC
Address: Potosi 1517
City: Montevideo
StateProv:
PostalCode: 11500
Country: UY

ReferralServer: whois://whois.lacnic.net

NetRange: 200.0.0.0 - 200.255.255.255
CIDR: 200.0.0.0/8
NetName: LACNIC-200
NetHandle: NET-200-0-0-0-1
Parent:
NetType: Allocated to LACNIC
NameServer: TINNIE.ARIN.NET
NameServer: NS.LACNIC.ORG
NameServer: NS.DNS.BR
NameServer: NS2.DNS.BR
Comment: This IP address range is under LACNIC responsibility for further
Comment: allocations to users in LACNIC region.
Comment: Please see HTTP://www.lacnic.net/ for further details, or check the
Comment: WHOIS server located at whois.lacnic.net
RegDate: 2002-07-27
Updated: 2003-06-12

TechHandle: LACNIC-ARIN
TechName: LACNIC Hostmaster
TechPhone: (+55) 11 5509-3522
TechEmail: abuse@lacnic.net

OrgTechHandle: LACNIC-ARIN
OrgTechName: LACNIC Hostmaster
OrgTechPhone: (+55) 11 5509-3522
OrgTechEmail: abuse@lacnic.net

3.4.7.3 - University of Haifa

The following Registration relates to the Alerts section on:
Most Sever Alert #9

IRC evil - running XDCC	Moderate	197

Search results for: 132.74.40.10

OrgName: RIPE Network Coordination Centre
OrgID: RIPE
Address: Singel 258
Address: 1016 AB
City: Amsterdam
StateProv:
PostalCode:

Country: NL

ReferralServer: whois://whois.ripe.net

NetRange: 132.74.0.0 - 132.74.255.255
CIDR: 132.74.0.0/16
NetName: ILAN-HAIFA-1
NetHandle: NET-132-74-0-0-1
Parent: NET-132-0-0-0-0
NetType: Allocated to RIPE NCC
NameServer: NS.HAIFA.AC.IL
NameServer: SUNNY.HAIFA.AC.IL
Comment:
RegDate:
Updated: 2003-06-17

AbuseHandle: ZU77-ARIN
AbuseName: University of Haifa
AbusePhone: +972-4-8249249
AbuseEmail: herakel@univ.haifa.ac.il

TechHandle: SS342-ARIN
TechName: Shickman, Simon
TechPhone: +972 2 6584138
TechEmail: simon@cc.huji.ac.il

OrgTechHandle: RIPE-NCC-ARIN
OrgTechName: RIPE NCC Hostmaster
OrgTechPhone: +31 20 535 4444
OrgTechEmail: search-ripe-ncc-not-arin@ripe.net

3.4.7.4 - US WEST Internet Services

The following Registration relates to the Alerts section on:
Most Severe Alert #2

TFTP - External TCP connection to internal TFTP server	High	7
TFTP - External UDP connection to internal TFTP server	High	17220

Search results for: 216.161.210.126

OrgName: U S WEST Internet Services
OrgID: USW
Address: 950 17th Street
Address: Suite 1900
City: Denver
StateProv: CO
PostalCode: 80202
Country: US

NetRange: 216.160.0.0 - 216.161.255.255
CIDR: 216.160.0.0/15
NetName: USW-INTERACT98
NetHandle: NET-216-160-0-0-1
Parent: NET-216-0-0-0-0
NetType: Direct Allocation

NameServer: NS1.USWEST.NET
NameServer: NS2.DNVR.USWEST.NET
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate: 1998-12-16
Updated: 2000-07-18

TechHandle: ZU24-ARIN
TechName: U S WEST ISOps
TechPhone: +1-612-664-4689
TechEmail: abuse@uswest.net

OrgAbuseHandle: QIA2-ARIN
OrgAbuseName: Qwest IP Abuse
OrgAbusePhone: +1-877-886-6515
OrgAbuseEmail: abuse@qwest.net

OrgNOCHandle: QIN-ARIN
OrgNOCName: Qwest IP NOC
OrgNOCPhone: +1-877-886-6515
OrgNOCEmail: support@qwestip.net

OrgTechHandle: QIA-ARIN
OrgTechName: Qwest IP Admin
OrgTechPhone: +1-877-886-6515
OrgTechEmail: ipadmin@qwest.com

3.4.7.5 - Michigan State University

The following Registration relates to the Alerts section on:
Most Severe Alert #4

Spp_HTTP_decode: CGI Null Byte attack detected	Noise	2067

Search results for: 35.10.87.70

Merit Network Inc. MICH-1 (NET-35-0-0-0-1)
35.0.0.0 - 35.255.255.255
Michigan State University MICH-618 (NET-35-8-0-0-1)
35.8.0.0 - 35.10.255.255

3.4.8 - Detailed Correlations and References

General Resources:

Sans Institute. Reading Room
URL: HTTP://www.sans.org/rr (Dec 6, 2003)

[7]SANS Institute. Top 20 Internet Security Vulnerabilities
URL: HTTP://www.sans.org/top20.htm (Dec 6, 2003)

[17]SANS Institute. Intrusion Detection FAQ
URL: http://www.sans.org/resources/idfaq/host_based.php (Dec 6, 2003)

SANS GIAC. Mailing List Archives
URL: HTTP://www.incidents.org/archives/ (Dec 6, 2003)

Insecure.org. News, Vulnerability Info, and NMAP Scanning
URL: HTTP://insecure.org (Dec 6, 2003)

CERT Coordination Center. Vulnerability Info, Incidents, and Fixes
URL: www.cert.org (Dec 6, 2003)

SecurityFocus. Bugtraq Vulnerability Database
URL: www.securityfocus.com

Snort.org T.M. of Sourcefire, Inc. Authors, Brian Caswell and Marty Roesch 2002, 2003
URL: www.snort.org/dl (Dec 6, 2003)

Tcpdump.org. tcpdump and libpcap downloads
URL: www.tcpdump.org (Dec 6, 2003)

Snort.org. Users Manual:
URL: HTTP://www.snort.org/docs/writing_rules/ (Dec 6, 2003)

Snort.org. Signature Database:
URL: HTTP://www/snort.org/snort-db (Dec 6, 2003)

IANA. RFC Port Number Assignments, No Author Cited (Dec 3, 2003)
URL: HTTP://www.iana.org/assignments/port-numbers (Dec 6, 2003)

American Registry for Internet Numbers
URL: HTTP://www.arin.net (Dec 6, 2003)

Trojan Ports: Various Public Information Pages:
URL: HTTP://www.martijnjongen.com/eng/html/portnumbers.htm (Dec 6, 2003)
URL: HTTP://www.austin.rr.com/rrsec/computer_ports.html (Dec 6, 2003)
URL: HTTP://www.ldc.lu.se/security/P2P-list.shtml (Dec 6, 2003)

General Format Ideas:
SansGIAC. Author: Les Gordon, GIAC GCIA Practical (v3.2) Submitted (May 20, 2002)
URL: HTTP://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc (Dec 6, 2003)

[1]General Format Ideas and usage of Todd's published csv.pl script:
SansGIAC. Author: Todd Beardsley, GIAC GCIA Practical (v3.1) Submitted (May 8, 2002)
URL: HTTP://www.giac.org/practical/Tod_Beardsley_GCIA.doc (Dec 6, 2003)

Nimda Worm Analysis:
[2]Cisco Systems. How to Protect Your Network Against the Nimda Virus (Apr 23, 2003)
URL: www.cisco.com/warp/public/63/nimda.shtml (Dec 6, 2003)

[3]SANS Institute. Reading Room. An Overview, Eugene J Aronne, (Oct 8, 2001)
URL: HTTP://www.sans.org/rr/papers/index.php?id=95 (Dec 6, 2003)

[4]CERT Coordination Center. Original release date: (Sep 18, 2001)
Revised: (September 25, 2001) Source: CERT/CC
URL: HTTP://www.cert.org/advisories/CA-2001-26.html (Dec 6, 2003)

TFTP: External connection to Internal TFTP Server Analysis:
[5]FAQS.org. Network Working Group, Author: K. Sollins, MIT, STD: 33 (Jul 1992)
Request For Comments: 1350
URL: HTTP://www.faqs.org/rfcs/rfc1350.html (Dec 6, 2003)

[6]Incidents.org. InternetStormCenter: Result from a search on "port 69", from Neophasis
URL:HTTP://isc.incidents.org/port_details.html?port=69&repax=1&tarax=2&srcax=2&perce

nt=N&days=40 (Dec 6, 2003)

[8]**CGI Null Byte Analysis:**
SansGIAC. Author: Joe Ellis, GIAC GCIA Practical (v3.0) Submitted (May 14, 2002)
URL: HTTP://www.giac.org/practical/Joe_Ellis_GCIA.doc (Dec 6, 2003)

[9]Snort.org. Author: The Snort Core Team, Erek Adams (Apr 9, 2003)
URL: HTTP://www.snort.org/docs/FAQ.txt section 4.12. (Dec 6, 2003)

Possible Trojan server activity Analysis:
[10]SANS Institute. Deconstruction SubSeven, the Trojan Horse of Choice, Author: Jamie Crapanzano (2003)
URL: HTTP://www.sans.org/rr/papers/36/953.pdf (Dec 6, 2003)

[11]SecurityFocus. Bugtraq Message Board, Author: Matt Scarborough (Jun 30, 2001)
URL: HTTP://www.securityfocus.com/archive/75/194328 (Dec 6, 2003)

[12]**High port 65535 tcp - Possible Red Worm traffic Analysis:**
SANS Institute. Adore Worm, Version 0.8, No Author Cited (Apr 12, 2001)
URL: HTTP://www.sans.org/y2k/adore.htm (Dec 6, 2003)

[13]**spp_HTTP_decode: IIS Unicode attack detected Analysis:**
CERT Coordination Center. Vulnerability Note VU#111677, Author: Shawn Hernan (Oct 10, 2000)
URL: HTTP://www.kb.cert.org/vuls/id/111677 (Dec 6, 2003)

[14]Port 55850 tcp - Possible myserver activity - ref. 01:
Insecure.org. Security Incidents Message Board, Author Mike Worman (Oct 23,2000)
URL: HTTP://seclists.org/incidents/2000/Oct/0141.html (Dec 6, 2003)

[15]**IRC evil - running XDCC Analysis:**
Incidents.org. intrusions@incidents.org Mailing list posting LOGS: GIAC GCIA Version 3.3 Practical (Marcus Wu)
URL: HTTP://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00120.html (Dec 6, 2003)

[16]**SMB Name Wildcard Analysis:**
Insecure.org. Message Board, Thread: Intrusion Detection Systems: Re: Source port of Samba Scans? (Mar 11, 2000)
URL: HTTP://seclists.org/lists/ids/2000/Mar/0065.html (Dec 6, 2003)

Last Updated: May 26th, 2017

Upcoming Training

Security Operations Center Summit & Training	**Washington, DC**	**Jun 05, 2017 - Jun 12, 2017**	**Live Event**
SANS Houston 2017	**Houston, TX**	**Jun 05, 2017 - Jun 10, 2017**	**Live Event**
SANS Columbia, MD 2017	**Columbia, MD**	**Jun 26, 2017 - Jul 01, 2017**	**Live Event**
SANS London July 2017	**London, United Kingdom**	**Jul 03, 2017 - Jul 08, 2017**	**Live Event**
SANSFIRE 2017	**Washington, DC**	**Jul 22, 2017 - Jul 29, 2017**	**Live Event**
SANSFIRE 2017 - SEC503: Intrusion Detection In-Depth	**Washington, DC**	**Jul 24, 2017 - Jul 29, 2017**	**vLive**
SANS San Antonio 2017	**San Antonio, TX**	**Aug 06, 2017 - Aug 11, 2017**	**Live Event**
SANS Boston 2017	**Boston, MA**	**Aug 07, 2017 - Aug 12, 2017**	**Live Event**
SANS Adelaide 2017	**Adelaide, Australia**	**Aug 21, 2017 - Aug 26, 2017**	**Live Event**
SANS Virginia Beach 2017	**Virginia Beach, VA**	**Aug 21, 2017 - Sep 01, 2017**	**Live Event**
SANS Network Security 2017	**Las Vegas, NV**	**Sep 10, 2017 - Sep 17, 2017**	**Live Event**
SANS vLive - SEC503: Intrusion Detection In-Depth	**SEC503 - 201709,**	**Sep 11, 2017 - Oct 18, 2017**	**vLive**
SANS Baltimore Fall 2017	**Baltimore, MD**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS London September 2017	**London, United Kingdom**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
Baltimore September 2017 - SEC503: Intrusion Detection In-Depth	**Baltimore, MD**	**Sep 25, 2017 - Sep 30, 2017**	**vLive**
Community SANS Scottsdale SEC503	**Scottsdale, AZ**	**Oct 02, 2017 - Oct 07, 2017**	**Community SANS**
SANS October Singapore 2017	**Singapore, Singapore**	**Oct 09, 2017 - Oct 28, 2017**	**Live Event**
Community SANS Boston SEC503	**Boston, MA**	**Oct 09, 2017 - Oct 14, 2017**	**Community SANS**
Community SANS Ottawa SEC503	**Ottawa, ON**	**Oct 16, 2017 - Oct 21, 2017**	**Community SANS**
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	**San Diego, CA**	**Oct 30, 2017 - Nov 04, 2017**	**vLive**
SANS Seattle 2017	**Seattle, WA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SANS San Diego 2017	**San Diego, CA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SIEM & Tactical Analytics Summit & Training	**Scottsdale, AZ**	**Nov 28, 2017 - Dec 05, 2017**	**Live Event**
SANS Cyber Defense Initiative 2017	**Washington, DC**	**Dec 12, 2017 - Dec 19, 2017**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**