Global Information Assurance Certification Paper

Mark Embrich
Intrusion Detection In Depth
GCIA Practical Assignment, Version 3.0
February 14, 2002

1. Assignment 1: Describe the state of intrusion detection

1.1. Anomaly Detection: The Future of Intrusion Detection or Just Pretty Graphs?

Prior to working on these practical assignments, I had no real interest in looking at anomaly detection because I felt that it would only be useful for extremely big picture things like seeing what port is receiving the most scans. However, after wading eyeball-deep in data for the "Analyze This" scenario for assignment 3, I can see a need for something other than manually looking through each alert entry.

1.2. Definition of Anomaly Detection:

A simplistic definition of anomaly detection is that the detection engine gains some understanding of what is normal use on a network, then alerts on anything that deviates from that norm (Ghosh). The purpose of anomaly detection is to address some shortcomings of signature detection.

1.3. The problem with signature based intrusion detection:

The biggest problem with signature based intrusion detection is that the only way to detect a problem is by predefined rules (or signatures). These rules are based on the content and characteristics of an attack. For example, to alert on Nimda attacks, one of the signatures in the Snort IDS says to watch out for packets coming from the outside to an internal webserver on TCP port 80 where the packet's URI content includes the string "scripts/root.exe?"

Opponents of signature based intrusion detection say that it is simple to change the characteristics or content of the attack to let these attacks sneak by a signature based system. This has been proven correct by the unicode vulnerability, where nearly all intrusion detection systems will alert immediately when the URI content of an http packet contains "\..", the directory traversal signature, attackers encoded the slash into unicode, making the string "%5c.." which would slip right past an IDS looking for "\.." (Rain Forest Puppy).

1.4. Why Anomaly Detection?

Where signature based detection rely on signatures to identify an attack, anomaly detection monitors the activity. Therefore companies that sell anomaly based intrusion detection systems strongly push the idea that these systems are proactive rather than reactive. Entercept Security Technologies offers software for securing servers:

> Entercept's powerful behavioral rules protect servers against attack, even if the attacks used are new and previously unknown. Products based on signatures only cannot detect attacks for which they do not have a signature. By enforcing correct behavior, Entercept can prevent new attacks from succeeding ("How Entercept Protects").

1.5. How does it work?

Sounds too good to be true right? Just install this software and you're secure, past, present and future. I chose Entercept as my example because they were very forward with documenting how they can provide intrusion detection based on behavior. They provide a whitepaper titled: "System Call Interception" in which they explain that their product installs into the kernel area, then acts like a proxy to manage system calls from the user mode to the OS kernel ("System Call Interception").

The thing that you should immediately object to is that Entercept is a host IDS, not a network IDS. That is absolutely correct – Entercept installs on a server and protects that particular server from attacks. The Entercept product is the first that convinced me that anomaly based intrusion detection can work, although with a smaller focus. This type of a product will be a necessary component for an anomaly based network IDS.

1.6. The problem with Anomaly Detection in the network:

The main reason I've always doubted anomaly based network intrusion detection is that network traffic is a complex thing. Marcus Ranum, CEO of Network Flight Recorder, Inc. explains it better than I can:

> The sad reality of networking is that it's very hard to classify "normal" traffic. As networks get sufficiently large, the applications mix they carry becomes so complex that it looks effectively random. An attacker may even generate traffic to generate a distorted model of "normal" so that sooner or later, an attack may look "normal" and get past the IDS. (Ranum)

Beyond Ranum's statement, we also know that network traffic changes as servers and applications come and go. It may be more difficult to define normal network activity than it is to manually go through the signature based alert logs. There is a huge array of devices that need to be protected and a huge array of attacks to watch for.

The ideal way for an anomaly based system to work is to have the system look back at past activity. For example, in the Incidents.org whitepaper on the Nimda worm, "NIMDA Worm/Virus Report – Final," the analysts were able to spot when the Nimda worm became active because of a huge jump in scans of the TCP 80, http port on September 18, 2001. This is where I admit that anomaly based detection is effective for scans or other huge events. However, this is where I begin to doubt the anomaly based method's effectiveness in spotting a small, slick attack. If the anomaly based system is looking through over 500,000 alerts per week (there were 574,184 alerts for the seven days in my "Analyze This" assignment 3), will it pay any attention to a buffer overflow type of attack, where there is no flood? At the network level, I don't think an anomaly based system will be able to spot such an attack. However, a host based anomaly intrusion detection system like Entercept will.

The point here is that anomaly detection can work, but you need to give it a smaller focus than all of the traffic of your network. If, as Ranum says, your network becomes so complex that it looks random, your anomaly detection engine will not be able to discern an attack from legitimate use. However, if you can limit the anomaly detection engine to a single server, the anomaly detection engine can then learn, or be taught what is legitimate use and what isn't. For example, the anomaly detection engine can be taught that permission escalation is a common goal of attackers, then the anomaly detection engine can immediately alert, or prevent any changes to user permissions.

1.7. Complementary technologies:

Therefore, I suggest that anomaly based intrusion detection should not be seen as mutually exclusive of signature based intrusion detection. We should utilize anomaly based detection on important servers and signature based intrusion detection for monitoring the network at large.

Given the smaller focus of a single host will allow an anomaly based system to prevent attacks effectively. The closer the anomaly based system is to the host, the more effective it will be. By putting the anomaly based system on the kernel of the host, there is a much smaller set of attacks to prevent. Rather than watching out for a few dozen ways to cause a buffer overflow, the system can simply prevent the buffer overflow.

To take it even further, anomaly based systems can be built upon the signature based systems, such that the output of the signature based system is analyzed by the anomaly based system. The way acceleration is a function of velocity, anomaly detection should be a function of signature detection. Where on September 18, 2001, the signature based systems started overflowing with attacks to TCP 80, the anomaly based system would immediately raise an alert to the analyst that there is an attack occurring involving these 16 attempts, we've received this many attacks, from these addresses.

SPADE, the Statistical Packet Anomaly Detection Engine, is a Snort plugin produced by Silicon Defense, the same people who produced SnortSnarf. SPADE does exactly what I mention above, it is an anomaly based detection engine that evaluates the output of a signature based IDS (Snort). SPADE assigns values to anomalous packets based inversely on how often they've been seen before, such that a frequently seen packet gets a lower value than a seldom seen packet (Hoagland).

1.8. Works Cited:

Ghosh A., A. Schwartzbard and M. Schatz. "Learning Program Behavior Profiles for Intrusion Detection." 19 February 1999. URL: http://www.usenix.org/publications/library/proceedings/detection99/full_papers/ghosh/ghosh_htm (14 February 2002).

Hoagland, J. and S. Staniford. "README file for the Spade v010818.1." URL: http://www.silicondefense.com/software/spice/spicereadme.htm (14 February 2002).

"How Entercept Protects." 2002. URL: http://www.entercept.com/products/technology/behavioralrules.asp (14 February 2002).

"NIMDA Worm/Virus Report – Final." 3 October 2001. URL: http://www.incidents.org/react/nimda.pdf (28 January 2002).

Rain Forest Puppy. "A look at whisker's anti-IDS tactics." 24 December 1999. URL: http://www.wiretrip.net/rfp/pages/whitepapers/whiskerids.html (14 February 2002).

Ranum, Marcus J. "Intrusion Detection: Challenges and Myths." 1998. URL: http://secinf.net/info/ids/ids_mythe.html (14 February 2002).

"System Call Interception." 2001. URL: http://www.entercept.com/products/entercept/whitepapers/downloads/systemcall.pdf (14 February 2002).

2. Assignment 2: Five Network Detects

2.1. Detect #1: Nimda probe

We received the following Code Red type alerts on 1/25/2002:

GET /scripts/root.exe?/c+dir HTTP/1.0
GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0
GET /d/winnt/system32/cmd.exe?/c+dir HTTP/1.0
GET /scripts/..%5c../winnt/system32/cmd.exe?/c+dir r HTTP/1.0
GET /_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe?/c+dir c+dir HTTP/1.0
GET /_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe?/c+dir c+dir HTTP/1.0
GET /msadc/..%5c../..%5c../..%5c/..55../..c1../../.../winnt/system32/cmd.exe?/c+dir
32/cmd.exe?/c+dir HTTP/1.0
GET /scripts/..%c../winnt/system32/cmd.exe?/c+dir dir HTTP/1.0
GET /scripts/..%c../winnt/system32/cmd.exe?/c+dir dir HTTP/1.0
GET /scripts/..%c../winnt/system32/cmd.exe?/c+dir dir HTTP/1.0
GET /scripts/..%c../winnt/system32/cmd.exe?/c+dir dir HTTP/1.0
GET /scripts/..%5c../winnt/system32/cmd.exe?/c+dir dir HTTP/1.0
GET /scripts/..%5c../winnt/system32/cmd.exe?/c+dir r HTTP/1.0
GET /scripts/..%5c../winnt/system32/cmd.exe?/c+dir c+dir HTTP/1.0
GET /scripts/..%2f../winnt/system32/cmd.exe?/c+dir r HTTP/1.0

This is almost exactly what is written up in the Incidents.org "NIMDA Worm/Virus Report – Final" available at: http://www.incidents.org/react/nimda.pdf. However, there are a few minor differences. The most important difference is that I've captured 15 probes and the Incidents.org paper explains that there are 16 probes – this difference appears to be due to my Snort rules. My Snort rules caught the first packet because it had the string "scripts/root.exe?", however there is no rule for the script "MSADC/root.exe?". Other than that, I also appear to have captured an earlier (or less sophisticated) version, where the encoded backslashes are only encoded once ("% 5" instead of the double encoded "%255").

2.1.1. Source of the Trace.

This trace was captured on Jan. 25, 2002 at 8:47pm PST, on the network owned by my employer.

2.1.2. Detect was generated by:

Snort intrusion detection system, version 1.8.3.
The information above is an extract of the Ethereal output of the snort alerts. It was much easier to compare that extract with the Incidents.org paper to confirm that this is Nimda than it was to try to make that determination from the snort information. However, to remain complete, I'll also include the snort output in Appendix 1 at the end of this practical (it's 5 pages long so I didn't want to include all of it here).

2.1.3. Probability the source address was spoofed:

It is highly unlikely that the source address was spoofed because this is a worm, not an active (human) attacker. The originator of the worm no longer needs to participate, so there is no

reason for him to protect the source address.

2.1.4. Description of the attack:

The Nimda worm is self-replicating code that attacks any version of MS Windows. An infected host gives full administrative permissions to remote attackers as well as scan for more hosts to infect.

This worm caused a huge increase in http probes beginning on September 18, 2001 at 1pm GMT, resulting in over 86,000 unique IP hosts reported to the Internet Storm Center by October 3, 2001 ("NIMDA Worm/Virus Report – Final").

Side effects of the worm's self propagation through network scanning and email sending may result in a denial of service attack.

This particular attack is an example of the attack on MS IIS web server vulnerabilities, see attack mechanisms (below) for more information.

There are three CVE numbers assigned to different vulnerabilities that Nimda takes advantage of:

CVE-2000-0884: "IIS 4.0 and 5.0 allows remote attackers to read documents outside of the web root, and possibly execute arbitrary commands, via malformed URLs that contain UNICODE encoded characters, aka the "Web Server Folder Traversal" vulnerability" (CVE-2000-0884).

CVE-2001-0154: "HTML e-mail feature in Internet Explorer 5.5 and earlier allows attackers to execute attachments by setting an unusual MIME type for the attachment, which Internet Explorer does not process correctly" (CVE-2001-0154).

CVE-2001-0333: "Directory traversal vulnerability in IIS 5.0 and earlier allows remote attackers to execute arbitrary commands by encoding .. (dot dot) and "\" characters twice." (CVE-2001-0333).

2.1.5. Attack mechanism:

There are four attack mechanisms built into Nimda: Microsoft web server vulnerabilities; email attachment; automatic download to web browser; and network share vulnerabilities. I'll discuss each one briefly:

MS IIS web server vulnerabilities:
Nimda looks for various directory traversal vulnerabilities, like the unicode encoded backslashes that are shown in the introduction to this detect. Cert.org provides two papers on these vulnerabilities: VU#111677 (http://www.kb.cert.org/vuls/id/111677) and CA-2001-12 (http://www.cert.org/advisories/CA-2001-12.html). The VU#111677 vulnerability is that IIS server accepts directory traversal characters, such that an attacker can execute arbitrary commands as long as they call the executable file from a relative path that is executable. VU#111677 explains that an attacker that wants to execute a command from the \Winnt directory can do that by sending an URL like: http://www.example.org/scripts/../../../winnt/prog2.exe.

Where the /scripts directory has executable permissions (Hernan, Vulnerability Note).

CA-2001-12 goes a step further, where we find out IIS accepts unicoded characters (not really Microsoft's fault, since this is per RFC2396) such that an attacker can hide the telltale backslash by encoding it as "%5c" where the percent sign denotes that the following is a hex value to be decoded and 5c is the unicode value for the backslash character (Hernan, CERT Advisory).

Email Attachment:
The Nimda worm can also spread through a Microsoft IE vulnerability, where an incorrect MIME header can cause an attachment to automatically launch (Microsoft Security Bulletin (MS01-020)). Combine that vulnerability with the fact that the Nimda worm sends a crafted email that takes advantage of that vulnerability to the entire address book on each computer it infects and you have an extremely effective attack mechanism.

Automatic download to web browser:
The same IE vulnerability as above, however instead of sending the email directly to the target, the Nimda worm changes an infected web server to automatically download itself within a ".eml" file that downloads automatically to IE browsers. This means you can get infected by simply browsing to an infected web server.

Network share vulnerabilities:
Nimda is also able to propagate itself by copying itself to all shares where that user has write permissions. Nimda will attach itself to every executable file it can find on those shares ("NIMDA Worm/Virus Report – Final").

2.1.6. Correlations:

Since Nimda is such a notorious worm, the correlations are extremely strong. In the introduction we can see that it is almost exactly the same content that the Incidents.org paper shows, making this my strongest correlation. The worm is so widespread that correlating the source IP addresses is virtually meaningless, but I have no problem with sharing the source IP addresses that have sent the Nimda attack at us within the last few days (January 25, 2002 through January 29, 2002):

66.60.146.9 (this is the one documented in this paper)
66.2.25.55
66.25.34.229
66.60.146.28
66.168.222.9

In addition to Incidents.org, ARIS (Attack Registry & Intelligence Service) also has an extensive white paper on the Nimda worm. The ARIS paper includes a list of attacks that also correlates to my capture (Mackie pp.6-7).

CERT.org also shows the same attack in their advisory on the Nimda worm (http://www.cert.org/advisories/CA-2001-26.html, under the "System Footprint" section). This may be the best correlation for my capture since their capture matches mine exactly (singly encoded backslashes).

Rick Yuen shows the same attacks on September 18, 2001, the day Nimda started attacking, in his GCIA Practical (http://www.giac.org/practical/Rick_Yuen_GCIA.doc). Unfortunately, since Nimda is relatively new (just over 4 months old at the time of writing) and the most recent GCIA Practicals have not yet been made public, Rick was the only GCIA Practical I could use to correlate my findings against.

There is a multitude of information on Nimda, but the four above are the best correlations that could possibly be found – the top three CIRTs and a predecessor in this course.

2.1.7. Evidence of active targeting:

The Nimda worm is not a live (human) attacker. It uses an algorithm to decide which IP addresses to attack, therefore it does not use active targeting.

According to the CERT.org advisory on the Nimda worm, infected computers begin scanning for other vulnerable computers quite randomly, with a rough formula of:

- 50% of the scans will be to computers where the first two octets of the IP address are the same as the scanning computer.
- 25% of the scans will be to computers where the first octet of the IP address is the same as the scanning computer.
- 25% of the scans will be to a random IP address.

(Danyliw)

2.1.8. Severity:

Criticality: The computers that are the destination of the attacks are production web servers. If any of these computers went down it would result in lost revenue and would be very costly in the confidence our clients have in us. We do not have hot backups yet for these machines, so any down where we need to rebuild the server would take almost an entire business day. These computers are extremely critical to our operations, so I'm going with the highest value of 5.
Criticality = 5

Lethality: The Nimda worm is indeed lethal to any unpatched Microsoft computers, however none of the targeted computers are running a Microsoft operating system. Therefore these attacks cannot do any damage, giving us the lowest possible value of 1.
Lethality = 1

System Countermeasures: The reason these computers would take so long to replace (see Criticality) is that these machines are not default installs, they are hardened by hand, with all unnecessary services removed (not just disabled), all the latest patches applied, and special security software like TCP Wrappers and OpenSSH are being used. Being quite proud of these computers, I'll give us the highest possible value of 5.
System Countermeasures = 5

Network Countermeasures: This attack is coming through a router and a firewall that we manage, but neither piece can distinguish these attacks from permitted HTTP traffic. The edge router is doing packet filtering as a first line of defense, getting rid of the easiest things to block,

basically just to help the firewall not get overwhelmed. The firewall does stateful packet inspection, so picks up where the router's packet filtering does not, however the firewall cannot filter based on content (or at least I don't know how to configure it). In general I would say we have a restrictive firewall, but in this particular case, we have a permissive firewall, since it's letting the attack right though. For this reason, I'm giving us a value of 2.
Network Countermeasures = 2

Calculate Severity:

Severity = (Criticality + Lethality) – (System + Network Countermeasures)
Severity = (5 + 1) – (5 + 2)
Severity = -1

2.1.9. Defensive recommendation:

Since the attacks can do no harm to the targeted computers, this particular attack is useless, meaning our defenses are fine. We do still have one important server that runs a Microsoft operating system connected to the Internet (on another network) that even though the patches are up to date, we've decided to remove it from the Internet. We are in the process of doing that right now, after which we will have no computers directly connected to the Internet that run a Microsoft operating system and all of those computers will be hardened with the latest security patches.

I'll look into what it takes to get our firewall to do such content filtering, then present a Cost/Benefit analysis to my supervisors who will decide whether we will implement it or not. If we can filter by content in either the firewall or router, we'll want to drop packets that are coming in on TCP port 80 with URI content that includes "root.exe" or "cmd.exe".

I'll also look into notifying the Sources of the scans to let them know they've been compromised. We do not want them wasting the bandwidth we are paying for.

2.1.10. Multiple choice question:

Question 1:
All of the following are probes launched by the Nimda worm. Which of the following are examples of the Microsoft web server folder traversal vulnerability (MS00-078):

A. GET /scripts/root.exe?/c+dir
B. GET /c/winnt/system32/cmd.exe?/c+dir
C. GET /d/winnt/system32/cmd.exe?/c+dir
D. GET /scripts/..%5c../winnt/system32/cmd.exe?/c+dir

Answer: D. The key to the folder traversal vulnerability is that you start off in a folder with execute permissions, like /scripts/, then use ../ (represented as ..%5c here) to craft the path such that you launch an executable that you shouldn't be able to (cmd.exe here). The first three choices are probes for backdoors left by Code Red and other prior worms/viruses.

Question 2:

Which of the following ways can Nimda infect a vulnerable computer (you may choose more than one):

A. Automatic launch of an email attachment.
B. Open network shares.
C. Automatic download to your web browser.
D. Microsoft web server folder traversal vulnerability.

Answer: All of the choices are attack mechanisms of Nimda

Question 3:
The following is the output of the payloads in a captured attack. What kind of attack is it? (Choose the best answer):

GET /scripts/root.exe?/c+dir
GET /MSADC/root.exe?/c+dir
GET /c/winnt/system32/cmd.exe?/c+dir
GET /d/winnt/system32/cmd.exe?/c+dir
GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir
GET /_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
GET /_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
GET /msadc/..%255c../..%255c../..%255c/..%c1%1c../
..%c1%1c../..%c1%1c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%%35%63../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%%35c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%25%35%63../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir

A. Code Red II
B. Code Blue
C. Nimda
D. Sadmind

Answer: C. This is a Nimda scan, which checks for backdoors left by Code Red II and Sadmind, Code Blue is just a red herring.

2.2. Detect #2: Proxy scans

These alerts were captured on different days and had different source IP addresses, so they do not appear to be related, but these are all I received in the period of 5 days: January 8, 2002 through January 12, 2002.

[**] SCAN Proxy attempt [**]
01/08-21:58:30.762026 xxx.xxx.xxx.xxx:3612 -> xxx.xxx.xxx.xxx:1080

TCP TTL:49 TOS:0x0 ID:65458 IpLen:20 DgmLen:44 DF
******S* Seq: 0xC5512C34 Ack: 0x0 Win: 0x4000 TcpLen: 24
TCP Options (1) => MSS: 1460

=+=

[**] SCAN Proxy attempt [**]
01/12-05:05:38.042697 xxx.xxx.xxx.xxx:8080 -> xxx.xxx.xxx.xxx:8080
TCP TTL:106 TOS:0x0 ID:45236 IpLen:20 DgmLen:40
******S* Seq: 0x6D96F48E Ack: 0x47115E80 Win: 0xAE2 TcpLen: 20

=+=

2.2.1. Source of trace:

This trace came from the network of my employer.

2.2.2. Detect was generated by:

The detect was generated by a Snort 1.8.3 sensor.
The rules that raised these alerts are:

scan.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN Proxy attempt";flags:S; classtype:attempted-recon; sid:615; rev:1;)

scan.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET 8080 (msg:"SCAN Proxy attempt";flags:S; classtype:attempted-recon; sid:620; rev:1;)

These are very simple rules, simply alerting when an outsider sends SYN packets to an internal computer on TCP ports 1080 and 8080.

2.2.3. Probability the source address was spoofed:

It is unlikely the source address was spoofed since this is a reconnaissance probe. If the information cannot get back to the reconnaissance source, the attackers haven't gained anything.

2.2.4. Description of the attack:

An attacker may be scanning for proxy servers for a couple of reasons. First, if the attacker can use the proxy server to attack a third party, that makes it appear that the attack came from the proxy server. Second, a poorly configured proxy server can become a backdoor for attack from the outside.

These scans are neither of the above mentioned attacks, but these are the precursor, finding out which machines have proxy servers installed and thus could be vulnerable to either of the above. The scans are done by simply sending a SYN packet to a machine on the port that proxy servers listen on, in this case TCP ports 1080 and 8080.

One of my GCIA practical predecessors found quite a few CVE vulnerabilities to list in the description section of a similar detect, but I disagree with his selection of CVE vulnerabilities (Rubio, p.12). The CVE vulnerabilities Reuben Rubio lists are in regards to attacks on proxy servers to run arbitrary code, not related to simply scanning for those proxy servers. While he is correct to warn that these vulnerabilities may follow a proxy server scan, they are not about the scan itself.

There are no CVE vulnerabilities for a proxy port scan because this is expected behavior – when you send a SYN packet to a proxy server, you should get an ACK in return.

2.2.5. Attack mechanism:

The attacker sends a SYN packet to any IP addresses available on TCP ports 1080 and 8080, if an ACK is sent back, it shows the attacker that the responding machine is a proxy server. Once a proxy server is found, the attacker will probe for vulnerabilities that can lead to launching attacks through the proxy server, making it seem that the attacks came from the proxy server (and not the attacker) or an open proxy server can allow an outsider to browse through the victim organization anonymously.

The latter is documented in an article from CNET news: "Hacker had WorldCom in his hands" by Robert Lemos. Lemos writes of Adrian Lamo who has wandered through the corporate networks of industry giants like WorldCom, Microsoft, Excite@Home and Yahoo, all through the vulnerability of an open proxy server (Lemos). Actually, there are other vulnerabilities that Adrian Lamo uses when performing his "security research" (he prefers to be called a security researcher rather than a hacker). The other vulnerabilities and tools Lamo uses are documented in Kevin Poulsen's article for Security Focus (Poulsen).

Joel Scambray and Stuart McClure or "Hacking Exposed" fame write of the former reason in their article: "How hackers cover their tracks" for cnn.com. They explain that more than half of the proxy servers they have scanned (151 out of 282) were open to the world, allowing anyone to use their proxy server to anonymously surf the web (McClure). One can think of these proxy servers like the way the detectives in movies trace phone calls: Each proxy server is a switch which hides the true source of the attack. If any of the proxy servers does not track who goes where, the backtracking trace dies at that point. So it isn't very much like the movies where it's only a matter of time to track back to the source. In this case, the trace can actually disappear.

2.2.6. Correlations:

Like the previous detect, I was able to find a correlation in Rick Yuen's GCIA practical (available at: http://www.giac.org/practical/Rick_Yuen_GCIA.doc). Rick's detect was much more persistent than mine, where his attacker tried 4 attempts on each of two computers on TCP port 1080, mine appear to be two unrelated attempts at different ports.

Lloyd Webb also found the same type of scan, although it is called a Socks scan in his practical (which is available at: http://www.giac.org/practical/Lloyd_Webb_GCIA.doc). By taking a look at the rule that cause the alerts, you can see that this is the same alert (Webb, p.14).

Reuben Rubio is another who found proxy scans while writing his GCIA practical. His detect is

focused on TCP port 8080 unlike the previous two who saw the scans on TCP port 1080. Reuben's GCIA practical is available at:
http://www.giac.org/practical/REUBEN_RUBIO_GCIA.doc (Rubio, p.9).

Dwayne Spriggs also saw scans to TCP port 1080 (referred to as Socks scans), similar to those Lloyd Webb saw. Dwayne's GCIA practical is available at:
http://www.giac.org/practical/Dwayne_Spriggs_GCIA.doc (Spriggs, p.2).

2.2.7. Evidence of active targeting:

The host that was scanned does have a proxy server running on TCP port 1080, but not on 8080. It is likely that this computer was actively targeted, however since I didn't see any following attacks, nor did I see any further probing, I would just as likely consider this a "wrong number," where someone went to the wrong IP address by accident.

2.2.8. Severity:

Criticality: The criticality of this particular computer is quite high since it is also our internal DNS server, but it wouldn't prevent us from doing business if this computer were brought down, so I'm giving this a score of 4.
Criticality = 4.

Lethality: This is only a port scan, which in itself is not lethal. Later attacks that are facilitated by this scan can be lethal, but this particular scan is not, so I'm giving this a score of 2.
Lethality = 2.

System Countermeasures: This is a Windows NT 4.0 server with the latest service packs and patches installed, so it is not the latest operating system, but we have kept up the maintenance, giving this a score of 3.

Network Countermeasures: This server is behind a restrictive firewall and router, but this scan got through, giving this a score of 3.

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
Severity = (4 + 2) – (3 +3)
Severity = 0

2.2.9. Defensive recommendation:

The proxy server ports do not need to be permitted to the Internet, so should be blocked by the edge router and the firewall.
On the router, I'll add an entry to the ingress ACL that prevents incoming packets with a destination TCP port of 1080 or 8080.
On the firewall, I'll do the same.

2.2.10. Multiple choice question:

Question 1:
Which of the following TCP ports are often used by proxy servers? (Select all that apply)

A. 80
B. 1080
C. 3128
D. 8080

Answer: All of them (3128 is squid).

2.3. Detect #3: WEB-ATTACKS id command attempt [False Positive]

[**] WEB-ATTACKS id command attempt [**]
01/30-12:17:06.166082 xxx.xxx.xxx.xxx:50111 -> xxx.xxx.xxx.xxx:80
TCP TTL:126 TOS:0x0 ID:53706 IpLen:20 DgmLen:552 DF
AP Seq: 0xB5E0B92B Ack: 0x32044705 Win: 0x1EF8 TcpLen: 20
65 3E 84 52 91 F0 29 AC EB CF A9 9F CA 6B 7F 6F e>.R..)......k.o
BC 16 3B 49 64 EC A7 A3 49 E1 DF 79 26 66 BB 23 ..;Id...I..y&f.#
<snipped>
=+=

2.3.1. Source of the trace:

This trace came from the network owned by my employer.

2.3.2. Detect was generated by:

The detect was generated by a Snort 1.8.3 sensor.
The rule that tripped the sensor was:

web-attacks.rules:alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-ATTACKS id command attempt"; flags:A+; content:"\;id";nocase; sid:1333; rev:1; classtype:web-application-attack;)

2.3.3. Probability the source address was spoofed:

Unlikely that the source address was spoofed because this is a fingerprinting technique, so the attacker is trying to get a response for the Unix /bin/id command. This attack would be useless if the information was not returned.

2.3.4. Description of the attack:

The purpose of the attack is to run the /bin/id command on a Unix machine to gain information. Here is an example of /bin/id:

```
# /bin/id
uid=0(root) gid=1(other)
```

It will tell you the username, uid, groupname and gid that you are using or (more likely for fingerprinting) you can pass a name as an argument. There are also switches to list all groups that user belongs to and project ID's. Here is an example of me as root using /bin/id to find out about user membrich:

```
# /bin/id
uid=0(root) gid=1(other)
# /bin/id membrich
uid=xxxx(membrich) gid=xxxx(opr)
# /bin/id -ap membrich
uid=xxxx(membrich) gid=xxxx(opr) groups=xxxx(opr) projid=x(default)
```

I've sanitized the information. The first execution was to show which user I was running it as (root). The second to show the default information, the third to show output with both switches.

The rule that caused the alert was written by Zenomorph of cgisecurity.com who provides an explanation for all of the rules in the web-attacks.rules rule set in his paper titled: "Fingerprinting Port 80 Attacks: A look into web server, and web application attack signatures." at: http://www.cgisecurity.com/papers/fingerprint-port80.txt.

However, upon further research into the contents of the payload, we can see that this is a false positive, where the contents contain ";Id" and the rule states that anything matching ";id", regardless of case is a match, so our random ";Id" fits.

2.3.5. Attack mechanism:

The attack mechanism is to take advantage of cgi vulnerabilities to run executables that shouldn't be available to the attacker. Zenomorph lists many ways to attempt to run an executable by using a crafted URL: by using folder traversal ("../"); by unicode encoding the space character ("%20"); by using the pipe character ("|"); by using the semicolon character to signify multiple commands on a single line (";"); by using redirect characters to pass data to an executable ("<", ">"); by using the exclamation point, which is used in Server Side Include (SSI) attacks ("!"); by using PHP insert characters ("<?"); by using back ticks to run Unix commands ("`"); and then combinations of these techniques (Zenomorph).

In this particular attack, the alert is raised because it found the semicolon character with a fingerprinting command following immediately after: ";id." Which means that the URL may be something like: http://host/cgi-bin/cgiscript;id, where the real cgiscript is executed, but immediately followed by the "id" command to do some fingerprinting.

2.3.6. Correlations:

I have not been able to find correlations to this attack. I searched google on the combination of "WEB-ATTACKS" and "snort," none of the returned links resulted in a correlation. I also went through previous GCIA practicals as far as September 2001, no correlations to this attack.

I suspect the reason for the lack of correlative examples is not that these alerts never get raised, but there are better, less obvious ways to do fingerprinting. You can certainly get information on

a user, but you only get information for yourself, or you need to have previous knowledge of a username. I have no experience as an attacker, but I don't see how knowing a user's uid or gid will be any better than knowing the username.

2.3.7. Severity:

Criticality: This capture was actually in the egress direction, where the source is from our network, the destination is an external web server. The criticality of the workstation is quite low, since the attack is not directed at our system at all, however I will give it a 1 rather than 0 because it could cost us in reputation if it became known that attacks come from our networks.
Criticality = 1

Lethality: The lethality is also quite low, since this is a fingerprinting, not a destructive attack, I also doubt how effective this technique would be for fingerprinting. Some of the other techniques in Zenomorph's paper sound more convincing, but I really doubt this particular technique would work very often.
Lethality = 1

System Countermeasures: Since the target is external to us, I cannot tell what system countermeasures are in place. As such, I can't make any assumptions and will expect the worst.
System Countermeasures = 0

Network Countermeasures: Again, the target is external to us, so I cannot tell what network countermeasures are in place. However, an attack of this nature would have to pass through my network countermeasures as both our firewall and edge router perform packet filtering. Unfortunately this is an attack over HTTP, which will pass through both without being halted since neither does content filtering. But since we do have some network countermeasures, I'll go with 2 points for a permissive firewall.
Network Countermeasures = 2

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
Severity = (1 + 1) – (0 + 2)
Severity = 0

2.3.9. Defensive recommendation:

Since this is a false positive, the defenses are fine.
If this were a real attack (and it was directed at one of our public servers), neither our edge router, nor our firewall would have stopped it. If we could filter by content, we can reject any HTTP packets that have the content of ";id" in the URI. However, since the lethality is very low and this is a seldom seen alert, it may not be worth slowing down the throughput of the content filtering devices to check for that string.

2.3.10. Multiple choice test question:

Question 1:
What is the purpose of the "WEB-ATTACKS id command attempt" attack? The snort rule is shown below:

web-attacks.rules:alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-ATTACKS id command attempt"; flags:A+; content:"\;id";nocase; sid:1333; rev:1; classtype:web-application-attack;)

A. Port scan
B. Fingerprinting
C. Session hijack
D. Denial of service

Answer: B, fingerprinting. The rules show you that the alert is raised when a packet is found going to a server with destination port TCP 80, while containing ";id" in the payload – so it is probably not a port scan, denial of service or session hijacking attack. Also the content of "id" should make it a fairly obvious guess of fingerprinting.

Question 2:
For the following snort rule, which of the following will cause the alert to be raised? (select all that apply)

web-attacks.rules:alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-ATTACKS id command attempt"; flags:A+; content:"\;id";nocase; sid:1333; rev:1; classtype:web-application-attack;)

A. TCP, Port 80, flags = AP, content includes "id"
B. TCP, Port 80, flags = S, content includes "\;id"
C. TCP, Port 80, flags = 12UAPRSF, content includes "\;id"
D. TCP, Port 80, flags = AP, content includes ";ID"

Answer: C and D. A is not correct because the content needs to include the semicolon. B is not correct because the flags must include the ACK flag. C is correct, the extra flags won't prevent this alert from going off. D is the tricky one. It looks like the rule requires a backslash before the ";id" content, but that's not the case. The backslash is there to escape the semicolon. Also the nocase option is there so ID is the same as id.

2.4. Detect #4: TCP Scan for sshd

inetnum: 194.94.24.0 - 194.94.27.255
netname: HSB
descr: Hochschule Bremen
descr: Rechenzentrum
country: DE

Jan 27 16:02:41 194.94.27.181:22 -> a.b.c.20:22 SYN ******S*
Jan 27 16:02:41 194.94.27.181:22 -> a.b.c.27:22 SYN ******S*
<snipped for brevity>
Jan 27 16:02:41 194.94.27.181:22 -> a.b.f.252:22 SYN ******S*
Jan 27 16:02:41 hostl sshd[4603]: Did not receive identification string from 194.94.27.181.

2.4.1. Source of trace:

This trace was found on the incidents.org website at:
http://www.incidents.org/archives/intrusions/msg03561.html.
It was posted by Laurie Zirkle of Virginia Tech on January 29, 2002 and is only a portion of the post.

2.4.2. Detect was generated by:

Although it is not noted in the posting, the scan packets (all except the last line) are in the format of snort portscan captures. The last message appears to be from a UNIX message log.

The snort portscan captures are in the format of:
<Date> <Time> <SourceIP>:<SourcePort> -> <DestIP>:<DestPort> <type of scan> <TCP flags>
Where for the first line:
Jan 27 16:02:41 194.94.27.181:22 -> a.b.c.20:22 SYN ******S*
Date = Jan 27
Time = 16:02:41
SourceIP = 194.94.27.181
SourcePort = 22
DestIP = a.b.c.20
DestPort = 22
Type of scan = SYN scan
TCP flags = ******S* (or only the SYN flag is set)

The UNIX message log format is:
<Date> <Time> <hostname> <process>[<processID>]: <log message>.
Where for the last line:
Jan 27 16:02:41 hostl sshd[4603]: Did not receive identification string from 194.94.27.181.
Date = Jan 27
Time = 16:02:41
Hostname = hostl
Process = sshd
ProcessID = 4603
Log message = Did not receive identification string from 194.94.27.181.

2.4.3. Probability the source address was spoofed:

Very unlikely, since the scan relies on receipt of an ACK packet from the destination IP address. If the address was spoofed, the ACK would never reach the attacker.

2.4.4. Description of the attack:

An attacker ran a simple TCP scan of four class C networks to try to find computers that respond to TCP port 22, which is reserved for ssh connections. The last line of the capture, the UNIX message log tells us that the attacker did indeed find a computer that is responding to TCP port

22 and is running the ssh daemon (sshd).

Unfortunately, this is where the capture ends, we do not know whether the attacker attempted to exploit any of the ssh vulnerabilities. The scanning for computers that are running sshd is not in itself a vulnerability, however a search of the CVE and CERT databases produces dozens of known ssh vulnerabilities.

The one that is most prevalent is: CVE-2001-0144:
"CORE SDI SSH1 CRC-32 compensation attack detector allows remote attackers to execute arbitrary commands on an SSH server or client via an integer overflow." (CVE-2001-0144).

David Dittrich from the University of Washington performed an analysis of this attack in his white paper "Analysis of SSH crc32 compensation attack detector exploit." Dittrich shows that an attack originating from the Netherlands was able to exploit the vulnerability to install a trojan horse on a computer running a vulnerable version of sshd (Dittrich).

2.4.5. Attack mechanism:

This is a TCP port scan, where a single SYN packet is sent from the attacker's computer to a set target. The target can be a single IP address or a group of IP addresses, the latter appears to be the case here. The extremely tight time grouping (all of the scans occurred within the same second) tells us the entire scan was probably the result of a single command.

2.4.6. Correlations:

The ssh vulnerabilities are relatively new, so I couldn't find direct correlations to scans for ssh in previous practicals. However, I was able to find many correlations elsewhere.

John Sage, a frequent contributor to the incidents.org intrusions forum, captured a scan for computers running ssh in a message to PlusVentures, LLC (where the attack came from). This message was posted on December 21, 2001, "port 22 ssh probe from your host 209.61.148.63 - dev.plussize.com" available at: http://www.incidents.org/archives/intrusions/msg03009.html (Sage).
On December 4, 2001, Russell Fulton of the University of Auckland, New Zealand posted suspicious, slow scans to his computers on port 22 in the message titled "slowish ssh scan from 149.69.85.65", available at: http://archives.neohapsis.com/archives/incidents/2001-12/0047.html (Fulton).
Fulton's message brought out correlations by:
Andreas Ostling in his reply to Fulton on December 5, 2001, available at:
http://archives.neohapsis.com/archives/incidents/2001-12/0060.html
Glenn Larratt in his reply on December 5, 2001, available at:
http://archives.neohapsis.com/archives/incidents/2001-12/0056.html
Jim Watt in his reply on December 5, 2001, available at:
http://archives.neohapsis.com/archives/incidents/2001-12/0058.html

2.4.7. Evidence of active targeting:

Due to the number of computers scanned, it is definitely a specific scan. The attackers definitely

wanted to know which of a large number of computers was running sshd.

2.4.8. Severity:

Criticality: The criticality is unknown, since the scanned computers are not a part of my network, I don't know the function of each machine. However, since there were a large number of computers that were scanned, it's safe to guess that at least one of those computers is absolutely critical.
Criticality = 5

Lethality: The lethality of the scan is quite low, but given that this scan may lead to the exploit Dittrich exposed, I have to give this at least a middle value for lethality.
Lethality = 3

System Countermeasures: The system countermeasures are unknown, but we do know that one computer did respond. Getting a response is half of the battle here, the other half is whether that computer is running a version of ssh that is vulnerable to the exploit the attacker wishes to utilize. We do not have that information, but we do know that one computer did respond, so I'll give a median value.
System Countermeasures = 3

Network Countermeasures: A look through Zirkle's posted logs tells us that there are multiple IDS systems running on her network, at least including snort, portsentry, and UNIX logging. So although I don't know this for a fact, it's safe to guess that the network security is quite extensive, almost certainly including a restrictive firewall and routers. However, connecting to an sshd server from an external network is probably expected behavior, so connection attempts to computers running an sshd daemon will continue to be let through.
Network Countermeasures = 4

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
Severity = (5 + 3) – (3 + 4)
Severity = 1

Note: This is a pessimistic value, since I do not have all of the information. Important factors that can change this value are:
The functions of the scanned computers – I don't know if they are critical or not, so had to guess extremely critical.
Whether or not any attack followed – The scans alone are not lethal, the lethality value is inflated on the assumption that the scan is facilitating a later attack.
The versions of sshd is very important to the System Countermeasures value. It's highly likely that the computers that do run sshd are running versions that are not vulnerable to most (if not all) of the known ssh exploits.
Lastly, I don't know whether the computers that run sshd should be available to the Internet. If they don't need to be available to the Internet, TCP port 22 can be blocked entirely, giving a higher value for Network Countermeasures.

2.4.9. Defensive recommendations:

Most importantly, upgrade or patch any versions of sshd that are vulnerable to known exploits. For example, the exploit in Dittrich's paper has been fixed since version 2.3.0 of OpenSSH (OpenSSH Security). The newest version, 3.0.2 also addresses other vulnerabilities.

Other recommendations depend on whether external computers should be able to connect to those computers via ssh. If the scanned computers do not need to accept ssh connections from external computers, we can block traffic on TCP port 22 at the routers and firewalls, preventing even the scans from getting to the internal computers.

2.4.10. Multiple choice question:

Question #1:
Scans to the TCP port 22 are searching for what service?

A. FTP
B. Telnet
C. SSH
D. DNS

Answer: C (FTP is 21, Telnet is 23, DNS is 53)

2.5. ICMP redirect

Nov 10 00:13:37 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
Nov 10 00:17:08 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

2.5.1. Source of trace:

I originally was interested in a recurrence of this attack that occurred on January 23, 2002, but on that date, only a single ICMP redirect was sent, which makes a weak argument for a denial of service attack. However that detect contained a notation that there were more ICMP redirects that occurred on November 10, 2001, so I looked up that date and found the above posted.

This trace was found at the incidents.org website at:
http://www.incidents.org/archives/intrusions/msg02420.html
It was posted on November 12, 2001 by Laurie Zirkle of Virginia Tech.
The January 23, 2002 trace was found at the incidents.org website at:
http://www.incidents.org/archives/intrusions/msg03504.html.
It was posted on January 24, 2002 by Laurie Zirkle of Virginia Tech.

The above capture is a small sample of what Zirkle captured on November 10, 2001, even a small sample of what she posted. Zirkle received thousands of these ICMP redirects per hour for a 15 hour period. The full posting is in Appendix 2.

2.5.2. Detect was generated by:

The detect was generated by snort, the presentation is in the snort syslog format where the fields are:
<Date> <Time> <Hostname> <Process>: [ID <LOG_PID> <LOG_AUTH>] <Snort alert message>: <SourceIP> -> <DestIP>
For example, for the capture record:
Nov 10 00:13:37 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
Date = Nov 10
Time = 00:13:37
Hostname = hostmi
Process = snort
LOG_PID = 992550
LOG_AUTH = auth.alert
Snort alert message = ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]
SourceIP = 194.225.65.1
DestIP = z.y.x.98

This is the snort rule that caused the alert:
icmp.rules:alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP redirect net";itype:5;icode:0; reference:arachnids,199; reference:cve,CVE-1999-0265; classtype:bad-unknown; sid:473; rev:1;)

2.5.3. Probability the source address was spoofed:

Since this is a denial of service attack, there is a good chance the source address was spoofed. The attacker does not need any response from the victim in a denial of service attack.

2.5.4. Description of attack:

This attack is comprised of ICMP redirect packets from a single source address to a single destination address. There were thousands of ICMP redirect packets received per hour for a stretch of 15 hours on the date of November 10, 2001.

There are a couple of CVE vulnerabilities that can apply to this attack:

CVE-1999-0265: ICMP redirect messages may crash or lock up a host.
However this is related to embedded controllers, not computers. The ISS advisory that was the precursor to the CVE item states that “embedded controllers are found in a wide variety of automation equipment, manufacturing equipment, HVAC
(Heating, Ventilation, and Air Conditioning) equipment, and medical
equipment” (ICMP Redirects Against Embedded Controllers).

CAN-1999-1254: ** CANDIDATE (under review) ** Windows 95, 98, and NT 4.0 allow remote attackers to cause a denial of service by spoofing ICMP redirect messages from a router, which causes Windows to change its routing tables.
Delmore, a Russian hacker found he could cause Windows computers to change their routing tables by sending a storm of ICMP redirect messages where the source address is spoofed such that it appears to be coming from the victim’s router. The routing change and ICMP redirect

storm resulted in an unresponsive computer or degraded performance. This is the WinFreez.c code that can be found at the PacketStorm web site: http://packetstormsecurity.nl/DoS/winfreez.c. Delmore has also ported this code to Solaris on sparc architecture: http://packetstormsecurity.nl/9903-exploits/Winfreeze-sparc.c

A whois lookup of the attacker's source IP address is the thing that got me the most interested in checking out this detect:

```
Trying whois -h whois.ripe.net 194.225.65.1
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      194.225.65.0 - 194.225.65.255
netname:      NRCGEBNET
descr:        National Reasearch Centre for Genetic Enineering and
descr:        Biotechnology. 15, Shafiel St, Ghods Ave, Tehran IRAN
country:      IR
admin-c:      BG4-RIPE
tech-c:       BG4-RIPE
status:       ASSIGNED PA
notify:       saeed@rose.ipm.ac.ir
mnt-by:       RIPE-NCC-NONE-MNT
changed:      GeertJan.deGroot@ripe.net 19950106
changed:      saeed@rose.ipm.ac.ir 19970312
changed:      saeed@rose.ipm.ac.ir 19971004
source:       RIPE

person:       Bahram Goliaei
address:      National Research Centre for Genetic Engineering and Biotechnology
address:      15, Shafiel Street, Ghods Ave, Tehran IRAN
address:      P O box 14155, 6343 Tehran, IRAN
phone:        +98 21 6419738
fax-no:       +98 21 6419834
e-mail:       goliaei@irearn.bitnet
nic-hdl:      BG4-RIPE
changed:      GeertJan.deGroot@ripe.net 19950106
source:       RIPE
```

The National Research Centre for Genetic Engineering and Biotechnology in Tehran, Iran… that's very interesting. Given that Zirkle is from Virginia Tech, I'd say there's a good chance that this Iranian organization would find some value in the information stored in the Virginia Tech computers. Unfortunately, my analysis leads me to the conclusion that the source address is probably spoofed.

2.5.5. Attack mechanism:

The purpose of an ICMP redirect message is to make routing faster in a local network.
Given that router1 is the default gateway for computer1.
If computer1 wants to communicate to computer2 (and computer2 is on a different network), it will send the packet to router1.
If router1 finds the next hop to computer2 is router2, where router2 must be on the same network as router1 and computer1, router1 will forward the packet to router2 and send an ICMP redirect message to computer1 telling it to send future packets destined for computer2 through router2 directly, thereby adding a route to computer1. (Stevens, pp.119-123)

The important things to note are that:
Only routers can issue ICMP redirects.
Computer1 and router2 must be on the same network for the ICMP redirect to be issued.

We know that the source device and the destination device are on different networks, therefore this ICMP redirect message is illegal. Also, an ICMP redirect message to a computer on the source network would come from the source-side router, not from anything in the destination network.

The key to the WinFreez.c denial of service attack is that a storm of ICMP redirect packets are generated, with the source address spoofed such that the ICMP redirects appear to be coming from the victim's router. The denial of service comes by causing the victim computer to add routes to random IP addresses, using the victim's own IP address as the gateway. The goal is to make the victim computer cause a denial of service against itself (Northcutt, p.63).

Given those rules, we see that this attack does not follow the modus operandi of CAN-1999-1254, where the goal is to add false routes to a Windows computer causing a denial of service. The difference is that WinFreez.c spoofs the victim computer's router address, thus appearing to come from a valid source. In this attack, the source is from the National Research Centre for Genetic Engineering and Biotechnology in Tehran, Iran, so no computer should add routes because the victim computer and the router that issues the ICMP redirect are on different networks. Therefore either this is a very poor attack, which will never work, or there was some other intention.

If there were some other intention, I'm not sure what it is, since you wouldn't send thousands of packets over 15 hours if the goal was fingerprinting or any other kind of reconnaissance, assuming you get some distinctive type of response to an ICMP redirect.

Since the number of packets sent per hour varies widely (between 512 to 4134), I don't think it was a plain flooding type of denial of service attack.

Since this is an ICMP packet, there should be no type of error message sent back to the source, so it is not like a smurf attack, where the victim computer is really an amplifier to send messages to the source address.

Lastly, it also cannot be a case where the intention was to add a route to the victim computer such that all of it's traffic flows through the Iranian router, allowing the attacker to capture all of the victim's traffic. Since even if a route is added to the victim computer, the victim computer and the Iranian router are on two different networks, making it impossible to reach the Iranian router.

For example:

Before:
Victim's IP address is z.y.x.98
Victim's default gateway is z.y.x.1

ICMP redirect arrives, adds route saying change default gateway to 194.225.65.1, Victim makes the change

After:
Victim's IP address is z.y.x.98
Victim's default gateway is 194.225.65.1

Victim will not be able to connect to anything outside of the z.y.x. network, because it cannot reach it's default gateway.

Which takes us back to the first problem, where we know from the rules of ICMP redirects that the redirect cannot come from the remote router, nor can it come from a router on a different network than the victim computer.

Which leaves the only options for the attack at: a misconfigured router (unintentional), or something else that I don't see.

2.5.6. Correlations:

Zirkle also captured similar suspicious traffic on:
January 23, 2002 (http://www.incidents.org/archives/intrusions/msg03504.html)
December 31, 2001 (http://www.incidents.org/archives/intrusions/msg03098.html)
November 8, 2001 (http://www.incidents.org/archives/intrusions/msg02408.html)
November 7, 2001 (http://www.incidents.org/archives/intrusions/msg02392.html)
I was only able to find a few others that reported receiving similar traffic:
Brian Carpio captured similar packets on May 7, 2001
(http://archive.lug.boulder.co.us/bymonth/2001.05/msg00124.html)
I couldn't find any others, which is not surprising since I doubt that these packets are a real attack.

2.5.7. Evidence of active targeting:

The traffic is very specific, with each of the attacks captured by Zirkle involving a single source address and a single destination address. In the November 7, 8, and 10 attacks, the destination was z.y.x.98. In the December 31 and January 23 attacks, the destination address was z.y.x.34. This tells us the targeting is specific, but I need more information to know whether this was active targeting or the ICMP redirects were a response. The information necessary to make a real judgment would be the content of the ICMP redirects and possibly the traffic going in the other direction. The content would tell us what routes the Iranian router would have the victim computer add. The traffic going in the other direction would tell us if the ICMP redirects are caused by an error on the victim's end and if there is anything valuable gained by the attack.

From the information we have: thousands of packets sent in bursts for a period as long as 15 consecutive hours, I would say these packets are not a result of active targeting, more likely a response to some stimuli.

2.5.8. Severity:

Criticality: I do not have enough information to give a real value for the criticality of the z.y.x.98 computer. To make that judgment, I would need to know the purpose of the computer and what backup measures are available. Therefore I'll give a median value.
Criticality = 3

Lethality: Since this traffic does not seem to be a real attack, I will have to give this a low value.
Lethality = 1

System Countermeasures: I do not have enough information to give a real value here either, but since I cannot determine how the traffic can be performing a successful attack, I cannot say how the system countermeasures can be improved. Therefore I need to give the highest value.
System Countermeasures = 5

Network Countermeasures: This is another area where I do not have enough information. I don't know where the snort sensor lies, so I cannot assume that Zirkle's routers and firewalls are letting the ICMP redirects into her network. In fact, Cisco makes it well known that they encourage all routers to block ICMP redirects, so I'd say it's more than likely that these packets are blocked at Zirkle's routers and firewalls. However, I do not know that for a fact, so I'll give a median value.
Network Countermeasures = 3

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
Severity = (3 + 1) – (5 + 3)
Severity = -4

2.5.9. Defensive recommendation:

The most obvious and most effective defensive step would be to block ICMP redirect messages at all border routers.
On a cisco router, the following line on the ACL applied to the incoming direction on the Internet interface will block ICMP redirect messages coming from the Internet:

access-list 101 deny icmp any any redirect

You would also want to do the same for the outgoing direction, since you don't want to be the source of the attack either.

2.5.10. Multiple choice question:

Question #1:
Which of the following conditions must be true for a computer to add a route to its routing table

upon receiving an ICMP redirect message (choose all that apply):

A. The redirect must come from the current router for that destination.
B: The redirect cannot tell the host to use itself as the router.
C: The new router must be on a directly connected network.
D: The new router must be running the OSPF routing protocol.

Answer: A, B, C (These are direct quotes from TCP/IP Illustrated, Vol.1 by Richard Stevens, page 123).

Question #2:
What is type of ICMP message is an ICMP redirect message?

A. 1
B. 3
C. 5
D. 8

Answer: C (Type 1 is undefined, type 3 is "Destination unreachable", type 5 is "Redirect", type 8 is "Echo")

3. Assignment 3: "Analyze This" Scenario

3.1. Executive summary:

Stephen Northcutt made a statement that Christmas Day is a great day to attack computers. No one is actively monitoring them, yet they're up and running. Therefore, I chose to do my analysis on the days surrounding Christmas Day. However, I didn't want the judges to think I chose those dates because they happen to have much less alerts, scans and OOS data than the typical week. To prove that wasn't the case, I added on two more days, causing my analysis to include seven consecutive days instead of the required five.

Those seven days comprised of the period of December 20, 2001 through December 26, 2001. The alerts included 769,499 total alerts, made up of 195,341 scans and 574,158 other alerts. I removed the scan records from the alerts, since there is a separate analysis for the scans later. Of the 574,158 alerts, there were 149 unique alert types that occurred. The top 10 alert types make up 85.43% of the total number of alerts, therefore a disproportionate amount of research has been done on the more frequent alerts.

The scans included 2,627,761 records, 781,305 UDP scans and 1,846,456 TCP scans.

There were 8,390 Out of Spec. packets, most of which were scans or probes.

In investigating the alerts we found a few compromised computers that should be taken off the network and cleaned immediately.

The files selected for analysis include:

Alerts	Scans	OOS
alert.011220.gz	scans.011220.gz	oos_Dec.20.2001.gz
alert.011221.gz	scans.011221.gz	oos_Dec.21.2001.gz
alert.011222.gz	scans.011222.gz	oos_Dec.22.2001.gz
alert.011223.gz	scans.011223.gz	oos_Dec.23.2001.gz
alert.011224.gz	scans.011224.gz	oos_Dec.24.2001.gz
alert.011225.gz	scans.011225.gz	oos_Dec.25.2001.gz
alert.011226.gz	scans.011226.gz	oos_Dec.26.2001.gz

3.2. Alerts analysis:

3.2.1. Description of alerts:

Snort Attacks, grouped by type of attack, sorted by the number of occurrences:

alert	count
Tiny Fragments - Possible Hostile Activity	75653
MISC traceroute	66898
Watchlist 000220 IL-ISDNNET-990517	62991
MISC Large UDP Packet	59073
MISC source port 53 to <1024	51767
ICMP Source Quench	51202
CS WEBSERVER - external web traffic	49293
INFO MSN IM Chat data	28246
WEB-MISC prefix-get //	24069
SYN-FIN scan!	21297
ICMP Echo Request BSDtype	12509
ICMP Destination Unreachable (Host Unreachable)	8994
SCAN Proxy attempt	7305
ICMP Destination Unreachable (Communication Admini	6935
Queso fingerprint	5796
ICMP Fragment Reassembly Time Exceeded	4923
ICMP Echo Request Nmap or HPING2	4031
BACKDOOR NetMetro File List	3586
ICMP Echo Request L3retriever Ping	2557
SMB Name Wildcard	2257
Watchlist 000222 NET-NCFC	2141
ICMP Destination Unreachable (Protocol Unreachable	1780
INFO FTP anonymous FTP	1543
External RPC call	1478
connect to 515 from outside	1380
BACKDOOR NetMetro Incoming Traffic	1098
WEB-MISC Attempt to execute cmd	1041

WEB-MISC 403 Forbidden	1019
SMTP relaying denied	865
EXPLOIT x86 NOOP	789
INFO Inbound GNUTella Connect accept	736
ICMP Echo Request Windows	729
spp_http_decode: IIS Unicode attack detected	695
ICMP traceroute	683
TCP SRC and DST outside network	604
Incomplete Packet Fragments Discarded	601
Null scan!	594
ICMP Echo Request Sun Solaris	560
FTP DoS ftpd globbing	485
INFO Napster Client Data	451
INFO Possible IRC Access	432
TELNET login incorrect	418
INFO Outbound GNUTella Connect accept	361
Port 55850 tcp - Possible myserver activity - ref.	323
ICMP Echo Request CyberKit 2.2 Windows	306
NMAP TCP ping!	292
WEB-MISC http directory traversal	292
INFO - Possible Squid Scan	279
CS WEBSERVER - external ftp traffic	263
WEB-MISC count.cgi access	220
High port 65535 tcp - possible Red Worm - traffic	164
WEB-IIS view source via translate header	139
High port 65535 udp - possible Red Worm - traffic	138
ICMP Destination Unreachable (Fragmentation Needed	134
SCAN FIN	128
WEB-CGI scriptalias access	124
Possible trojan server activity	118
MISC Large ICMP Packet	88
WEB-IIS _vti_inf access	84
WEB-FRONTPAGE _vti_rpc access	83
INFO Inbound GNUTella Connect request	71
connect to 515 from inside	69
beetle.ucs	65
TFTP - Internal TCP connection to external tftp se	65
WEB-CGI redirect access	63
WEB-IIS Unauthorized IP Access Attempt	62
SUNRPC highport access!	61
INFO - Web Cmd completed	46
WEB-CGI rsh access	43
TELNET access	32
SCAN Synscan Portscan ID 19104	30
WEB-CGI formmail access	29
Virus - Possible scr Worm	28
WEB-MISC compaq nsight directory traversal	27

DDOS shaft client to handler	25
SNMP public access	25
WEB-CGI csh access	19
X11 outgoing	19
EXPLOIT x86 setuid 0	16
Port 55850 udp - Possible myserver activity - ref.	16
Attempted Sun RPC high port access	15
EXPLOIT x86 NOPS	15
EXPLOIT x86 setgid 0	15
ICMP redirect (Host)	15
WEB-FRONTPAGE shtml.exe	14
WEB-FRONTPAGE fpcount.exe access	13
SMTP chameleon overflow	12
DNS zone transfer	11
ICMP Destination Unreachable (Network Unreachable)	11
Virus - Possible pif Worm	10
WEB-IIS File permission canonicalization	8
WEB-CGI archie access	7
WEB-FRONTPAGE posting	7
INFO - Web Dir listing	6
INFO napster login	6
MISC PCAnywhere Startup	6
spp_http_decode: CGI Null Byte attack detected	6
TFTP - External UDP connection to internal tftp se	6
WEB-FRONTPAGE shtml.dll	6
WEB-MISC Lotus Domino directory traversal	6
IDS475/web-iis_web-webdav-propfind	5
RFB - Possible WinVNC - 010708-1	5
WEB-CGI finger access	5
DDOS mstream handler to client	4
EXPLOIT NTPDX buffer overflow	4
HelpDesk MY.NET.83.197 to External FTP	4
IDS50/trojan_trojan-active-subseven	4
TFTP - External TCP connection to internal tftp se	4
TFTP - Internal UDP connection to external tftp se	4
Virus - Possible NAIL Worm	4
WEB-MISC /....	4
MISC solaris 2.5 backdoor attempt	3
Virus - Possible MyRomeo Worm	3
Virus - Possible shs Worm	3
WEB-CGI survey.cgi access	3
WEB-IIS .cnf access	3
EXPLOIT x86 stealth noop	2
External FTP to HelpDesk MY.NET.70.49	2
External FTP to HelpDesk MY.NET.70.50	2
FTP CWD / - possible warez site	2
INFO - Web Command Error	2

Probable NMAP fingerprint attempt	2
RPC portmap request rstatd	2
RPC tcp traffic contains bin_sh	2
SCAN XMAS	2
WEB-CGI glimpse access	2
WEB-CGI ksh access	2
WEB-CGI tsch access	2
WEB-CGI w3-msql access	2
WEB-IIS MSProxy access	2
WEB-MISC guestbook.cgi access	2
WEB-MISC ICQ Webfront HTTP DOS	2
x86 NOOP - unicode BUFFER OVERFLOW ATTACK	2
NULL	1
External FTP to HelpDesk MY.NET.53.29	1
External FTP to HelpDesk MY.NET.83.197	1
FTP passwd attempt	1
FTP RETR 1MB possible warez site	1
ICMP Parameter Problem (Unspecified Error)	1
ICMP Photuris (Undefined Code!)	1
ICMP Redirect (Undefined Code!)	1
ICMP Reserved for Security (Type 19) (Undefined Co	1
ICMP SRC and DST outside network	1
ICMP Timestamp Request	1
SCAN - wayboard request - allows reading of arbitr	1
TCP SMTP Source Port traffic	1
Traffic from port 53 to port 123	1
WEB-FRONTPAGE form_results access	1
WEB-MISC handler access	1

There are 149 unique alerts that were raised in this seven day period. Describing each of those alerts would be very long in preparation and make it very difficult to read, I'm limiting the descriptions to the top 100 alerts, which leaves out nearly all of the alerts that have five or fewer occurrences.

I have also done further investigation on five of the alerts I thought were the most dangerous, which can be found in the Appendix section:
Appendix 3: Tiny Fragments (#1 of 149)
Appendix 4: Watchlist 000220 IL-ISDNNET-990517 (#3 of 149)
Appendix 5: BACKDOOR NetMetro File List (#17 of 149)
Appendix 6: connect to 515 from outside (#24 of 149)
Appendix 7: EXPLOIT x86 NOOP (#29 of 149)
Each of these investigations includes looking up the registration information on at least one external source address.

3.2.1.1. Tiny Fragments - Possible Hostile Activity 75,653 alerts 13.1763% of all alerts

IP fragments are created when a packet to be transmitted is larger than the MTU (Maximum Transmission Unit) of the device (which can be a computer, router, or other networking equipment). For example, a computer may receive a packet of 1500 bytes, then try to forward it through a modem. If that modem has an MTU setting of 256 bytes, the packet will be fragmented into packets of 256 bytes each.

The malicious use of fragments would include fragmenting the IP header to get it past a firewall, or various denial of service attacks like Teardrop or Jolt.

I did further research on this alert, found that most of these alerts were more likely caused by a piece of defective equipment than an attack. It does look like some of the "Tiny Fragments…" alerts were due to a finger printing session. For the detailed research, refer to Appendix A: Tiny Fragments Alert Research

3.2.1.2. MISC traceroute 66,898 alerts 11.6515% of all alerts

The ICMP traceroute is one of the most basic and simple ways to get information on a network. Traceroute sends an ICMP packet with a TTL of 1 to the destination. The first device that receives the packet decrements the TTL, causing an ICMP Time Exceeded packet to be sent back.
Traceroute then sends an ICMP packet with a TTL of 2 to the destination. The second device that receives the packet sends back an ICMP Time Exceeded packet back to the source. Traceroute continues this process until it reaches the destination. The result is that you get back a map of the path taken from the source to the destination and the time it took for each hop. Traceroute can be used to mark routers for a denial of service attack.

3.2.1.3. Watchlist 000220 IL-ISDNNET-990517 62,991 alerts 10.9710% of all alerts

The Watchlist is a custom rule. It appears that traffic from this Israeli ISP has been responsible for many alerts in the past so the security personnel have created a rule to watch any traffic to or from that ISP.

3.2.1.4. MISC Large UDP Packet 59,073 alerts 10.2886% of all alerts

In my snort rules, this alert is raised when a UDP packet larger than 4000 bytes comes into the internal network from an external address. It is unusual to have packets greater than 4000 bytes, so these should be monitored.

3.2.1.5. MISC source port 53 to <1024 51,767 alerts 9.0162% of all alerts

This alert is raised when a TCP packet from an external host has a source port of 53 and the destination is an internal address with a port number less than 1024. TCP port 53 is a DNS zone transfer, which should not be coming from an external address on a well known port.

3.2.1.6. ICMP Source Quench 51,202 alerts 8.9178% of all alerts

An ICMP source quench error message is sent when a device is overwhelmed with incoming

traffic. This may be an early signal that a denial of service attack is occurring.

3.2.1.7. CS WEBSERVER - external web traffic 49,293 alerts 8.5853% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic to a certain set of external web servers is seen. I do not know what the "CS" part stands for, which would probably explain the what type of external web servers are watched by this rule.

3.2.1.8. INFO MSN IM Chat data 28,246 alerts 4.9196% of all alerts

This alert is raised when traffic from an internal address is headed to an external address with a destination port of TCP 1863, which is reserved for Microsoft's Instant Messenger chat software.

3.2.1.9. WEB-MISC prefix-get // 24,069 alerts 4.1921% of all alerts

This alert is raised when a packet with an external address has a destination address of one of the internal web servers on TCP port 80, with the content of "get //" in the URI. This is classified as an attempted reconnaissance.

3.2.1.10. SYN-FIN scan! 21,297 alerts 3.7093% of all alerts

The SYN-FIN scan is used to do recon through firewalls and ACLs. The setting of conflicting TCP flags can confuse a firewall or router, which may let the packet through. A similarly confuses server may return a SYN-ACK in response.

3.2.1.11. ICMP Echo Request BSDtype 12,509 alerts 2.1787% of all alerts

This alert is raised when an ICMP echo request arrives with the following contents: "08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17" at a depth of 32. This usually means the echo request is coming from a BSD computer.

3.2.1.12. ICMP Destination Unreachable (Host Unreachable) 8,994 alerts 1.5665% of all alerts

ICMP destination unreachable messages are sent back to the initial sender when the last device can't find the next hop or the destination address. This can signify a routing problem, a downed device, or can be used for reconnaissance.

3.2.1.13. SCAN Proxy attempt 7,305 alerts 1.2723% of all alerts

Proxies allow attackers to appear as if they are coming from somewhere else, so they often scan for proxy servers, hoping to find vulnerable proxies to launch attacks from.

3.2.1.14. ICMP Destination Unreachable (Communication Admini 6,935 alerts 1.2079% of all alerts

ICMP destination unreachable messages are sent back to the initial sender when the last device

can't find the next hop or the destination address. This can signify a routing problem, a downed device, or can be used for reconnaissance. The difference from the previous ICMP destination unreachable is that this message is coming from a device that has a rule blocking the forwarding of that packet. It's not that the device sending the message doesn't know where to forward the packet, rather the device has been told not to forward such a packet.

3.2.1.15. Queso fingerprint 5,796 alerts 1.0095% of all alerts

Queso is a freeware network management software package that includes the ability to poll nodes and guess their function (much like HP OpenView Network Node Manager). Attackers have found Queso to be helpful for fingerprinting potential targets.

3.2.1.16. ICMP Fragment Reassembly Time Exceeded 4,923 alerts 0.8574% of all alerts

ICMP fragment reassembly time exceeded is sent when a receiving device does not receive all of the fragments needed to reassemble a fragmented packet within a predefined time period.

3.2.1.17. ICMP Echo Request Nmap or HPING2 4,031 alerts 0.7021% of all alerts

This alert is raised when an ICMP echo request arrives in the format used by Nmap or HPING2.

3.2.1.17. BACKDOOR NetMetro File List 3,586 alerts 0.6246% of all alerts

This alert is raised when a packet going from the internal network has a external destination address with a destination port of 5032 and the content includes two consecutive dashes (unicode "2D"). The NetMetro backdoor has a file manager feature.

3.2.1.18. ICMP Echo Request L3retriever Ping 2,557 alerts 0.4453% of all alerts

This alert is raised when an ICMP echo request arrives from an external address to an internal address and contains the following content:
"ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI" at depth 32, which usually signifies that it is the L3retriever tool sending the ping.

3.2.1.19. SMB Name Wildcard 2,257 alerts 0.3931% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address is coming into an internal address on the SMB server port where the content includes a wildcard.

3.2.1.20. Watchlist 000222 NET-NCFC 2,141 alerts 0.3729% of all alerts

The Watchlist is a custom rule. It appears that traffic from this ISP has been responsible for many alerts in the past so the security personnel have created a rule to watch any traffic to or from that ISP.

3.2.1.21. ICMP Destination Unreachable (Protocol Unreachable 1,780 alerts 0.3100% of all alerts

ICMP destination unreachable messages are sent back to the initial sender when the last device can't find the next hop or the destination address. This can signify a routing problem, a downed device, or can be used for reconnaissance.

3.2.1.22. INFO FTP anonymous FTP 1,543 alerts 0.2687% of all alerts

This alert is raised when a packet with an external source address has an internal destination address and a destination port of TCP 21 and includes content of "USER anonymous|0D0A|" (the 0D0A is unicode for carriage return and line feed).

3.2.1.23. External RPC call 1,478 alerts 0.2574% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address is coming into an internal address on TCP port 111.

3.2.1.24. connect to 515 from outside 1,380 alerts 0.2404% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address is coming into an internal address on TCP port 515, the lpd port.

3.2.1.25. BACKDOOR NetMetro Incoming Traffic 1,098 alerts 0.1912% of all alerts

This alert is raised when a packet coming from the external network on TCP port 5031 to an internal destination address with a destination port of anything except the range of 53 to 80. NetMetro is a dangerous backdoor.

3.2.1.26. WEB-MISC Attempt to execute cmd 1,041 alerts 0.1813% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address is coming into an internal web server on TCP port 80, with content including "cmd".

3.2.1.27. WEB-MISC 403 Forbidden 1,019 alerts 0.1775% of all alerts

This alert is raised when a packet from an internal web server with source port of TCP 80 is sent to an external address with "HTTP/1.1 403" in the content at depth 12. This would tell you that an external attacker is trying to get to web pages for which he doesn't have permission.

3.2.1.28. SMTP relaying denied 865 alerts 0.1507% of all alerts

This alert is raised when a packet from an internal SMTP email server with source port of TCP

25 is sent to an external address with “550 5.7.1” in the content at depth 70. This is the code that signals SMTP relaying denied. Relaying is when a third party tries to use your email server to send email. Spammers try to do this, attackers may do this to make their messages look authentic.

3.2.1.29. EXPLOIT x86 NOOP 789 alerts 0.1374% of all alerts

This exact rule does not exist in the snort current rules that were downloaded on Feb. 2, 2002. It looks like this rule has been expanded to more operating systems and encoding types. I predict this rule what is now the following:

shellcode.rules:alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"SHELLCODE x86 NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth: 128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:4;)

Which alerts when there are many of the x86 NOOP characters. When there are many NOOP characters in a row, there’s a good chance the sender is looking to run a buffer overflow type of attack.

3.2.1.30. INFO Inbound GNUTella Connect accept 736 alerts 0.1282% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an internal address goes out to an external address with “GNUTELLA CONNECT” in the content. GNUTella is a peer-to-peer file sharing application that can be used to spread worms/viruses.

3.2.1.31. ICMP Echo Request Windows 729 alerts 0.1270% of all alerts

This alert is raised when an ICMP echo request arrives from an external address to an internal address and contains the content that tells you the source was probably a windows computer.

3.2.1.32. spp_http_decode: IIS Unicode attack detected 695 alerts 0.1210% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address is going to an internal address on TCP port 80, with URI content that would exploit the IIS unicode vulnerability.

3.2.1.33. ICMP traceroute 683 alerts 0.1190% of all alerts

The ICMP traceroute is one of the most basic and simple ways to get information on a network. Traceroute sends an ICMP packet with a TTL of 1 to the destination. The first device that receives the packet decrements the TTL, causing an ICMP Time Exceeded packet to be sent back.
Traceroute then sends an ICMP packet with a TTL of 2 to the destination. The second device that receives the packet sends back an ICMP Time Exceeded packet back to the source. Traceroute continues this process until it reaches the destination. The result is that you get back

a map of the path taken from the source to the destination and the time it took for each hop. Traceroute can be used to mark routers for a denial of service attack.

3.2.1.34. TCP SRC and DST outside network 604 alerts 0.1052% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address is going to another external address. When both the source and destination address are external, we should not receive the packet, or the packet is crafted in some way.

3.2.1.35. Incomplete Packet Fragments Discarded 601 alerts 0.1047% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic where the TCP header says there are more fragments, and not all of the fragments arrived.

3.2.1.36. Null scan! 594 alerts 0.1035% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic where the TCP header has no TCP flags set.

3.2.1.37. ICMP Echo Request Sun Solaris 560 alerts 0.0975% of all alerts

This alert is raised when an ICMP echo request arrives from an external address to an internal address and contains the content that tells you the source was probably a Solaris computer.

3.2.1.38. FTP DoS ftpd globbing 485 alerts 0.0845% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic when a TCP packet arrives with an internal destination on destination port 21 with certain content that would cause a denial of service to an FTP daemon.

3.2.1.39 INFO Napster Client Data 451 alerts 0.0785% of all alerts

This is one of multiple alerts that only differ on the destination port. This alert is raised when a packet from the internal network destined for an external address on one of the following TCP ports: 6699, 7777, 6666 or 5555 and has ".mp3" in the content.

3.2.1.40. INFO Possible IRC Access 432 alerts 0.0752% of all alerts

This alert is raised when a TCP packet from the internal network is headed to an external address on a port between 6666 and 7000 with "NICK" in the content.

3.2.1.41. TELNET login incorrect 418 alerts 0.0728% of all alerts

This alert is raised when a TCP packet from an internal address is headed to an external address with source port of 23 and "Login incorrect" in the content.

3.2.1.42. INFO Outbound GNUTella Connect accept 361 alerts 0.0629% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an internal address goes out to an external address with "GNUTELLA CONNECT" in the content. GNUTella is a peer-to-peer file sharing application that can be used to spread worms/viruses.

3.2.1.43. Port 55850 tcp - Possible myserver activity - ref. 323 alerts 0.0563% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an internal address goes out to an external address with a destination port of TCP 55850.

3.2.1.44. ICMP Echo Request CyberKit 2.2 Windows 306 alerts 0.0533% of all alerts

This alert is raised when an ICMP echo request arrives from an external address to an internal address and contains the content that tells you the source was probably a windows computer using CyberKit 2.2.

3.2.1.45. NMAP TCP ping! 292 alerts 0.0509% of all alerts

This alert is raised when an packet arrives from an external address to an internal address and contains the content that tells you the source was probably using NMAP to ping using a TCP packet.

3.2.1.46. WEB-MISC http directory traversal 292 alerts 0.0509% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 with ".." in the content.

3.2.1.47. INFO - Possible Squid Scan 279 alerts 0.0486% of all alerts

This alert is raised when a packet arrives from an external address to an internal address on TCP port 3128 with the SYN flag set.

3.2.1.48. CS WEBSERVER - external ftp traffic 263 alerts 0.0458% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic to a certain set of external ftp servers is seen. I do not know what the "CS" part stands for, which would probably explain the what type of external ftp servers are watched by this rule.

3.2.1.49. WEB-MISC count.cgi access 220 alerts 0.0383% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 with "/count.cgi" in the content.

3.2.1.50. High port 65535 tcp - possible Red Worm - traffic 164 alerts 0.0286% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic to or from port TCP 65535.

3.2.1.51. WEB-IIS view source via translate header 139 alerts 0.0242% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 with "Translate: F" in the content.

3.2.1.52. High port 65535 udp - possible Red Worm - traffic 138 alerts 0.0240% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic to or from port UDP 65535.

3.2.1.53. ICMP Destination Unreachable (Fragmentation Needed 134 alerts 0.0233% of all alerts

ICMP destination unreachable messages are sent back to the initial sender when the last device can't find the next hop or the destination address. This can signify a routing problem, a downed device, or can be used for reconnaissance. This message is slightly different in that the fragmentation needed is caused by a device that has an MTU setting of smaller than the size of the received packet, but the packet has the do not fragment flag set.

3.2.1.54. SCAN FIN 128 alerts 0.0223% of all alerts

This alert is raised when a packet arrives from an external address to an internal address where only the FIN flag is set among the TCP flags.

3.2.1.55. WEB-CGI scriptalias access 124 alerts 0.0216% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 with "///" in the content.

3.2.1.56. Possible trojan server activity 118 alerts 0.0206% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic to or from known trojan ports like TCP 31337.

3.2.1.57. MISC Large ICMP Packet 88 alerts 0.0153% of all alerts

In my snort rules, this alert is raised when a ICMP packet larger than 800 bytes comes into the internal network from an external address. It is unusual to have packets greater than 800 bytes, so these should be monitored.

3.2.1.58. WEB-IIS _vti_inf access 84 alerts 0.0146% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 with "_vti_inf.html" in the content.

3.2.1.59. WEB-FRONTPAGE _vti_rpc access 83 alerts 0.0145% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 with "_vti_rpc" in the content.

3.2.1.60. INFO Inbound GNUTella Connect request 71 alerts 0.0124% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an internal address goes out to an external address with "GNUTELLA CONNECT" in the content. GNUTella is a peer-to-peer file sharing application that can be used to spread worms/viruses.

3.2.1.61. connect to 515 from inside 69 alerts 0.0120% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an internal address is coming into an internal address on TCP port 515, the lpd port.

3.2.1.62. beetle.ucs 65 alerts 0.0113% of all alerts

This is a custom rule, I found a link that identified beetle.ucs.umbc.edu as a computer with a CD burner. Other than that, I'd guess that all traffic to and from this computer is monitored.

3.2.1.63. TFTP - Internal TCP connection to external tftp se 65 alerts 0.0113% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an internal address is going to an external address with destination port of UDP 69.

3.2.1.64. WEB-CGI redirect access 63 alerts 0.0110% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 includes "redirect" in the URI content.

3.2.1.65. WEB-IIS Unauthorized IP Access Attempt 62 alerts 0.0108% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 includes “403” and “Forbidden” in the URI content.

3.2.1.66. SUNRPC highport access! 61 alerts 0.0106% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address going to an internal address on a high numbered port used by Sun rpc, like UDP 32770.

3.2.1.67. INFO - Web Cmd completed 46 alerts 0.0080% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address to an internal address with “cmd” in the URI content.

3.2.1.68. WEB-CGI rsh access 43 alerts 0.0075% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 includes “rsh” in the URI content.

3.2.1.69. TELNET access 32 alerts 0.0056% of all alerts

This alert is raised when a packet arrives from an external address to an internal address on TCP port 23 includes “|FF FD 18 FF FD 1F FF FD 23 FF FD 27 FF FD 24|” in the content.

3.2.1.70. SCAN Synscan Portscan ID 19104 30 alerts 0.0052% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when a certain type of portscan was found.

3.2.1.71. WEB-CGI formmail access 29 alerts 0.0051% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 includes “formmail” in the URI content.

3.2.1.72. Virus - Possible scr Worm 28 alerts 0.0049% of all alerts

This alert is raised when a packet arrives from an source port TCP 110 to any address with “.scr” in the content.

3.2.1.73. WEB-MISC compaq nsight directory traversal 27 alerts 0.0047% of all alerts

This alert is raised when a packet arrives from an external address to an internal address on TCP port 2301 with “..” in the content.

3.2.1.74. DDOS shaft client to handler 25 alerts 0.0044% of all alerts

This alert is raised when a packet arrives from an external address to an internal address on TCP port 20432.

3.2.1.75. SNMP public access 25 alerts 0.0044% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when a packet from an external address to an internal address on UDP 161 with "public" in the content.

3.2.1.76. WEB-CGI csh access 19 alerts 0.0033% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 includes "csh" in the URI content.

3.2.1.77. X11 outgoing 19 alerts 0.0033% of all alerts

This alert is raised when a packet arrives from an external address with source port between TCP 6000 and 6005 to an internal address.

3.2.1.78. EXPLOIT x86 setuid 0 16 alerts 0.0028% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when a packet from an external address to an internal address with content that signals a setuid.

3.2.1.79. Port 55850 udp - Possible myserver activity - ref. 16 alerts 0.0028% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an internal address goes out to an external address with a destination port of UDP 55850.

3.2.1.80. Attempted Sun RPC high port access 15 alerts 0.0026% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address going to an internal address on a high numbered port used by Sun rpc, like UDP 32770.

3.2.1.81. EXPLOIT x86 NOPS 15 alerts 0.0026% of all alerts

This exact rule does not exist in the snort current rules that were downloaded on Feb. 2, 2002. It looks like this rule has been expanded to more operating systems and encoding types. I predict this rule what is now the following:

shellcode.rules:alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"SHELLCODE x86 NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth: 128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:4;)

Which alerts when there are many of the x86 NOOP characters. When there are many NOOP

characters in a row, there's a good chance the sender is looking to run a buffer overflow type of attack.

3.2.1.82. EXPLOIT x86 setgid 0 15 alerts 0.0026% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when a packet from an external address to an internal address with content that signals a setgid.

3.2.1.83. ICMP redirect (Host) 15 alerts 0.0026% of all alerts

The purpose of an ICMP redirect message is to make routing faster in a local network.
Given that router1 is the default gateway for computer1.
If computer1 wants to communicate to computer2, it will send the packet to router1.
If router1 finds the next hop to computer2 is router2, where router2 must be on the same network as router1 and computer1, router1 will forward the packet to router2 and send an ICMP redirect message to computer1 telling it to send future packets destined for computer2 through router2 directly, thereby adding a route to computer1. (Stevens, pp.119-123)
This alert is raised when an ICMP redirect message is received from an external address to an internal address.

3.2.1.84. WEB-FRONTPAGE shtml.exe 14 alerts 0.0024% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 includes "_vti_bin/shtml.exe" in the URI content.

3.2.1.85. WEB-FRONTPAGE fpcount.exe access 13 alerts 0.0023% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 includes "fpcount.exe" in the URI content.

3.2.1.86. SMTP chameleon overflow 12 alerts 0.0021% of all alerts

This alert is raised when a packet arrives from an external address to an internal SMTP server on TCP port 25 includes "HELP " in the content and the size of the packet is greater than 500 bytes..

3.2.1.87. DNS zone transfer 11 alerts 0.0019% of all alerts

This alert is raised when a packet arrives from an external address to an internal address on TCP port 53 includes "|00 00 FC|" in the content with an offset of 13.

3.2.1.88. ICMP Destination Unreachable (Network Unreachable) 11 alerts 0.0019% of all alerts

ICMP destination unreachable messages are sent back to the initial sender when the last device can't find the next hop or the destination address. This can signify a routing problem, a downed device, or can be used for reconnaissance. This message is slightly different in that the device sending the error message does not know how to get to the destination network.

3.2.1.89. Possible pif Worm 10 alerts 0.0017% of all alerts

This alert is raised when a packet arrives from an source port TCP 110 to any address with ".scr" in the content.

3.2.1.90. WEB-IIS File permission canonicalization 8 alerts 0.0014% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 includes "/scripts/..%c0%af../" or "/scripts/..%c1%1c../" or "/scripts/..%c1%9c../" in the URI content.

3.2.1.91. WEB-CGI archie access 7 alerts 0.0012% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 includes "archie" in the URI content.

3.2.1.92. WEB-FRONTPAGE posting 7 alerts 0.0012% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 includes "POST" in the content and "author.dll" in the URI content.

3.2.1.93. INFO - Web Dir listing 6 alerts 0.0010% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address going to an internal address on TCP port 80 with "Dir" in the content.

3.2.1.94. INFO napster login 6 alerts 0.0010% of all alerts

This alert is raised when a packet arrives from an internal address to an external address on TCP port 8888 includes "|00 0200|" in the content at offset 1 and depth 3.

3.2.1.95. MISC PCAnywhere Startup 6 alerts 0.0010% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic either to or from TCP port 5631 or 5632.

3.2.1.96. spp_http_decode: CGI Null Byte attack detected 6 alerts 0.0010% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address going to an internal web server on TCP port 80 where there is content that signals a CGI null byte attack.

3.2.1.97. TFTP - External UDP connection to internal tftp se 6 alerts 0.0010% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address is going to an internal address with destination port of UDP 69.

3.2.1.98. WEB-FRONTPAGE shtml.dll 6 alerts 0.0010% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 includes "_vti_bin/shtml.dll" in the URI content.

3.2.1.99. WEB-MISC Lotus Domino directory traversal 6 alerts 0.0010% of all alerts

This alert is raised when a packet arrives from an external address to an internal web server on TCP port 80 includes ".nsf" and ".." in the URI content.

3.2.1.100. IDS475/web-iis_web-webdav-propfind 5 alerts 0.0009% of all alerts

This is not one of the standard rules in the snort rules downloaded on Feb. 2, 2002, so I cannot be sure of what this rule alerts on. I would predict that the alert is raised when traffic from an external address is going to an internal address with destination port of TCP 80 with "propfind" in the content.

3.3. Scans analysis

There were 2,627,761 scans captured during the seven day stretch of this analysis.
Of that group, 781,305 were UDP scans and 1,846,456 were TCP scans.
There were 1,665 unique source addresses, which is too many to print, even in an index (1,665 at 60 lines per page = 28 pages).
I'm including the top 10 scanners here.
There were 1,728,087 unique destination addresses, which is far too many to print. The top 10 destination addresses are included below.

More detailed analysis of the scans can be found in Appendix 8.

3.3.1. Address based analysis

I found these results rather perplexing:
That there are far more unique destination addresses than source addresses.
That most of the source addresses are from the MY.NET.x.x network.
That many of the top destination addresses are from the 24.x.x.x network, which is known to host many cable modems.

I expected that many of the scans would have source addresses from the 24.x.x.x network, since the cable modem computers are often compromised and used by attackers to attack others with impunity.
I expected that most of the destination addresses would be on the MY.NET.x.x network, where outsiders are scanning the computers on the MY.NET.x.x network.
I expected that there would be few unique destination addresses and many unique source

addresses, where the computers on the MY.NET.x.x network are scanned by many outsiders.

These results are the opposite of what I expected. These results caused me to go back to the downloaded files to make sure that I didn't switch around the sources and destinations. That wasn't the case.

These findings lead me to the conclusion that most of the scans are coming from the MY.NET.x.x network. The MY.NET.x.x network is not the victim, quite the opposite. The scans are coming from the MY.NET.x.x network, scanning a wide variety of destination addresses.

3.3.1.1. Top source addresses

The top 10 sources of scans:

srcip	count
MY.NET.162.233	926049
MY.NET.163.15	685164
MY.NET.87.50	520160
MY.NET.98.203	27085
209.190.237.123	20606
205.188.233.121	15308
205.188.228.33	12818
62.211.247.3	12251
205.188.246.121	11413
24.205.153.114	9895

The most obvious thing we learn from this is that a vast majority of the scans are coming from the inside. The top four scanners all belong to the MY.NET.x.x network, the sum of their scans is 2,158,558, which is 82.14% of the total number of scans. The top scanner, MY.NET.162.233 alone accounts for 35.24% of the total scans with a whopping 926,049 scans.

3.3.1.2. Top destination addresses:

Top 10 destinations of scans:

dstip	count
24.164.41.210	21952
MY.NET.70.134	20614
MY.NET.190.15	15144
216.33.98.254	11066
194.251.249.182	7144
24.157.184.117	7080
MY.NET.70.148	6598
24.23.140.185	5574

```
| 128.8.240.71     |  5500 |
| MY.NET.106.178   |  5195 |
+------------------+-------+
```

Of the top 20 destination addresses, six belong to the 24.x.x.x network and nine belong to the MY.NET.x.x network.
24.164.41.210 belongs to Road Runner, rr.com.
24.157.184.117 belongs to @Home, home.com.

3.3.2. Port based analysis:

3.3.2.1. Source port analysis:

Looking at source ports isn't really going to tell much, so I skipped this section.

3.3.2.2. Destination port analysis:

The destination port will give us some idea of what services attackers are hoping to exploit.

Here are the top 10 ports scanned including both UDP and TCP:

```
+---------+----------+---------+
| dstport | protocol | count   |
+---------+----------+---------+
| 22      | T        | 1645747 |
| 27005   | U        |  261949 |
| 6970    | U        |   76742 |
| 1214    | T        |   61333 |
| 21      | T        |   50331 |
| 6112    | U        |   42212 |
| 27500   | U        |   21624 |
| 25      | T        |   19289 |
| 0       | U        |   15722 |
| 6346    | T        |   11141 |
+---------+----------+---------+
```

3.3.2.2.1. Top scanned port: TCP 22, ssh

By far the most scanned port was TCP port 22, the ssh port, with 1,645,747 (62.63% of the total scans), so by this information it would be a good guess that most of the scans were to look for ssh daemons, which were probably followed by an ssh exploit.

3.3.2.2.2. Second most scanned port: UDP 27005, Half Life game client

The second highest scanned port was UDP 27005, which is attributed to the client end of the Half Life game. Which means the scans to UDP port 27005 were probably not scans at all, but the traffic exceeded the highwatermark that snort uses to determine whether it is a scan.

3.3.2.2.3. Third most scanned port: UDP 6970, streaming ads in Real Player

The third highest scanned port was UDP 6970, which is attributed to the RTP protocol (RFC 1889), "which provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video." (Audio-Video Transport Working Group) I was able to find some correlations for this port where others have found this to be caused by streaming ads in Real Player (Allerdice).

3.3.2.2.4. Fourth most scanned port: TCP 1214, KaZaA peer-to-peer file sharing (like Napster)

3.3.2.2.5. Fifth most scanned port: TCP 21, FTP

3.3.2.2.6. Sixth most scanned port: UDP 6112, Battle.net game servers

3.3.2.2.7. Seventh most scanned port: UDP 27500, Quake game server

3.3.2.2.8. Eighth most scanned port: TCP 25, SMTP

3.3.2.2.9. Ninth most scanned port: UDP 0, fingerprinting?

I could not find two consistent correlations of this port, but I did find two articles that both explain that a packet to UDP port 0 can be used to fingerprint BSD 4.4 (Hagino) and Foundry networks networking devices (Arkin).

3.3.2.2.10. Tenth most scanned port: TCP 6346, Gnutella

3.3.3. Conclusions and defensive recommendations:

The research done in Index 3 shows that over half of all the scans collected (1,611,200 of 2,627,761) were from two internal sources (MY.NET.162.233 and MY.NET.163.15), scanning for a specific port (TCP 22).

We also find that there is a significant amount of gaming and peer-to-peer file sharing activity which should be curtailed. Index 3 lists the top offenders in each of the top ten scanned ports. This should give the security personnel the information necessary to reduce the number of scans and other illicit activities. For example, we find that much of the gaming alerts come from a single source address, MY.NET.87.50, therefore we should let the user at that computer know that the gaming needs to stop, or block the traffic altogether.

The gaming ports, UDP 27005, UDP 6112 and UDP 27500 should be blocked at the routers and/or firewalls.
The peer-to-peer file sharing ports, TCP 1214 and TCP 6346 should also be blocked at the routers and/or firewalls.
The ssh, FTP, and SMTP ports can also be blocked for all except the computers providing those services.

3.4. OOS analysis:

There were 8,390 captured Out of spec. packets. These are packets that violate some rule of TCP

behavior, most likely these are crafted packets.
Of the 8,390 total captures, 7,988 occurred on Christmas Day, 12/25/2001, with none occurring on 12/20/2001, 12/21/2001 and 12/26/2001.
There were 82 unique source addresses and 7,954 unique destination addresses.

More detailed analysis on the OOS packets can be found in Appendix 9.

3.4.1. Address analysis:

We find that there are 82 sources and 7,954 destinations, this kind of disparity instantly makes the analyst think these OOS packets are almost entirely made up of scans.

3.4.1.1. Source address analysis:

The top 10 source addresses:

srcip	count
24.0.28.234	7931
210.125.178.52	167
199.183.24.194	80
64.172.24.155	40
24.36.185.188	15
141.157.92.22	12
211.39.150.91	11
65.165.238.50	9
202.168.254.178	7
213.84.157.192	7

We can see that almost all of the OOS packets came from a single host, 24.0.28.234.
There were a total of 82 unique source addresses.
The good news is that not a single one of these was from the internal network.

3.4.1.2. Destination address analysis:

The top 10 destination addresses:

dstip	count
MY.NET.163.15	168
MY.NET.253.43	89
MY.NET.70.70	44
MY.NET.253.114	17
MY.NET.253.125	16
MY.NET.70.49	16
MY.NET.253.41	14
MY.NET.1.6	12

```
| MY.NET.60.14    |     10 |
| MY.NET.100.165 |      9 |
+----------------+-------+
```

There was a much wider range of destination addresses, 7,954 unique addresses. All of the OOS packets had a destination address on the internal network.

3.4.2. Port analysis:

Investigating the source ports won't help us much, so I'm skipping it.

Top 10 destination ports:

```
+---------+-------+
| dstport | count |
+---------+-------+
| 22      |  7932 |
| 25      |   116 |
|         |    42 |
| 80      |    42 |
| 1214    |    34 |
| 21536   |    19 |
| 563     |    12 |
| 0       |    10 |
| 113     |     7 |
| 6346    |     6 |
+---------+-------+
```

There are 141 unique destination ports, but all after the first 10 are either 1 or 2 occurrences. I'll describe the ports for the top 5 here, more detailed analysis is done in the Appendix 9.

3.4.2.1. Top OOS port #1: TCP 22, ssh:

We've already taken a look at 7,931 of the packets to port 22, those are scans from 24.0.28.234.

3.4.2.2. Top OOS port #2: TCP 25, SMTP:

One single source accounts for 80 of the 116 packets to destination port 25.
199.183.24.194 belongs to Red Hat Software, Inc., which sent all 80 of those packets to MY.NET.253.43 at pretty random times with the two reserved TCP flag bits set along with the SYN flag.

3.4.2.3. Top OOS port #3: <blank>:

All of these are crafted packets that have both the DF (do not fragment) and MF (more fragments) bits set.
Each packet has a fragment offset of 0x0, so each packet is claiming to be the first packet.
Lastly, none of these packets includes a source or destination port.

3.4.2.4. Top OOS port #4: TCP 80, http:

These packets were fairly spread out among source and destinations, so I wrote them off as wrong numbers.

3.4.2.5. Top OOS port #5: TCP 1214, KaZaA file sharing:

We've already identified a couple of KaZaA servers, these packets may lead us to other KaZaA servers which we need to clean.

A1. Appendix 1: Network Detect #1: NIMDA snort output:

```
[**] WEB-IIS CodeRed v2 root.exe access [**]
01/25-20:47:25.264108 xxx.xxx.xxx.xxx:2057 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:853 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0xB8891867  Ack: 0xA85085D2  Win: 0xFD5C  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 72 6F 6F  GET /scripts/roo
74 2E 65 78 65 3F 2F 63 2B 64 69 72 20 48 54 54  t.exe?/c+dir HTT
50 2F 31 2E 30 0D 0A 48 6F 73 74 3A 20 77 77 77  P/1.0..Host: www
0D 0A 43 6F 6E 6E 6E 65 63 74 69 6F 6E 3A 20 63  ..Connnection: c
6C 6F 73 65 0D 0A 0D 0A                          lose....

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

[**] WEB-IIS cmd.exe access [**]
01/25-20:47:25.943226 xxx.xxx.xxx.xxx:2062 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1046 IpLen:20 DgmLen:120 DF
***AP*** Seq: 0xB88E9ED3  Ack: 0x65EC2E38  Win: 0xFD5C  TcpLen: 20
47 45 54 20 2F 63 2F 77 69 6E 6E 74 2F 73 79 73  GET /c/winnt/sys
74 65 6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63  tem32/cmd.exe?/c
2B 64 69 72 20 48 54 54 50 2F 31 2E 30 0D 0A 48  +dir HTTP/1.0..H
6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E 6E 65  ost: www..Connne
63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A 0D 0A  ction: close....

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

[**] WEB-IIS cmd.exe access [**]
01/25-20:47:26.256298 xxx.xxx.xxx.xxx:2070 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1134 IpLen:20 DgmLen:120 DF
***AP*** Seq: 0xB8957188  Ack: 0x86FC8DBE  Win: 0xFD5C  TcpLen: 20
47 45 54 20 2F 64 2F 77 69 6E 6E 74 2F 73 79 73  GET /d/winnt/sys
74 65 6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63  tem32/cmd.exe?/c
2B 64 69 72 20 48 54 54 50 2F 31 2E 30 0D 0A 48  +dir HTTP/1.0..H
6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E 6E 65  ost: www..Connne
63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A 0D 0A  ction: close....

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

[**] WEB-IIS cmd.exe access [**]
01/25-20:47:26.487337 xxx.xxx.xxx.xxx:2077 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1153 IpLen:20 DgmLen:136 DF
***AP*** Seq: 0xB89BAE37  Ack: 0x40259543  Win: 0xFD5C  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25  GET /scripts/..%
35 63 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74 65  5c../winnt/syste
6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 64  m32/cmd.exe?/c+d
69 72 20 72 20 48 54 54 50 2F 31 2E 30 0D 0A 48  ir r HTTP/1.0..H
```

```
6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E 6E 65  ost: www..Connne
63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A        ction: close..

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

[**] WEB-FRONTPAGE /_vti_bin/ access [**]
01/25-20:47:26.755936 xxx.xxx.xxx.xxx:2081 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1184 IpLen:20 DgmLen:157 DF
***AP*** Seq: 0xB89F83D8  Ack: 0xB2C7BE4A  Win: 0xFD5C  TcpLen: 20
47 45 54 20 2F 5F 76 74 69 5F 62 69 6E 2F 2E 2E  GET /_vti_bin/..
25 35 63 2E 2E 2F 2E 2E 25 35 63 2E 2E 2F 2E 2E  %5c../..%5c../..
25 35 63 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74  %5c../winnt/syst
65 6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B  em32/cmd.exe?/c+
64 69 72 20 63 2B 64 69 72 20 48 54 54 50 2F 31  dir c+dir HTTP/1
2E 30 0D 0A 48 6F 73 74 3A 20 77 77 77 0D 0A 43  .0..Host: www..C
6F 6E 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C 6F     onnnection: clo

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

[**] WEB-IIS cmd.exe access [**]
01/25-20:47:26.999196 xxx.xxx.xxx.xxx:2082 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1201 IpLen:20 DgmLen:157 DF
***AP*** Seq: 0xB8A13850  Ack: 0x68505AF4  Win: 0xFD5C  TcpLen: 20
47 45 54 20 2F 5F 6D 65 6D 5F 62 69 6E 2F 2E 2E  GET /_mem_bin/..
25 35 63 2E 2E 2F 2E 2E 25 35 63 2E 2E 2F 2E 2E  %5c../..%5c../..
25 35 63 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74  %5c../winnt/syst
65 6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B  em32/cmd.exe?/c+
64 69 72 20 63 2B 64 69 72 20 48 54 54 50 2F 31  dir c+dir HTTP/1
2E 30 0D 0A 48 6F 73 74 3A 20 77 77 77 0D 0A 43  .0..Host: www..C
6F 6E 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C 6F     onnnection: clo

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

[**] WEB-IIS cmd.exe access [**]
01/25-20:47:27.233015 xxx.xxx.xxx.xxx:2085 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1217 IpLen:20 DgmLen:185 DF
***AP*** Seq: 0xB8A40FB5  Ack: 0xFE04F22C  Win: 0xFD5C  TcpLen: 20
47 45 54 20 2F 6D 73 61 64 63 2F 2E 2E 25 35 63  GET /msadc/..%5c
2E 2E 2F 2E 2E 25 35 63 2E 2E 2F 2E 2E 25 35 63  ../..%5c../..%5c
2F 2E 2E 35 35 2E 2E 2F 2E 2E 63 31 2E 2E 2F 2E  /..55../..c1../.
2E 2F 2E 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74  ./.../winnt/syst
65 6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B  em32/cmd.exe?/c+
64 69 72 20 33 32 2F 63 6D 64 2E 65 78 65 3F 2F  dir 32/cmd.exe?/
63 2B 64 69 72 20 48 54 54 50 2F 31 2E 30 0D 0A  c+dir HTTP/1.0..
48 6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E     Host: www..Conn

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

```
[**] WEB-IIS cmd.exe access [**]
01/25-20:47:27.459504 xxx.xxx.xxx.xxx:2087 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1231 IpLen:20 DgmLen:137 DF
***AP*** Seq: 0xB8A68384  Ack: 0x81E8DFDE  Win: 0xFD5C  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25  GET /scripts/..%
63 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74 65 6D  c../winnt/system
33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 64 69  32/cmd.exe?/c+di
72 20 64 69 72 20 48 54 54 50 2F 31 2E 30 0D 0A  r dir HTTP/1.0..
48 6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E 6E  Host: www..Connn
65 63 74 69 6F 6E 3A 20 63 6C 6F 73 65           ection: close

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

[**] WEB-IIS cmd.exe access [**]
01/25-20:47:27.707917 xxx.xxx.xxx.xxx:2105 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1255 IpLen:20 DgmLen:137 DF
***AP*** Seq: 0xB8B11958  Ack: 0x1DFB5A2F  Win: 0xFD5C  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25  GET /scripts/..%
63 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74 65 6D  c../winnt/system
33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 64 69  32/cmd.exe?/c+di
72 20 64 69 72 20 48 54 54 50 2F 31 2E 30 0D 0A  r dir HTTP/1.0..
48 6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E 6E  Host: www..Connn
65 63 74 69 6F 6E 3A 20 63 6C 6F 73 65           ection: close

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

[**] WEB-IIS cmd.exe access [**]
01/25-20:47:27.882915 xxx.xxx.xxx.xxx:2112 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1327 IpLen:20 DgmLen:137 DF
***AP*** Seq: 0xB8B64979  Ack: 0x24C50AC0  Win: 0xFD5C  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25  GET /scripts/..%
63 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74 65 6D  c../winnt/system
33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 64 69  32/cmd.exe?/c+di
72 20 64 69 72 20 48 54 54 50 2F 31 2E 30 0D 0A  r dir HTTP/1.0..
48 6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E 6E  Host: www..Connn
65 63 74 69 6F 6E 3A 20 63 6C 6F 73 65           ection: close

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

[**] WEB-IIS cmd.exe access [**]
01/25-20:47:28.040355 xxx.xxx.xxx.xxx:2118 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1344 IpLen:20 DgmLen:137 DF
***AP*** Seq: 0xB8BACD2B  Ack: 0x8429AD5C  Win: 0xFD5C  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25  GET /scripts/..%
63 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74 65 6D  c../winnt/system
33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 64 69  32/cmd.exe?/c+di
72 20 64 69 72 20 48 54 54 50 2F 31 2E 30 0D 0A  r dir HTTP/1.0..
48 6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E 6E  Host: www..Connn
```

65 63 74 69 6F 6E 3A 20 63 6C 6F 73 65 ection: close

=+=

[**] WEB-IIS cmd.exe access [**]
01/25-20:47:28.368999 xxx.xxx.xxx.xxx:2125 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1413 IpLen:20 DgmLen:138 DF
AP Seq: 0xB8C13C2D Ack: 0xD2FD3916 Win: 0xFD5C TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25 GET /scripts/..%
35 63 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74 65 5c../winnt/syste
6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 64 m32/cmd.exe?/c+d
69 72 20 64 69 72 20 48 54 54 50 2F 31 2E 30 0D ir dir HTTP/1.0.
0A 48 6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E .Host: www..Conn
6E 65 63 74 69 6F 6E 3A 20 63 6C 6F 73 65 nection: close

=+=

[**] WEB-IIS cmd.exe access [**]
01/25-20:47:28.456453 xxx.xxx.xxx.xxx:2131 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1421 IpLen:20 DgmLen:136 DF
AP Seq: 0xB8C6C294 Ack: 0xB0DFB16D Win: 0xFD5C TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25 GET /scripts/..%
35 63 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74 65 5c../winnt/syste
6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 64 m32/cmd.exe?/c+d
69 72 20 72 20 48 54 54 50 2F 31 2E 30 0D 0A 48 ir r HTTP/1.0..H
6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E 6E 65 ost: www..Connne
63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A ction: close..

=+=

[**] WEB-IIS cmd.exe access [**]
01/25-20:47:28.649520 xxx.xxx.xxx.xxx:2134 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1437 IpLen:20 DgmLen:140 DF
AP Seq: 0xB8C9A538 Ack: 0xD98908F5 Win: 0xFD5C TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25 GET /scripts/..%
35 63 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74 65 5c../winnt/syste
6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 64 m32/cmd.exe?/c+d
69 72 20 63 2B 64 69 72 20 48 54 54 50 2F 31 2E ir c+dir HTTP/1.
30 0D 0A 48 6F 73 74 3A 20 77 77 77 0D 0A 43 6F 0..Host: www..Co
6E 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C 6F nnnection: clo

=+=

[**] WEB-IIS cmd.exe access [**]
01/25-20:47:28.876023 xxx.xxx.xxx.xxx:2139 -> xxx.xxx.xxx.xxx:80
TCP TTL:125 TOS:0x0 ID:1452 IpLen:20 DgmLen:136 DF
AP Seq: 0xB8CDFAA7 Ack: 0xF0C51E7D Win: 0xFD5C TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25 GET /scripts/..%

```
32 66 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74 65  2f../winnt/syste
6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 64  m32/cmd.exe?/c+d
69 72 20 72 20 48 54 54 50 2F 31 2E 30 0D 0A 48  ir r HTTP/1.0..H
6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E 6E 65  ost: www..Connne
63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A        ction: close..
```

=+=

A2. Appendix 2: Network Detect #5: ICMP redirect

inetnum: 194.225.65.0 - 194.225.65.255
netname: NRCGEBNET
descr: National Reasearch Centre for Genetic Enineering and
descr: Biotechnology. 15, Shafiel St, Ghods Ave, Tehran IRAN
country: IR

Also on Also on Nov 7 09:43:36 -> Nov 7 10:35:19, Nov 7 20:34:05 -> Nov 7 21:24:52, Nov 8 02:56:55 -> Nov 8 02:59:14, Nov 9 02:01:02 -> Nov 9 02:03:37, Nov 9 10:20:32 -> Nov 9 10:21:31, Nov 9 11:44:23 -> Nov 9 11:46:11, Nov 9 21:27:05 -> Nov 9 21:27:25

Total of 2558
Nov 10 00:13:37 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 00:17:08 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

Total of 1640
Nov 10 01:36:56 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 01:39:17 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

Total of 2676
Nov 10 03:00:28 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 03:04:18 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

Total of 2146
Nov 10 04:23:38 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 04:28:08 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

Total of 1502
Nov 10 05:46:59 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 05:52:07 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

Total of 2272
Nov 10 07:10:19 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 07:15:38 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

Total of 4134
Nov 10 08:33:39 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 08:37:08 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

Total of 3196
Nov 10 09:56:59 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 09:59:59 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

Total of 512
Nov 10 10:00:00 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 10:00:14 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

Total of 2694
Nov 10 11:20:30 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 11:25:27 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

Total of 3780
Nov 10 12:43:38 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification: Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 12:48:29 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification:

Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

Total of 3440
Nov 10 14:06:58 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification:
Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 14:11:59 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification:
Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

Total of 3900
Nov 10 15:30:17 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification:
Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98
.....
Nov 10 15:35:01 hostmi snort: [ID 992550 auth.alert] ICMP redirect net [Classification:
Potentially Bad Traffic Priority: 2]: 194.225.65.1 -> z.y.x.98

A3. Appendix 3: Tiny Fragments - Possible Hostile Activity [False Positive]

At the top of the list with 75,653 occurrences, the "Tiny Fragments…" alert begs for analysis.

A3.1. Source of trace:

"Analyze this" data from 12/20/2001 through 12/26/2001

A3.2. Detect was generated by:

Snort IDS

A3.3. Probability the source address was spoofed:

Unlikely since the goal of the tiny fragments attack is to make a connection through a restrictive firewall.

A3.4. Description of the attack:

A3.4.1. Definition of the "Tiny Fragments - Possible Hostile Activity" alert:

A3.4.1.1. Book definition of IP Fragmentation:

IP fragmentation is defined by W. Richard Stevens as:
The physical network layer normally imposes an upper limit on the size of the frame that can be transmitted. Whenever the IP layer receives an IP datagram to send, it determines which local interface the datagram is being sent on (routing), and queries that interface to obtain its MTU. IP compares the MTU with the datagram size and performs fragmentation, if necessary. Fragmentation can take place either at the original sending host or an intermediate router. (Stevens, p.148)

Meaning the purpose of fragmentation is to break a large packet into more manageable sized pieces, however those more manageable sized pieces will be at or near the maximum MTU (Maximum Transmission Unit) as that provides the best efficiency (least overhead). Therefore, we can say at some point there is a limit that no modern computers or networking equipment will go below in MTU size.

The more nefarious purpose of IP fragmentation is to attempt to bypass firewalls and/or many denial of service attacks. Jason Anderson's paper "An Analysis of Fragmentation Attacks" discusses several of the denial of service attacks including: Ping O' Death, Tiny Fragment, Teardrop, Overlapping Fragment (Anderson). He also shows that several firewalls were vulnerable to fragmentation attacks.

A3.4.1.2. Snort definition of "Tiny Fragments – Possible Hostile Activity":

Many of my predecessors in their GCIA practical assignments have defined the "Tiny Fragments - Possible Hostile Activity" alert as the tiny fragments attack defined in RFC 1858, where packets are crafted such that the fragments are so small that parts of the IP header are transmitted in separate packets allowing the IP header to pass through a firewall. However, upon further research, I found this alert to have a different meaning in the context of Snort alerts.

In an email message from 5/14/2000, Marty Roesch explains to a snort user that this alert is raised by the minfrag preprocessor when it sees fragments that are smaller than a defined threshold value (Roesch). To confirm whether or not this is the only instance in which the "Tiny Fragments - Possible Hostile Activity" alert would be raised, I took a look through the rules files and source code of the most recent version of Snort, version 1.8.3. Unfortunately this search turned out to be somewhat inconclusive since the "Tiny Fragments - Possible Hostile Activity" string didn't show up within any of the rules or source code. I couldn't find an older version of snort to confirm that the minfrag preprocessor is the only way a "Tiny Fragments - Possible Hostile Activity" alert could be raised, but I did find confirmation that the alert came from the minfrag preprocessor (Baker). Given that it would not be a good thing to have different events raise the same alert and given that many of the specific fragmentation attacks have their own alerts (Teardrop, Jolt, etc.), I will work on the assumption that this alert only means that the packets that raised the alert were fragments that were smaller than the defined threshold.

This leads to the next problem in this alert: I do not know what size threshold has been used to generate those alerts. The minfrag preprocessor (which has been deprecated) does not have a default threshold value, one must be set when the preprocessor is included. The point is that if the threshold value is set too high, these fragments may be legitimate fragmented packets.

A3.5. Attack mechanism:

The tiny fragments attack defined in RFC 1858 explains that if an attacker can make fragments extremely small, they can put only a portion of the TCP header on each fragment, with each fragment being separately able to pass through a firewall without being caught by a filter.

However, since our definition is different, so is our attack mechanism. The snort alert is raised whenever there is a fragment that is smaller than a predefined threshold in the minfrag preprocessor. Therefore this alert has a high likelihood of false positives. Since we do not know yet what type of attack this is, or even whether it is an attack, we need to investigate further. See

the active targeting section for the investigation.

A3.6. Correlations:

Most of the previous GCIA practicals have also found the "Tiny Fragments - Possible Hostile Activity" alert, including:

Gregory Lajohn, who captured alerts between: Sept. 1, 2001 – Sept. 5, 2001
(http://www.giac.org/practical/Gregory_Lajon_GCIA.doc)
John Jenkinson, who captured alerts between: Sept. 29, 2001 – Oct. 3, 2001
(http://www.giac.org/practical/John_Jenkinson_GCIA.doc)
Reuben Rubio, who captured alerts between: April 6, 2001 – April 10, 2001
(http://www.giac.org/practical/REUBEN_RUBIO_GCIA.doc)

A3.7. Evidence of active targeting:

To define whether these alerts were a case of active targeting, I need to get some idea of the intention of the attacker. I can only find that by doing further research.

A3.7.1. Who is sending and receiving these alerts?

Of the 75,653 alerts, I found nearly all (75,162, over 99%) of the alerts originated at the IP address MY.NET.8.1, which is an internal IP address.

srcip	count
MY.NET.8.1	75162
67.161.190.60	223
65.2.208.87	190
66.168.137.179	74
24.34.101.43	2
24.8.58.167	2

I also found that for each Source IP address, there was only one Destination IP address:

srcip	dstip
MY.NET.8.1	MY.NET.16.42
67.161.190.60	MY.NET.99.39
65.2.208.87	MY.NET.99.39
66.168.137.179	MY.NET.87.50
24.34.101.43	MY.NET.100.236
24.8.58.167	MY.NET.253.125

We can see that the second and third highest offenders have targeted the same Destination IP address. This may mean that there was an attack on MY.NET.99.39, but we need more information. On the other hand, I question whether the alerts with source of MY.NET.8.1 are an attack – I don't yet see a reason to send 75,000 of the same attack to a single destination, unless it is a flood-type denial of service.

In continuing that thought, I took a look at the time frame of the alerts.

srcip	MIN(time)	MAX(time)
MY.NET.8.1	2001-12-20 00:00:36	2001-12-21 15:59:36
67.161.190.60	2001-12-26 13:50:46	2001-12-26 13:50:59
66.168.137.179	2001-12-24 10:57:12	2001-12-24 12:07:56
65.2.208.87	2001-12-26 20:30:17	2001-12-26 20:30:39
24.8.58.167	2001-12-22 13:00:16	2001-12-22 13:00:17
24.34.101.43	2001-12-25 11:25:00	2001-12-25 11:25:00

From this, we can see that each set of alerts occurred in a relatively short time frame. We still cannot tell from what we have whether these were attacks or not, so I'll continue researching. Each source seems to be unrelated, so I will be looking at each Source IP and Destination IP set individually to determine active targeting.

A3.7.1.1. Top Talker: MY.NET.8.1 75,162 of 75,653 alerts (99.35%)

We don't know when the alerts with source of MY.NET.8.1 began, since it looks quite likely that the alerts were already occurring before the beginning of 12/20/2001 (the first day of this analysis). But we do know that it ended some 40 hours later at 4pm on 12/21/2001. I would like more information regarding the machines at MY.NET.8.1 and MY.NET.16.42 before making a statement whether this was an attack or not. On one hand, it did go on continuously for at least 40 hours, so it could have been a denial of service attack. On the other hand, I stopped abruptly at 4pm on the second day of the analysis period – was the attack discovered and stopped? Were one of these machines malfunctioning and finally fixed or rebooted?

A3.7.1.1.1. Investigate the source address: MY.NET.8.1

srcip	dstip	min(time)	max(time)	count(*)
MY.NET.8.1	MY.NET.16.42	2001-12-20 00:00:36	2001-12-21 15:59:36	75162

To try to answer these questions myself, I took a look through the data to see if there were any other alerts involving either the source or destination IP address. The source, MY.NET.8.1 came up fairly clean, with no other alerts where MY.NET.8.1 was the source IP address and only two alerts where MY.NET.8.1 was the destination IP address. The two alerts where MY.NET.8.1 was a destination IP address were SYN/FIN scans to the FTP port and SSH port:

time	alert	srcip	srcport	dstip	dstport

```
+---------------------+--------------+-------------+---------+------------+---------+
| 2001-12-20 08:16:57 | SYN-FIN scan! | 62.211.247.3 | 21      | MY.NET.8.1 | 21      |
| 2001-12-25 21:51:14 | SYN-FIN scan! | 24.0.28.234  | 22      | MY.NET.8.1 | 22      |
+---------------------+--------------+-------------+---------+------------+---------+
```

This tells us that MY.NET.8.1 is accessible from the Internet, but little else because these scans occurred well after the "Tiny Fragments…" alerts began.

A3.7.1.1.2. Investigate the destination address

However, we can see that there is much more activity related to the destination of MY.NET.16.42.
The following alerts were logged where MY.NET.16.42 is the source IP address.

```
+-------------------------------------------------+-------------+---------+----------------+-
--------+----------+
| alert                                           | srcip       | srcport | dstip          |
dstport | count(*) |
+-------------------------------------------------+-------------+---------+----------------+-
--------+----------+
| High port 65535 tcp - possible Red Worm - traffic  | MY.NET.16.42 | 65535   | MY.NET.11.4    |
80      |        2 |
| Port 55850 tcp - Possible myserver activity - ref. | MY.NET.16.42 | 55850   | MY.NET.11.4    |
80      |       16 |
| Port 55850 udp - Possible myserver activity - ref. | MY.NET.16.42 | 55850   | MY.NET.1.3     |
53      |        2 |
| Possible trojan server activity                 | MY.NET.16.42 | 27374   | MY.NET.70.183  |
80      |        2 |
| spp_http_decode: IIS Unicode attack detected    | MY.NET.16.42 | 13120   | MY.NET.11.4    |
80      |        2 |
| spp_http_decode: IIS Unicode attack detected    | MY.NET.16.42 | 39756   | MY.NET.111.140 |
80      |        7 |
+-------------------------------------------------+-------------+---------+----------------+-
--------+----------+
```

And we have the following alerts where MY.NET.16.42 is the destination IP address (excluding the "Tiny Fragments…" alerts):

```
+-------------------------------------------------+--------------+---------+--------------+--
-------+----------+
| alert                                           | srcip        | srcport | dstip        |
dstport | count(*) |
+-------------------------------------------------+--------------+---------+--------------+--
-------+----------+
| High port 65535 tcp - possible Red Worm - traffic  | MY.NET.11.4   | 80      | MY.NET.16.42 |
65535   |        3 |
| ICMP Destination Unreachable (Communication Admini | MY.NET.16.13  | NULL    | MY.NET.16.42 |
NULL    |        7 |
| ICMP Destination Unreachable (Communication Admini | MY.NET.16.26  | NULL    | MY.NET.16.42 |
NULL    |       16 |
| Port 55850 tcp - Possible myserver activity - ref. | MY.NET.11.4   | 80      | MY.NET.16.42 |
55850   |       14 |
| Port 55850 tcp - Possible myserver activity - ref. | MY.NET.162.67 | 80      | MY.NET.16.42 |
55850   |        6 |
| Port 55850 tcp - Possible myserver activity - ref. | MY.NET.70.38  | 80      | MY.NET.16.42 |
55850   |        5 |
| Possible trojan server activity                 | MY.NET.70.183 | 80      | MY.NET.16.42 |
27374   |        3 |
| SUNRPC highport access!                         | MY.NET.11.4   | 80      | MY.NET.16.42 |
32771   |        9 |
+-------------------------------------------------+--------------+---------+--------------+--
-------+----------+
```

Although these are quite a few alerts for a single workstation, we can see that most of these alerts

are probably false positives. The alerts where MY.NET.16.42 is the source and destination are almost a mirror of each other. Furthermore, nearly all of the alerts occurred where MY.NET.16.42 made an HTTP connection to a local web server (seven different ones here) while using an ephemeral source port that has been known to be used in attacks. There is also a DNS query using an ephemeral source port and a few ICMP Destination Unreachable messages.

A3.7.1.1.3. Conclusion and defensive recommendation: Probably not an attack, more likely something configured wrong, but we need more information to be sure.

I can not tell for certain whether the "Tiny Fragments…" alerts from MY.NET.8.1 to MY.NET.16.42 were malicious or not because we do not know the threshold set on the minfrag preprocessor. The threshold setting on the minfrag preprocessor is the first thing we'll want to check. If the threshold is greater than 256, we may want to change it to a lower setting, 256 bytes, or even 128 bytes depending on whether either computer or any intermediary networking equipment has an MTU setting that low. This means we don't know the size of the packets that raised the alerts, which would weigh heavily in the determination of whether this was malicious activity or not (if the packet size is very small, it's more likely to be an attack).

Another limitation is that we do not know anything about the content of the payload in these packets. Similar to the size of the packets, the payload would tell us much more on whether or not this was an attack.

Now that we know MY.NET.16.42 is a workstation, it seems unlikely that there would be much gained from performing a flooding denial of service attack against it. Rather I would suggest taking a look at MY.NET.8.1, check that the computer does not have an unusually low MTU set and check the hardware, perhaps a faulty network card? Also, if there is any intermediary networking equipment between MY.NET.8.1 and MY.NET.16.42, we would want to check the MTU on those items as well.

A3.7.1.2. Top Talker #2: 67.161.190.60 223 of 75,653 alerts (0.29%)

srcip	dstip	min(time)	max(time)	count(*)
67.161.190.60	MY.NET.99.39	2001-12-26 13:50:46	2001-12-26 13:50:59	223
65.2.208.87	MY.NET.99.39	2001-12-26 20:30:17	2001-12-26 20:30:39	190

From the above table, we can see that the alerts with source of 67.161.190.60 took place in a span of 13 seconds, however we must remember that the alerts from 65.2.208.87 were to the same destination, which makes it suspicious. That table shows the alerts from 67.161.190.60 occurred before the alerts from 65.2.208.87, so it could mean that the earlier set of alerts were a reconnaissance mission, the actual attack to take place later from another source at a later time.

A3.7.1.2.1. Investigate the source address: 67.161.190.60.

I did a little research on the offending source IP address, which proved to be a part of the @Home network, which is not surprising as cable modem users are a favorite target of attackers:

> Trying whois -h whois.arin.net 67.161.190.60
> @Home Network (NETBLK-HOME-6BLK)HOME-6BLK 67.160.0.0 -

67.175.255.255
@Home Network (NETBLK-NWRKNJ1-NJ-19) NWRKNJ1-NJ-19
67.161.128.0 - 67.161.191.255

I ran a query to see if this source IP address caused any other alerts:

```
+-----------------------------------------+--------------+--------------+----------+
| alert                                   | srcip        | dstip        | count(*) |
+-----------------------------------------+--------------+--------------+----------+
| Null scan!                              | 67.161.190.60 | MY.NET.99.39 |       18 |
| Tiny Fragments - Possible Hostile Activity | 67.161.190.60 | MY.NET.99.39 |      223 |
+-----------------------------------------+--------------+--------------+----------+
```

It appears that 67.161.190.60 also ran a "Null scan" against the same destination IP address. So I took a look at the timing of the two different types of alerts:

```
+------------+---------------+--------------+---------------------+---------------------+
| alert      | srcip         | dstip        | min(time)           | max(time)           |
+------------+---------------+--------------+---------------------+---------------------+
| Null scan! | 67.161.190.60 | MY.NET.99.39 | 2001-12-26 13:50:46 | 2001-12-26 13:50:58 |
+------------+---------------+--------------+---------------------+---------------------+

+-----------------------------------------+--------------+--------------+---------------------
+---------------------+
| alert                                   | srcip         | dstip        | min(time)           |
max(time)           |
+-----------------------------------------+--------------+--------------+---------------------
+---------------------+
| Tiny Fragments - Possible Hostile Activi | 67.161.190.60 | MY.NET.99.39 | 2001-12-26 13:50:46 |
2001-12-26 13:50:59 |
+-----------------------------------------+--------------+--------------+---------------------
+---------------------+
```

The times of the alerts are almost identical, which makes it look like both of these alerts are part of a single fingerprinting session of the MY.NET.99.39 machine. In the previously cited email from Marty Roesch, he explains that nmap and fragrouter use fragment sizes of 8 or 24 bytes (Roesch). Nmap is a well known fingerprinting tool, the Null scan is an option in nmap.

It's likely that the 67.161.190.60 machine was a compromised host used by an attacker to fingerprint the machine at MY.NET.99.39. The attacker may use a separate compromised host for attacking.

A3.7.1.2.2. Investigate the destination address: MY.NET.99.39.

The first thing that interested me about this machine is that it was the recipient of this alert from two different sources. I did some further research on MY.NET.99.39 to try to find out the role of the machine. Here is a list of all alerts where MY.NET.99.39 is the source IP address:

```
+--------------------------------------+--------------+---------+-----------------+---------+--
--------+
| alert                                | srcip        | srcport | dstip           | dstport |
count(*) |
+--------------------------------------+--------------+---------+-----------------+---------+--
--------+
| ICMP Echo Request CyberKit 2.2 Windows | MY.NET.99.39 | NULL    | 204.71.200.33   | NULL
|        2 |
| ICMP Echo Request CyberKit 2.2 Windows | MY.NET.99.39 | NULL    | 204.71.200.34   | NULL    |
       1 |
| ICMP Echo Request CyberKit 2.2 Windows | MY.NET.99.39 | NULL    | 216.136.175.143 | NULL
|        1 |
```

```
| ICMP traceroute                         | MY.NET.99.39 | NULL    | 202.139.107.111 | NULL
|        1 |
| ICMP traceroute                         | MY.NET.99.39 | NULL    | 202.139.87.61   | NULL
|        1 |
| ICMP traceroute                         | MY.NET.99.39 | NULL    | 204.152.107.233 | NULL
|        1 |
| ICMP traceroute                         | MY.NET.99.39 | NULL    | 204.152.47.99   | NULL
|        1 |
| ICMP traceroute                         | MY.NET.99.39 | NULL    | MY.NET.16.14    | NULL
|        4 |
| INFO MSN IM Chat data                   | MY.NET.99.39 | 4566    | 64.4.12.158     | 1863
|        2 |
| INFO MSN IM Chat data                   | MY.NET.99.39 | 4228    | 64.4.12.167     | 1863
|       13 |
| INFO MSN IM Chat data                   | MY.NET.99.39 | 4573    | 64.4.12.173     | 1863
|        6 |
+-----------------------------------------+--------------+---------+-----------------+---------+--
--------+
```

And here is a list of all alerts where MY.NET.99.39 is the destination IP address:

```
+-----------------------------------------+----------------+---------+--------------+--------
-+----------+
| alert                                   | srcip          | srcport | dstip        | dstport
| count(*) |
+-----------------------------------------+----------------+---------+--------------+--------
-+----------+
| INFO MSN IM Chat data                   | 64.4.12.158    | 1863    | MY.NET.99.39 | 4566
|        3 |
| INFO Outbound GNUTella Connect accept   | 24.185.240.31  | 6346    | MY.NET.99.39 | 2043
|        1 |
| Null scan!                              | 216.26.218.12  | 31439   | MY.NET.99.39 | 20292
|        3 |
| Null scan!                              | 63.156.28.5    | 6383    | MY.NET.99.39 | 65533
|        3 |
| Null scan!                              | 63.156.39.187  | 5420    | MY.NET.99.39 | 10048
|        3 |
| Null scan!                              | 63.159.104.10  | 48669   | MY.NET.99.39 | 20452
|        3 |
| Null scan!                              | 65.129.144.33  | 0       | MY.NET.99.39 | 0
|        3 |
| Null scan!                              | 65.129.88.87   | 21922   | MY.NET.99.39 | 20276
|        3 |
| Null scan!                              | 65.149.131.233 | 44986   | MY.NET.99.39 | 10978
|        3 |
| Null scan!                              | 65.2.208.87    | 0       | MY.NET.99.39 | 0
|       14 |
| Null scan!                              | 65.42.131.199  | 0       | MY.NET.99.39 | 0
|        2 |
| Null scan!                              | 67.161.190.60  | 0       | MY.NET.99.39 | 0
|       18 |
| Queso fingerprint                       | 128.93.24.104  | 46640   | MY.NET.99.39 | 1214
|        1 |
| SCAN Synscan Portscan ID 19104          | 132.239.25.137 | 3940    | MY.NET.99.39 | 1214
|        1 |
| SCAN Synscan Portscan ID 19104          | 172.138.110.237| 3272    | MY.NET.99.39 | 1214
|        1 |
| SCAN Synscan Portscan ID 19104          | 64.40.60.89    | 3992    | MY.NET.99.39 | 1214
|        1 |
| Tiny Fragments - Possible Hostile Activity | 65.2.208.87 | NULL    | MY.NET.99.39 | NULL
|      190 |
| Tiny Fragments - Possible Hostile Activity | 67.161.190.60 | NULL  | MY.NET.99.39 | NULL
|      223 |
| Watchlist 000220 IL-ISDNNET-990517      | 212.179.127.20 | 2204    | MY.NET.99.39 | 1214
|       15 |
| Watchlist 000220 IL-ISDNNET-990517      | 212.179.21.175 | 1073    | MY.NET.99.39 | 1214
|      174 |
| Watchlist 000220 IL-ISDNNET-990517      | 212.179.35.118 | 48707   | MY.NET.99.39 | 1214
|       30 |
| Watchlist 000220 IL-ISDNNET-990517      | 212.179.47.71  | 1184    | MY.NET.99.39 | 1214
```

```
|        9 |
+-----------------------------------------+----------------+---------+-------------+--------
-+----------+
```

We do not see any alerts that would define MY.NET.99.39 as a server (at least not intentionally). We do see that this computer made outbound MSN Chat connections to several machines in the MS Hotmail block (64.4.0.0 - 64.4.63.255), which would make it look like this computer is a workstation. We also see that this computer is definitely connected to the Internet and has received many scans and even attempts to create KaZaA/Morpheus connections (those with destination port of 1214).

A3.7.1.2.3. Conclusion and defensive recommendation: Not an RFC 1858 tiny fragments attack, but the Destination machine has been compromised.

Although it appears that the "Tiny Fragments…" alerts indicate a fingerprinting, not an attack, it does appear that the machine at MY.NET.99.39 is running a KaZaA or Morpheus server, which shares files in a peer-to-peer methodology, like napster and GNUTella. This tells us that the rules need to be updated on these snort sensors, because we found this alert by accident, not because the KaZaA/Morpheus alert was raised. (However, this rule may not yet have been released. I took a look through the standard rules that I downloaded on 12/10/2001 – this rule was not yet included in that rule set.)

We'll want to take MY.NET.99.39 off the Internet until it can be looked at. When it is examined, we'll want to make sure that the KaZaA server has been removed. Also, we see that the MS Instant Messenger has been used from this machine to connect to a Hotmail server. If Instant Messenger is not allowed in the security policy, we'll want to make sure to disable the Instant Messenger software and block the outgoing TCP destination port of 1863 and incoming TCP source port of 1863.

A3.7.1.3. Top Talker #3: 65.2.208.87 190 of 75,653 alerts (0.25%)

```
+-------------+--------------+---------------------+---------------------+----------+
| srcip       | dstip        | min(time)           | max(time)           | count(*) |
+-------------+--------------+---------------------+---------------------+----------+
| 65.2.208.87 | MY.NET.99.39 | 2001-12-26 20:30:17 | 2001-12-26 20:30:39 |      190 |
+-------------+--------------+---------------------+---------------------+----------+
```

The "Tiny Fragments…" alerts from this source IP address are going to the same destination as the second highest talker (67.161.190.60), so we want to especially investigate whether this source was part of a coordinated attack on the destination of MY.NET.99.39.

A3.7.1.3.1. Investigate the source address: 65.2.208.87.

> Trying whois -h whois.arin.net 65.2.208.87
> @Home Network (NETBLK-HOME-3BLK)HOME-3BLK 65.0.0.0 - 65.15.255.255
> @Home Network (NETBLK-NWRKNJ1-NJ-8) NWRKNJ1-NJ-8 65.2.208.0 - 65.2.223.255

In keeping our coordinated attack theory alive, we find that this machine is also a part of the @Home network, same as Top Talker #2.
I ran the same query to find out if this source IP address caused any other alerts:

alert	srcip	dstip	count(*)
Null scan!	65.2.208.87	MY.NET.99.39	14
Tiny Fragments - Possible Hostile Activity	65.2.208.87	MY.NET.99.39	190

Not surprising that this looks exactly like Top Talker #2 as well. However, this raises a new question: If this source and Top Talker #2 were working together, why would you do the same thing? It no longer appears that Top Talker #2 ran a recon mission for this source, since both did the same thing and both look like a fingerprinting session.

A3.7.1.3.2. Conclusion and defensive recommendation: same as Top Talker #2, fingerprinting session.

Virtually everything here is the same as Top Talker #2. The only thing different is the Source IP address. Since cable modem users often have dynamic IP addresses, and both sources are from the same ISP, it's quite possible that they came from the same computer, but obtaining a different IP address 6.5 hours later.

A3.7.1.4. Top Talker #4: 66.168.137.179 74 of 75,653 alerts (0.1%)

srcip	dstip	min(time)	max(time)	count(*)
66.168.137.179	MY.NET.87.50	2001-12-24 10:57:12	2001-12-24 12:07:56	74

Like the other sets of this alert, the alerts from 66.168.137.179 only went to a single destination: MY.NET.87.50. The most unique thing about this set of alerts is that this set has the most relatively spread out time range, causing 74 alerts in 70 minutes.

A3.7.1.4.1. Investigate the source address: 66.168.137.179.

Trying whois -h whois.arin.net 66.168.137.179
Charter Communications (NETBLK-CHARTER-NET-4BLK) CHARTER-NET-4BLK
66.168.0.0 - 66.169.255.255
Charter Communications (NETBLK-KRNY-NE-66-168-128) KRNY-NE-66-168-128
66.168.128.0 - 66.168.143.255

Since I hadn't previously heard of Charter Communications, I dug further:

whois -h whois.arin.net NETBLK-KRNY-NE-66-168-128
Charter Communications (NETBLK-KRNY-NE-66-168-128)
12405 Powerscourt
St. Louis, MO 63131
US

Netname: KRNY-NE-66-168-128
Netblock: 66.168.128.0 - 66.168.143.255

Coordinator:
Charter Communications (ZC119-ARIN) ipaddressing@chartercom.com
314-965-0555

Domain System inverse mapping provided by:

NS1.CHARTER.COM 24.196.241.11
NS2.CHARTER.COM 24.213.60.79

Record last updated on 15-Nov-2001.
Database last updated on 24-Jan-2002 19:56:53 EDT.

That still didn't tell me if this was an ISP or not, so I took a look at their homepage. Not surprisingly, Charter Communications is a cable modem ISP, who apparently is picking up new customers from the failure of @Home.

I also ran a query to find out if there were any other alerts from this IP address:

alert	srcip	dstip	count(*)
Tiny Fragments - Possible Hostile Activity	66.168.137.179	MY.NET.87.50	74

There were no other alerts with the same source IP address. This makes it more difficult to tell if these alerts constituted an attack. Perhaps researching the Destination address will tell us more.

A3.7.1.4.2. Investigate the destination address: MY.NET.87.50.

MY.NET.87.50 is an internal computer that we haven't researched yet, so I'll first run a couple of queries to find if there were any other alerts involving this computer.
Here is a list of alerts where MY.NET.87.50 is the source address:

alert	srcip	dstip	count(*)
ICMP Destination Unreachable (Protocol Unreachable	MY.NET.87.50	24.120.88.171	5
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	12.238.165.82	58
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	195.249.246.249	170
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	208.199.16.106	137
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	213.107.126.106	61
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	217.136.207.171	31
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	24.101.41.96	26
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	24.148.73.203	142
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	24.180.139.254	129
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	24.2.230.96	3

ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	24.2.93.28	17
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	24.205.76.45	88
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	24.21.226.73	101
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	24.23.140.185	1
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	24.242.216.94	46
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	24.65.82.75	83
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	24.69.15.231	24
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	24.93.75.181	96
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	62.238.37.227	614
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	64.208.160.107	13
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	65.11.188.225	65
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	65.30.118.240	32
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	65.7.224.113	1
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	65.8.111.91	27
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	65.81.147.61	25
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	65.9.34.41	190
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	66.75.1.72	161
ICMP Fragment Reassembly Time Exceeded	MY.NET.87.50	80.135.99.196	55

A list of alerts where MY.NET.87.50 is the destination address:

alert	srcip	dstip	count(*)
MISC Large UDP Packet	12.224.81.18	MY.NET.87.50	1
MISC Large UDP Packet	12.228.112.157	MY.NET.87.50	1
MISC Large UDP Packet	134.173.166.103	MY.NET.87.50	1
MISC Large UDP Packet	141.157.93.94	MY.NET.87.50	1
MISC Large UDP Packet	162.83.196.215	MY.NET.87.50	1
MISC Large UDP Packet	172.156.228.213	MY.NET.87.50	1
MISC Large UDP Packet	210.101.117.69	MY.NET.87.50	1
MISC Large UDP Packet	210.50.61.12	MY.NET.87.50	1
MISC Large UDP Packet	217.1.82.238	MY.NET.87.50	1
MISC Large UDP Packet	217.125.115.81	MY.NET.87.50	1
MISC Large UDP Packet	217.96.13.212	MY.NET.87.50	1
MISC Large UDP Packet	24.100.50.113	MY.NET.87.50	11
MISC Large UDP Packet	24.11.89.184	MY.NET.87.50	1
MISC Large UDP Packet	24.128.200.18	MY.NET.87.50	1
MISC Large UDP Packet	24.158.75.105	MY.NET.87.50	1
MISC Large UDP Packet	24.161.120.40	MY.NET.87.50	1
MISC Large UDP Packet	24.164.227.253	MY.NET.87.50	1
MISC Large UDP Packet	24.166.76.13	MY.NET.87.50	1
MISC Large UDP Packet	24.169.131.16	MY.NET.87.50	2
MISC Large UDP Packet	24.200.144.143	MY.NET.87.50	1
MISC Large UDP Packet	24.24.36.181	MY.NET.87.50	1
MISC Large UDP Packet	24.245.20.166	MY.NET.87.50	1
MISC Large UDP Packet	24.254.241.95	MY.NET.87.50	1
MISC Large UDP Packet	24.36.220.222	MY.NET.87.50	6
MISC Large UDP Packet	24.47.72.191	MY.NET.87.50	1

MISC Large UDP Packet	24.70.132.55	MY.NET.87.50	2
MISC Large UDP Packet	24.76.1.122	MY.NET.87.50	1
MISC Large UDP Packet	24.82.242.243	MY.NET.87.50	1
MISC Large UDP Packet	24.91.169.207	MY.NET.87.50	1
MISC Large UDP Packet	24.96.54.14	MY.NET.87.50	1
MISC Large UDP Packet	4.62.138.219	MY.NET.87.50	1
MISC Large UDP Packet	65.8.111.91	MY.NET.87.50	1
MISC Large UDP Packet	65.80.29.197	MY.NET.87.50	1
MISC Large UDP Packet	66.108.250.66	MY.NET.87.50	1
MISC Large UDP Packet	66.156.142.224	MY.NET.87.50	1
MISC Large UDP Packet	66.157.75.88	MY.NET.87.50	1
MISC Large UDP Packet	66.188.110.190	MY.NET.87.50	1
MISC Large UDP Packet	66.190.93.40	MY.NET.87.50	13
MISC Large UDP Packet	66.26.12.7	MY.NET.87.50	1
MISC Large UDP Packet	67.174.68.191	MY.NET.87.50	4
MISC PCAnywhere Startup	141.157.93.94	MY.NET.87.50	1
MISC PCAnywhere Startup	24.180.10.152	MY.NET.87.50	2
MISC traceroute	202.102.138.2	MY.NET.87.50	4
NMAP TCP ping!	61.221.200.2	MY.NET.87.50	2
NMAP TCP ping!	65.193.73.247	MY.NET.87.50	1
SYN-FIN scan!	62.211.247.3	MY.NET.87.50	1
Tiny Fragments - Possible Hostile Activity	66.168.137.179	MY.NET.87.50	74

We find that the MY.NET.87.50 destination address has quite a few alerts from quite a few sources. The ICMP fragment reassembly time exceeded messages tells us this computer is definitely receiving many fragments. The ICMP message is a response that occurs if the received fragments are not correctly assembled within a timeout period. Although these findings tell us MY.NET.87.50 is certainly dealing with many fragments, it doesn't tell us what MY.NET.87.50 is doing. Luckily this address shows up again in our analysis. In the analysis of the scans, we find that MY.NET.87.50 is the biggest user of several ports, namely UDP 27005, which is the Half Life client and UDP 27500, which is Quake.

A3.7.1.4.3. Conclusion and defensive recommendation: Tiny fragments are not an attack, but investigate use of MY.NET.87.50.

We've been able to find that the destination address in these alerts (MY.NET.87.50) is an avid on-line gamer. These games deal in UDP packets that are often large, needing to be fragmented. Almost certainly the Tiny fragments alerts were raised because there were fragments of small size, but not from an attack, just from the fact that there are many fragments going to this destination, some of them will be smaller than the minfrag threshold value.

Even though this isn't an attack, we should stop the practice of playing online games, through policy and enforce it through filtering at the routers and firewalls, blocking the appropriate ports.

A3.8. Severity:

The severity formula is not helpful in this case because we need more information for three of the four factors that determine the severity. I do not know the target criticality, system countermeasures, or network countermeasures.
As such, I'll grade only the attack lethality factor.

Lethality = 1, none of the above appears to be an attack, lethal or otherwise.

A3.9. Defensive recommendations:

We've indirectly found an internal computer that is a KaZaA server and another internal computer that is used for Internet gaming.
We'll want to take MY.NET.99.39 off the Internet until it can be looked at. When it is examined, we'll want to make sure that the KaZaA server has been removed. Also, we see that the MS Instant Messenger has been used from this machine to connect to a Hotmail server. If Instant Messenger is not allowed in the security policy, we'll want to make sure to disable the Instant Messenger software and block the outgoing TCP destination port of 1863 and incoming TCP source port of 1863.

We've been able to find that the destination address in these alerts (MY.NET.87.50) is an avid on-line gamer. These games deal in UDP packets that are often large, needing to be fragmented. Almost certainly the Tiny fragments alerts were raised because there were fragments of small size, but not from an attack, just from the fact that there are many fragments going to this destination, some of them will be smaller than the minfrag threshold value.

Even though this isn't an attack, we should stop the practice of playing online games, through policy and enforce it through filtering at the routers and firewalls, blocking the appropriate ports.

A3.10. Multiple choice question:

In the RFC 1858 Tiny fragments attack, the goal is to split which of the following to allow the packet to pass through firewalls:

A. Ethernet header
B. IP header
C. TCP header
D. ARP header

Answer: C, the TCP header (I don't think there is any such thing as an ARP header, and a problem in the Ethernet header or IP header would probably make the packet undeliverable).

A4. Appendix 4: Watchlist 000220 IL-ISDNNET-990517 (#3 of 149)

This alert is the third most frequent for the analysis period, and certainly contains more potential for attack than "MISC traceroute", the second most frequent alert.

A4.1. Source of trace:

"Analyze this" data from 12/20/2001 through 12/26/2001

A4.2. Detect was generated by:

Snort IDS

A4.3. Probability the source address was spoofed:

Unlikely since this is a custom rule that watches all traffic from a certain network, not watching for a specific attack.

A4.4. Description of the attack:

A4.4.1. Definition of the Watchlist 000220 IL-ISDNNET-990517 alert:

The "Watchlist 000220 IL-ISDNNET-990517" alert is a custom rule, which is not in the most current standard rule set. This alert is raised any time there is activity from the Israeli ISP: IL-ISDNNET-990517.

Here is the registration information for this domain:

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      212.179.0.0 - 212.179.255.255
netname:      IL-ISDNNET-990517
descr:        PROVIDER
country:      IL
admin-c:      NP469-RIPE
tech-c:       TP1233-RIPE
tech-c:       ZV140-RIPE
tech-c:       ES4966-RIPE
status:       ALLOCATED PA
mnt-by:       RIPE-NCC-HM-MNT
changed:      hostmaster@ripe.net 19990517
changed:      hostmaster@ripe.net 20000406
changed:      hostmaster@ripe.net 20010402
source:       RIPE

route:        212.179.0.0/17
descr:        ISDN Net Ltd.
origin:       AS8551
notify:       hostmaster@isdn.net.il
mnt-by:       AS8551-MNT
changed:      hostmaster@isdn.net.il 19990610
source:       RIPE

person:       Nati Pinko
address:      Bezeq International
address:      40 Hashacham St.
address:      Petach Tikvah  Israel
phone:        +972 3 9257761
e-mail:       hostmaster@isdn.net.il
nic-hdl:      NP469-RIPE
```

changed: registrar@ns.il 19990902
source: RIPE

person: Tomer Peer
address: Bezeq International
address: 40 Hashakham St.
address: Petakh Tiqwah Israel
phone: +972 3 9257761
e-mail: hostmaster@isdn.net.il
nic-hdl: TP1233-RIPE
changed: registrar@ns.il 19991113
source: RIPE

person: Zehavit Vigder
address: bezeq-international
address: 40 hashacham
address: petach tikva 49170 Israel
phone: +972 52 770145
fax-no: +972 9 8940763
e-mail: hostmaster@bezeqint.net
nic-hdl: ZV140-RIPE
changed: zehavitv@bezeqint.net 20000528
source: RIPE

person: Eran Shchori
address: BEZEQ INTERNATIONAL
address: 40 Hashacham Street
address: Petach-Tikva 49170 Israel
phone: +972 3 9257710
fax-no: +972 3 9257726
e-mail: hostmaster@bezeqint.net
nic-hdl: ES4966-RIPE
changed: registrar@ns.il 20000309
source: RIPE

A4.5. Attack mechanism:

Since this rule is watching a network, not a specific attack, there is no pre-defined attack mechanism.
The things we want to know are: Who is attacking? and What are they doing?

A4.5.1. Who is attacking? (Investigate sources)

Here is a list of the top 10 source addresses and the number of time each raised this alert:

```
+-----------------+-------+
| srcip           | count |
+-----------------+-------+
```

```
| 212.179.35.118  | 61330 |
| 212.179.27.6    |   376 |
| 212.179.38.210  |   273 |
| 212.179.79.2    |   232 |
| 212.179.21.175  |   174 |
| 212.179.68.65   |   130 |
| 212.179.112.100 |   121 |
| 212.179.45.75   |    94 |
| 212.179.48.194  |    35 |
| 212.179.126.3   |    27 |
+-----------------+-------+
```

Of a total 62,991 alerts, 61,330 were from a single source address: 212.179.35.118. That's 97.36% of these alerts.
My plan is to find out what this source is doing and discount the others as too small to worry about (we'd be better off spending the time investigating other alerts).

A4.5.2. What is the attacker doing? (Investigate ports and destinations)

The source address of 212.179.35.118 has been singled out for investigation in this appendix. The next task is to figure out what that address was doing.

The first query is to find out if 212.179.35.118 raised any other alerts and get a count for the number of times it hit each destination address:

```
+-----------------------------------+--------------+----------
+
| alert                             | dstip        | count(*)
|
+-----------------------------------+--------------+----------
+
| Watchlist 000220 IL-ISDNNET-990517 | MY.NET.70.192 |        2
|
| Watchlist 000220 IL-ISDNNET-990517 | MY.NET.70.70  |    61295
|
| Watchlist 000220 IL-ISDNNET-990517 | MY.NET.75.145 |        3
|
| Watchlist 000220 IL-ISDNNET-990517 | MY.NET.99.39  |       30
|
+-----------------------------------+--------------+----------
+
```

We now know that the source 212.179.35.118 did not trip any other alerts and nearly all of the alerts have the destination address of MY.NET.70.70. I also ran a query to find out if there were any alerts where 212.179.35.118 were the destination address, but there were no occurrences.

The next query adds the destination port into the query, so we have some idea of what the attacker is trying to do.

```
+-----------------------------------+--------------+---------
+----------+
| alert                             | dstip        | dstport |
count(*) |
+-----------------------------------+--------------+---------
+----------+
| Watchlist 000220 IL-ISDNNET-990517 | MY.NET.70.192 | 3232
|        2 |
| Watchlist 000220 IL-ISDNNET-990517 | MY.NET.70.70  | 1214
|    61295 |
| Watchlist 000220 IL-ISDNNET-990517 | MY.NET.75.145 | 4054
|        3 |
| Watchlist 000220 IL-ISDNNET-990517 | MY.NET.99.39  | 1214
|       28 |
| Watchlist 000220 IL-ISDNNET-990517 | MY.NET.99.39  | 2113
|        2 |
+-----------------------------------+--------------+---------
+----------+
```

We see from this query that this attacker has found (or created) a KaZaA server on MY.NET.70.70. We see a return of MY.NET.99.39, which we investigated in Appendix 3: Tiny Fragments. We already knew that computer probably is running as a KaZaA server.

We have a good idea of what happened and who it involved, but for the record, I'll get a few more facts.

Adding the source port to the query gives us an idea of how many connections to each destination were made (a new connection would probably be from a different source port):

```
+-----------------------------------+---------+--------------+---------+----------+
| alert                             | srcport | dstip        | dstport | count(*) |
+-----------------------------------+---------+--------------+---------+----------+
| Watchlist 000220 IL-ISDNNET-990517 | 1214    | MY.NET.70.192 | 3232    |        2 |
| Watchlist 000220 IL-ISDNNET-990517 | 1214    | MY.NET.75.145 | 4054    |        3 |
| Watchlist 000220 IL-ISDNNET-990517 | 1214    | MY.NET.99.39  | 2113    |        2 |
| Watchlist 000220 IL-ISDNNET-990517 | 33952   | MY.NET.99.39  | 1214    |        7 |
| Watchlist 000220 IL-ISDNNET-990517 | 48707   | MY.NET.99.39  | 1214    |        7 |
| Watchlist 000220 IL-ISDNNET-990517 | 49444   | MY.NET.99.39  | 1214    |        1 |
| Watchlist 000220 IL-ISDNNET-990517 | 52529   | MY.NET.99.39  | 1214    |        5 |
| Watchlist 000220 IL-ISDNNET-990517 | 52915   | MY.NET.99.39  | 1214    |        4 |
| Watchlist 000220 IL-ISDNNET-990517 | 54368   | MY.NET.99.39  | 1214    |        4 |
| Watchlist 000220 IL-ISDNNET-990517 | 60339   | MY.NET.70.70  | 1214    |    61295 |
+-----------------------------------+---------+--------------+---------+----------+
```

All of the alerts where MY.NET.70.70 occurred from a single source port, so it was almost certainly a single connection unless the attacker was able to connect multiple times while using the same source port of 60339. It's not impossible, but unlikely, since there isn't anything to gain by manipulating the source port in this situation.

A look at the times of the alerts gives us a timeframe:

```
+--------------+---------+--------------------+---------------
-----+----------+
| dstip        | dstport | min(time)          | max
(time)           | count(*) |
+--------------+---------+--------------------+---------------
-----+----------+
| MY.NET.70.70 | 1214    | 2001-12-25 15:47:49 | 2001-12-26
10:51:50 |    61295 |
+--------------+---------+--------------------+---------------
-----+----------+
```

A single connection spanning 19 hours.

A4.6. Correlations:

The "Watchlist 000220 IL-ISDNNET-990517" alert is found in most of the previous practicals: Rick Yuen reported similar findings, including the TCP 1214 port as the main offender during the period of Sept. 18, 2001 – Sept. 22, 2001 (http://www.giac.org/practical/Rick_Yuen_GCIA.doc).
Jeff Holland also found this as one of the top alerts for the period of April 13, 2001 – April 19, 2001 (http://www.giac.org/practical/Jeff_Holland_GCIA.doc).
Lloyd Webb found this to be the second most frequent alert for the period of August 25, 2001 – August 29, 2001 (http://www.giac.org/practical/Lloyd_Webb_GCIA.doc).

A4.7. Evidence of active targeting:

Nearly all of the traffic for this alert is between a single source address and a single destination address, probably on a single connection spanning 19 hours. That's about as specific as you can possibly get, this was definitely a case of active targeting.

A4.8. Severity:

The severity formula is not helpful in this case because we need more information for three of the four factors that determine the severity. I do not know the target criticality, system countermeasures, or network countermeasures.
As such, I'll grade only the attack lethality factor.

Lethality = 5, We found a compromised system that is running a file sharing server. Such a server is itself open to attack, as well as being a good staging point for further attacks like trojan horses, viruses and worms.

A4.9. Conclusion and defensive recommendation:

We've found another computer that has been compromised, where a KaZaA server has been installed. In Appendix 3, it was MY.NET.39.99, here it is MY.NET.70.70. Both computers should be taken off the network until the KaZaA servers can be removed.

More importantly, the preventive measures that should be taken include: blocking the KaZaA port (TCP 1214) at the routers and firewalls and updating the snort definitions so they include

the rule to spot KaZaA servers. This is the second one we found indirectly, so it's likely that there are even more that we don't know about yet.

A5. Appendix 5: BACKDOOR NetMetro File List (#17 of 149) [False Positive]

This is the seventeenth most frequent attack, but the word "backdoor" draws my attention immediately as something we should check out.

A5.1. Source of trace:

"Analyze this" data from 12/20/2001 through 12/26/2001

A5.2. Detect was generated by:

Snort IDS

A5.3. Probability the source address was spoofed:

Unlikely since this appears to be a file transfer alert. If the attacker wants the files, he needs to provide a correct source address.

A5.4. Description of the attack:

A5.4.1. Definition of "BACKDOOR NetMetro File List"

It was rather difficult to find any real information on how the NetMetro trojan horse works. The best I could do was find a little information on what features it offers. From the Dark Eclipse Software website, I was able to find a feature list which looks similar to the NetBus trojan feature list, where an attacker can stealthily capture keyboard entries as well as be a nuisance, opening/closing the CD-ROM, swap mouse buttons, and show images (Net Metropolitan 1.00).

The following is the snort rule that was raised by this alert:

backdoor.rules:alert tcp $HOME_NET any -> $EXTERNAL_NET 5032 (msg:"BACKDOOR NetMetro File List"; flags: A+; content:"|2D 2D|"; reference:arachnids,79; sid:159; classtype:misc-activity; rev:3;)

We can see that the alert is raised when there is traffic going to an external IP address on TCP port 5032, where there is at least the ACK flag and the content contains "2D 2D" in the payload.

TCP port 5032 is an ephemeral port, which means it will naturally occur in client-side connections. For example, when you connect to an HTTP server, you know the destination port is going to be TCP 80, but you don't normally have control of what the source port will be. There is nothing stopping your computer from choosing TCP 5032 as the source port, which will become the destination port when the HTTP server responds to your request.

The "2D 2D" content in the payload is also something that can naturally occur. Taking a look at

my unicode chart, I see that 2D is the dash character "-". Therefore, the content part of this alert will raise whenever there are two consecutive dashes in the content. Most obvious opportunity for this to be a false alarm is a separating line made up of dashes, like the following: "----------". Therefore, any text only file with lines, tables, or borders will likely raise this alert in a false positive.

Both of these conditions can occur naturally, when you AND them together, you will get far less false positives, but we have a very large network here, so we must expect at least some of these occurrences to be false positives. For example, a false positive will occur if an internal computer is an HTTP server, an external client makes a connection with the source port of TCP 5032, and the HTTP server sends a page with at least two consecutive dashes. More specifically, if the HTTP server were to display the INSTALL file from the Snort 1.8.4 source code, the following lines would result in a false positive:

Snort Configure-time switches
==========

`--enable-debug'
Enable debugging options (bugreports and developers only).

The two dashes before "enable-debug" would raise the content condition.

A5.5. Attack mechanism:

The NetMetro trojan horse is a remote control application that runs on Windows computers from Windows 95 through Windows 2000. If an attacker can trick a victim into running the NetMetro server application, the attacker will be able to do almost anything he wants to the victim's computer including opening a file manager. This alert is raised when files are transferred using NetMetro.

However, in the description section we've proven that there is a potential for false positives, we need to do more investigation to find out whether these are attacks or false positives. This investigation is performed in the active targeting section.

A5.6. Correlations:

John Jenkinson also found the "BACKDOOR NetMetro File List" alert in his analysis of the alerts for Sept. 29, 2001 – Oct. 3, 2001 (http://www.giac.org/practical/John_Jenkinson_GCIA.doc). Jenkinson is the only correlation I could find for this alert.

A5.7. Evidence of active targeting:

Certainly specific, in that all of the alerts are between a single source and destination, but it appears to be a false positive.

A5.7.1. Address analysis

The following query shows that there is a single source and destination address for all of the

3,586 reported alerts:

```
+--------------+--------------+----------+
| srcip        | dstip        | count(*) |
+--------------+--------------+----------+
| MY.NET.60.11 | 209.49.12.32 |     3586 |
+--------------+--------------+----------+
```

A5.7.2. Port analysis

The following query shows that when we take the port numbers into account, we find that this is probably a false positive, but may be a different alert, not just normal traffic:

```
+--------------+---------+--------------+---------+----------+
| srcip        | srcport | dstip        | dstport | count(*) |
+--------------+---------+--------------+---------+----------+
| MY.NET.60.11 | 20      | 209.49.12.32 | 5032    |     3586 |
+--------------+---------+--------------+---------+----------+
```

Rather than a NetMetro file transfer, this looks like an FTP file transfer, the tell-tale sign being the TCP port of 20 on the server end (MY.NET.60.11).

A5.7.3. Analysis of the source address, MY.NET.60.11

I do not know whether an FTP file transfer is permitted or not from the MY.NET.60.11. The best I can do is look further and make an educated guess.

In the following queries, I'm attempting to find out the functionality of the MY.NET.60.11 computer by seeing what other alerts were raised around it.

Where MY.NET.60.11 is the source address, we find this computer to be quite public:

```
+----------------------------+--------------+---------+--------
--+
| alert                      | srcip        | srcport | count
(*) |
+----------------------------+--------------+---------+--------
--+
| BACKDOOR NetMetro File List | MY.NET.60.11 | 20      |
3586 |
| ICMP Echo Request BSDtype   | MY.NET.60.11 | NULL    |
28 |
| INFO Possible IRC Access    | MY.NET.60.11 | 10128   |
28 |
| TELNET login incorrect      | MY.NET.60.11 | 23      |
109 |
+----------------------------+--------------+---------+--------
--+
```

We already know that it is allowing FTP, we find that it is also running a telnet daemon.

A look at alerts where MY.NET.60.11 as the destination address show that it has been fingerprinted a few times and may allow anonymous FTP access:

alert	dstip	dstport	count(*)
EXPLOIT x86 setuid 0	MY.NET.60.11	24921	1
ICMP Destination Unreachable (Communication Admini	MY.NET.60.11	NULL	27
ICMP Destination Unreachable (Fragmentation Needed	MY.NET.60.11	NULL	2
ICMP Destination Unreachable (Host Unreachable)	MY.NET.60.11	NULL	39
ICMP Echo Request BSDtype	MY.NET.60.11	NULL	6
ICMP Echo Request Sun Solaris	MY.NET.60.11	NULL	1
ICMP Source Quench	MY.NET.60.11	NULL	10
ICMP traceroute	MY.NET.60.11	NULL	2
INFO FTP anonymous FTP	MY.NET.60.11	21	2
MISC Large ICMP Packet	MY.NET.60.11	NULL	1
Null scan!	MY.NET.60.11	6399	21
Queso fingerprint	MY.NET.60.11	113	2
SCAN FIN	MY.NET.60.11	23	5
SCAN Proxy attempt	MY.NET.60.11	1080	4
SYN-FIN scan!	MY.NET.60.11	21	1

Another query showed that neither of the anonymous FTP connections were related to the connection from 209.49.12.32, so as far as we can tell, this looks like an allowed service.

A5.7.4. Analysis of the destination address (209.49.12.32):

It appears likely that this is allowed traffic, but we'll want to take a look at who this address belongs to before we write this off as a false positive.

Trying whois -h whois.arin.net 209.49.12.32
Business Internet, Inc. (NET-ICIX-MD-BLK14)
3625 Queen Palm Drive
Tampa, FL 33619
US

Netname: ICIX-MD-BLK14
Netblock: 209.48.0.0 - 209.49.255.255
Maintainer: IMBI

Coordinator:
Business Internet, Inc. (ZI44-ARIN) ipreq@icix.net
240-616-2000

Domain System inverse mapping provided by:

NS.DIGEX.NET 64.245.20.14
NS2.DIGEX.NET 64.245.43.14

Record last updated on 02-Jan-2001.

Database last updated on 10-Feb-2002 19:55:25 EDT.

This is not one of the ISPs that immediately raises a red flag, so this still looks like allowed traffic.

5.8. Severity:

The severity formula is not helpful in this case because we need more information for three of the four factors that determine the severity. I do not know the target criticality, system countermeasures, or network countermeasures.
As such, I'll grade only the attack lethality factor.

Lethality = 1, this appears to be a false positive, alerting on allowed behavior.

5.9. Defensive recommendation:

Since this is not a NetMetro attack, the only thing to check here is whether the source, MY.NET.60.11 is permitted to be an FTP server and to have telnet services available. If those are permitted, then we do not have a problem. If those services should not be available at this computer, then those services should be disabled at the computer and the router ACLs and firewall rules should be updated to not allow connections on those ports.

A6. Appendix 6: connect to 515 from outside (#24 of 149)

This alert was chosen above other more frequent alerts because it sounds like a connection was made, not just a scan. Most of the alerts are probes, but the attacks are most dangerous, so they deserve the most attention.

A6.1. Source of trace:

"Analyze this" data from 12/20/2001 through 12/26/2001

A6.2. Detect was generated by:

Snort IDS

A6.3. Probability the source address was spoofed:

Unlikely since this appears to be a file transfer alert. If the attacker wants the files, he needs to provide a correct source address.

A6.4. Description of the attack:

This is a custom rule, or at least not one of the standard rules in the current rules downloaded on Feb. 2, 2002. From the description I would predict that this alerts on any connections from an external source address to an internal destination address with destination port of TCP 515. Since TCP 515 is the lpd port, this means we have a large chance for false positives unless the security policy is that there should be no outside connections to TCP 515.

There are many exploits of the lpd port, here is a table of the CVE vulnerabilities:

Name	Description
CVE-1999-0299	Buffer overflow in FreeBSD lpd through long DNS hostnames.
CVE-2000-0534	The apsfilter software in the FreeBSD ports package does not properly read user filter configurations, which allows local users to execute commands as the lpd user.
CVE-2001-0353	Buffer overflow in the line printer daemon (in.lpd) for Solaris 8 and earlier allows local and remote attackers to gain root privileges via a "transfer job" routine.
CAN-1999-0061	** CANDIDATE (under review) ** File creation and deletion, and remote execution, in the BSD line printer daemon (lpd).
CAN-2000-0615	** CANDIDATE (under review) ** LPRng 3.6.x improperly installs lpd as setuid root, which can allow local users to append lpd trace and logging messages to files.
CAN-2000-0839	** CANDIDATE (under review) ** WinCOM LPD 1.00.90 allows remote attackers to cause a denial of service by sending a large number of LPD options to the LPD port (515).
CAN-2000-0879	** CANDIDATE (under review) ** LPPlus programs dccsched, dcclpdser, dccbkst, dccshut, dcclpdshut, and dccbkstshut are installed setuid root and world executable, which allows arbitrary local users to start and stop various LPD services.
CAN-2000-1064	** CANDIDATE (under review) ** Buffer overflow in the LPD service in HP JetDirect printer card Firmware x.08.20 and earlier allows remote attackers to cause a denial of service.
CAN-2001-0670	** CANDIDATE (under review) ** Buffer overflow in BSD line printer daemon (in.lpd or lpd) in various BSD-based operating systems allows remote attackers to execute arbitrary code via an incomplete print job followed by a request to display the printer queue.
CAN-2001-0671	** CANDIDATE (under review) ** Buffer overflows in (1) send_status, (2) kill_print, and (3) chk_fhost in lpd in AIX 4.3 and 5.1 allow remote attackers to gain root privileges.
CAN-2001-1002	** CANDIDATE (under review) ** The default configuration of the DVI print filter (dvips) in Red Hat Linux 7.0 and earlier does not run dvips in secure mode when dvips is executed by lpd, which could allow remote attackers to gain privileges by printing a DVI file that contains malicious commands.
CAN-2001-	** CANDIDATE (under review) ** Format string vulnerability in pic utility in groff 1.16.1 and other versions allows remote attackers to

1022	bypass the -S option and execute arbitrary commands via format string specifiers in the plot command.
CAN-2002-0003	** CANDIDATE (under review) ** Buffer overflow in the preprocessor in groff 1.16 and earlier allows remote attackers to gain privileges via lpd in the LPRng printing system.

This is a pretty intimidating list, most of them offer root privileges or execute arbitrary code, quite dangerous. However, we can't be sure these alerts are actually attacks until we do some research in the active targeting section and compare the results with the security policy to make sure we eliminate false positives.

A6.5. Attack mechanism:

There are numerous exploits on the lpd port (as seen in the above CVE table), too many to examine them here. Many of the above exploits are buffer overflows, so we should be watching for signs of buffer overflows when looking for attacks.

A6.6. Correlations:

John Jenkinson also found the "connect to 515 from outside" alert in his analysis of the alerts for Sept. 29, 2001 – Oct. 3, 2001 (http://www.giac.org/practical/John_Jenkinson_GCIA.doc).
Jeff Holland also found the "connect to 515 from outside" alert in his analysis of the alerts for Apr. 13, 2001 – Apr. 19, 2001 (http://www.giac.org/practical/Jeff_Holland_GCIA.doc).
Scott Shinberg also found the "connect to 515 from outside" alert in his analysis of the alerts for June 1, 2001 – June 8, 2001 (http://www.giac.org/practical/Scott_Shinberg_GCIA.doc).

A6.7. Evidence of active targeting:

These alerts turned out to be probes, meaning they are scanning whole class C networks.
The targeting was not specific, but the service appears to be quite specific. These were not TCP port scans where all TCP ports were scanned for each computer, only the TCP 515 port was scanned. I checked this against the alert results and scans results, the findings were consistent – all scans and alerts from the 211.215.16.24 and 62.71.248.52 addresses were to the destination port 515.

A6.7.1. Address analysis

When I ran a query to list the unique destination addresses, it returned 768 of them. It appears I was wrong in my assumption that this alert would not include probes.

When I run a query on the unique source addresses, my fears are confirmed. Of the 1,380 alerts, all came from 4 source addresses, 1,270 from a single address (211.215.16.24).

```
+-----------------+----------+
| srcip           | count(*) |
+-----------------+----------+
| 12.236.143.111  |        6 |
| 211.215.16.24   |     1270 |
```

```
| 255.255.255.255 |         1 |
| 62.71.248.52    |       103 |
+-----------------+-----------+
```

To do further analysis, I'll drop the two least frequently occurring sources: 12.236.143.111 and 255.255.255.255 since those occurred a total of 7 times, opposed to the 1,373 of the other two sources.

A6.7.2. Analysis of the top talker, 211.215.16.24:

The following query shows that the alerts from 211.215.16.24 appear to have scanned some of the MY.NET class C networks:

```
+---------------+------------+----------+
| srcip         | subnet     | count(*) |
+---------------+------------+----------+
| 211.215.16.24 | MY.NET.132 |      241 |
| 211.215.16.24 | MY.NET.133 |      255 |
| 211.215.16.24 | MY.NET.134 |      249 |
| 211.215.16.24 | MY.NET.135 |      234 |
| 211.215.16.24 | MY.NET.137 |      199 |
| 211.215.16.24 | MY.NET.190 |       88 |
| 211.215.16.24 | MY.NET.5.4 |        2 |
| 211.215.16.24 | MY.NET.5.5 |        1 |
| 211.215.16.24 | MY.NET.6.1 |        1 |
+---------------+------------+----------+
```

This gives us a pretty good idea that this source is trying to probe for a specific lpd exploit, so it would be a pretty good idea to find out to whom this address belongs:

> Trying whois -h whois.apnic.net 211.215.16.24
>
> % Rights restricted by copyright. See http://www.apnic.net/db/dbcopyright.html
> % (whois6.apnic.net)
>
> inetnum: 211.212.0.0 - 211.215.255.255
> netname: HANANET
> descr: Hanaro Telecom, Inc.
> country: KR
> admin-c: IS37-AP
> tech-c: SH243-AP
> remarks: **
> remarks: Allocated to KRNIC Member.
> remarks: If you would like to find assignment
> remarks: information in detail please refer to
> remarks: the KRNIC Whois Database at:
> remarks: http://whois.nic.or.kr/english/index.html
> remarks: **

```
mnt-by:      MNT-KRNIC-AP
mnt-lower:   MNT-KRNIC-AP
changed:     hostmaster@apnic.net 20010615
changed:     hostmaster@apnic.net 20010730
source:      APNIC

person:      Inyup Sung
address:     Hanaro Telecom Co.
address:     Kukje Electornics Cneter Bldg. 1445-3 Seocho-Dong Seocho-Ku
address:     SEOUL
address:     137-070
country:     KR
phone:       +82-2-106
fax-no:      +82-2-6266-6483
e-mail:      info@hananet.net
nic-hdl:     IS37-AP
mnt-by:      MNT-KRNIC-AP
changed:     hostmaster@nic.or.kr 20010523
source:      APNIC

person:      Seungchul Hwang
address:     Hanaro Telecom Co.
address:     Kukje Electornics Cneter Bldg., 1445-3 Seocho-Dong Seocho-Ku
address:     SEOUL
address:     137-070
country:     KR
phone:       +82-2-106
fax-no:      +82-2-6266-6483
e-mail:      info@hananet.net
nic-hdl:     SH243-AP
mnt-by:      MNT-KRNIC-AP
changed:     hostmaster@nic.or.kr 20010523
source:      APNIC

inetnum:     211.215.16.0 - 211.215.19.255
netname:     HANANET-IDC-NGENE-KR
descr:       HANARO Telecom
descr:       1445-3 Seocho-Dong Seocho-Ku
descr:       SEOUL
descr:       137-728
country:     KR
admin-c:     IS5109-KR
tech-c:      SH6642-KR
remarks:     This IP address space has been allocated to KRNIC.
remarks:     For more information, using KRNIC Whois Database
remarks:     whois -h whois.nic.or.kr
remarks:     This information has been partially mirrored by APNIC from
remarks:     KRNIC. To obtain more specific information, please use the
```

```
remarks:    KRNIC whois server at whois.krnic.net.
mnt-by:     MNT-KRNIC-AP
changed:    hostmaster@nic.or.kr 20020204
source:     KRNIC

person:     Inyup Sung
country:    KR
phone:      +82-80-8282-106
fax-no:     +82-2-6266-6483
e-mail:     info@hananet.net
nic-hdl:    IS5109-KR
remarks:    This information has been partially mirrored by APNIC from
remarks:    KRNIC. To obtain more specific information, please use the
remarks:    KRNIC whois server at whois.krnic.net.
mnt-by:     MNT-KRNIC-AP
changed:    hostmaster@nic.or.kr 20020204
source:     KRNIC
```

Not surprising that this is from Korea, another popular origin of attacks.

A6.7.3. Analysis of the second top talker, 62.71.248.52:

The attacks from this address are very similar to the top talker address.
Of the 103 alerts, there are 100 unique destination addresses, almost certainly meaning a scan for an lpd exploit.
That said, I jumped right ahead to looking up the registration information:

```
Trying whois -h whois.ripe.net 62.71.248.52
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:     62.71.248.0 - 62.71.255.255
netname:     SONERA-PLAZA-DSL
descr:       DSL access pool
descr:       Sonera Plaza Ltd
country:     FI
remarks:     Please send abuse and spam notifications to abuse@inet.fi
admin-c:     HP7172-RIPE
tech-c:      SIH3-RIPE
status:      ASSIGNED PA
notify:      ripe-manager@datanet.tele.fi
mnt-by:      DATANET-NOC
changed:     kristian.rastas@datanet.tele.fi 20010419
changed:     Pia.Loponen@datanet.tele.fi 20010420
changed:     kristian.rastas@datanet.tele.fi 20010920
```

changed: kristian.rastas@datanet.tele.fi 20011009
source: RIPE

route: 62.71.0.0/16
descr: DN-062071-BLOCK
origin: AS5515
remarks: Spam & abuse cases, please try to contact
remarks: 1. abuse/contact address mentioned in the inetnum object
remarks: or
remarks: 2. abuse@sonera.net
notify: peering@tfi.net
notify: ripe-manager@datanet.tele.fi
mnt-by: DATANET-NOC
changed: Kristian.Rastas@datanet.tele.fi 20010419
changed: Kristian.Rastas@datanet.tele.fi 20011114
source: RIPE

role: Sonera Inet Hostmaster
address: Sonera Juxto Oy
address: Development and Production
address: PL 650
address: 00051 SONERA
phone: +358 20401
fax-no: +358 2040 59133
e-mail: hostmaster@ns.inet.fi
trouble: Please send abuse and spam notifications to abuse@inet.fi
trouble: General information: http://www.sonera.com/
admin-c: JM11414-RIPE
tech-c: JR143-RIPE
nic-hdl: SIH3-RIPE
notify: ripe-manager@datanet.tele.fi
mnt-by: DATANET-NOC
changed: kristian.rastas@datanet.tele.fi 20001212
changed: kristian.rastas@datanet.tele.fi 20010920
changed: kristian.rastas@datanet.tele.fi 20011018
source: RIPE

person: Hannu Peltola
address: Sonera Plaza Ltd.
address: Kuortaneenkatu 1 A
address: P.O.Box 572
address: FIN-00051 Sonera
e-mail: hannu.peltola@sonera.com
phone: +358 20402 58025
nic-hdl: HP7172-RIPE
changed: kristian.rastas@datanet.tele.fi 20010515
source: RIPE

This is from a Finnish DSL provider, possibly the Finnish version of @Home – the high-speed provider of residential Internet connections, which makes easy targets for attackers, who then use these compromised computers to launch their attacks.

A6.8. Severity:

The severity formula is not helpful in this case because we need more information for three of the four factors that determine the severity. I do not know the target criticality, system countermeasures, or network countermeasures.
As such, I'll grade only the attack lethality factor.

Lethality = 2, these alerts are for scans of the TCP 515 port. The scans don't rate a lethality of 2, but the large number of lpd vulnerabilities and the lethality of each makes me more worried than a score of 1.

A6.9. Defensive recommendation:

I would predict from the existence of the custom rule that the security personnel are particularly worried about the many lpd exploits. If that is the case, they'll probably want to stop traffic destined for TCP 515 at the routers and firewalls, only letting traffic through to the few that actually need to respond to that port. Also, take a look at the CVE vulnerabilities and patch any servers that are still vulnerable.

A7. Appendix 7: EXPLOIT x86 NOOP (#29 of 149)

This alert was chosen above other more frequent alerts because it sounds like a connection was made, not just a scan. Most of the alerts are probes, but the attacks are most dangerous, so they deserve the most attention.

A7.1. Source of trace:

"Analyze this" data from 12/20/2001 through 12/26/2001

A7.2. Detect was generated by:

Snort IDS

A7.3. Probability the source address was spoofed:

Unlikely since this appears to be a file transfer alert. If the attacker wants the files, he needs to provide a correct source address.

A7.4. Description of the attack:

This exact rule does not exist in the snort current rules that were downloaded on Feb. 2, 2002.
It looks like this rule has been expanded to more operating systems and encoding types. I predict this rule what is now the following:

shellcode.rules:alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"SHELLCODE x86

NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth: 128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:4;)

Which alerts when there are many of the x86 NOOP characters. When there are many NOOP characters in a row, there's a good chance the sender is looking to run a buffer overflow type of attack.

The CVE has many vulnerabilities for x86 NOOP:

Name	Description
CVE-1999-0139	Buffer overflow in Solaris x86 mkcookie allows local users to obtain root access.
CVE-2000-0316	Buffer overflow in Solaris 7 lp allows local users to gain root privileges via a long -d option.
CVE-2000-0337	Buffer overflow in Xsun X server in Solaris 7 allows local users to gain root privileges via a long -dev parameter.
CAN-1999-1014	** CANDIDATE (under review) ** Buffer overflow in mail command in Solaris 2.7 and 2.7 allows local users to gain privileges via a long -m argument.
CAN-1999-1026	** CANDIDATE (under review) ** aspppd on Solaris 2.5 x86 allows local users to modify arbitrary files and gain root privileges via a symlink attack on the /tmp/.asppp.fifo file.
CAN-1999-1035	** CANDIDATE (under review) ** IIS 3.0 and 4.0 on x86 and Alpha allows remote attackers to cause a denial of service (hang) via a malformed GET request, aka the IIS "GET" vulnerability.
CAN-1999-1048	** CANDIDATE (under review) ** Buffer overflow in bash 2.0.0, 1.4.17, and other versions allows local attackers to gain privileges by creating an extremely large directory name, which is inserted into the password prompt via the \w option in the PS1 environmental variable when another user changes into that directory.
CAN-2000-0317	** CANDIDATE (under review) ** Buffer overflow in Solaris 7 lpset allows local users to gain root privileges via a long -r option.
CAN-2000-1183	** CANDIDATE (under review) ** Buffer overflow in socks5 server on Linux allows attackers to execute arbitrary commands via a long connection request.
CAN-2001-0423	** CANDIDATE (under review) ** Buffer overflow in ipcs in Solaris 7 x86 allows local users to execute arbitrary commands via a long TZ (timezone) environmental variable.
CAN-	** CANDIDATE (under review) ** Buffer overflow in mail included

2001-0686	with SunOS 5.8 for x86 allows a local user to elevate privileges via a long HOME environmental variable.
CAN-2001-0694	** CANDIDATE (under review) ** Directory traversal vulnerability in WFTPD 3.00 R5 allows a remote attacker to view arbitrary files via a dot dot attack in the CD command.

A7.5. Attack mechanism:

In a Buffer Overflow attack, a client sends more information than a server has allocated, in attempt to overwrite the instruction pointer for the next command to execute, causing the computer to execute something like /bin/sh, giving the client a shell rather than the expected command (Decker). This is an overly simplified summary of what I learned from Nicole LaRock Decker's white paper titled: "Buffer Overflows: Why, How and Prevention."

In the active targeting section we'll do some research into the attacks to try to determine the service to be exploited.

A7.6. Correlations:

On April 4, 2001, Cliff Yago reported x86 NOOPs had been used by a Linux-based worm to compromise one of his computers (through an lpd buffer overflow) and was attempting to spread further (Yago).
John Jenkinson also found the "EXPLOIT x86 NOOP" alert in his analysis of the alerts for Sept. 29, 2001 – Oct. 3, 2001 (http://www.giac.org/practical/John_Jenkinson_GCIA.doc).
David Oborn also found the "EXPLOIT x86 NOOP" alert in his detects section on March 13, 2001 (http://www.giac.org/practical/David_Oborn_GCIA.html - detect4).

A7.7. Evidence of active targeting:

To determine active targeting, we need more information on these alerts.
There were 789 alerts for the seven day period of Dec. 20, 2001 through Dec. 26, 2001.

A7.7.1. Address analysis:

First we need to find out who caused the alerts (Top 5 list):

```
+------------------+-----------------+-------+
| alert            | srcip           | count |
+------------------+-----------------+-------+
| EXPLOIT x86 NOOP | 193.197.70.61   |   437 |
| EXPLOIT x86 NOOP | 24.3.48.41      |   281 |
| EXPLOIT x86 NOOP | 205.138.230.234 |    44 |
| EXPLOIT x86 NOOP | 199.178.222.76  |     9 |
| EXPLOIT x86 NOOP | 209.150.97.11   |     4 |
+------------------+-----------------+-------+
```

It looks like we have two big offenders, another worth looking into and a bunch that may be wrong numbers.

When I bring in the destination addresses, it becomes more interesting because only one of the sources raised this alert to more than one destination address. The source 205.138.230.234, which is the third top talker, raised the alert to three different destinations.

srcip	dstip	count(*)
129.250.247.187	MY.NET.98.175	1
131.118.254.130	MY.NET.1.6	2
132.66.50.47	MY.NET.83.95	1
193.197.70.61	MY.NET.163.15	437
194.18.30.208	MY.NET.98.129	2
199.178.222.76	MY.NET.98.198	9
205.138.230.234	MY.NET.177.18	2
205.138.230.234	MY.NET.97.216	38
205.138.230.234	MY.NET.97.94	1
205.138.230.234	MY.NET.98.148	3
207.46.177.148	MY.NET.233.106	1
207.68.131.21	MY.NET.98.185	1
208.31.254.10	MY.NET.97.226	2
209.150.97.11	MY.NET.1.6	4
216.32.211.86	MY.NET.217.118	1
24.3.48.41	MY.NET.162.36	281
63.236.2.29	MY.NET.98.149	2
65.168.29.84	MY.NET.97.211	1

As I think about this alert more, I find myself thinking my approach is backwards for this alert. Where I wanted to immediately discount all the sources with only a few occurrences, I should probably be more doubtful of the sources with the highest frequency. We're talking about an exploit here, not a flood. If it's going to take 437 tries, that's probably not a very good attack. After the above query, I would guess that 205.138.230.234 is an attacker and the two top talkers are false positives.

A query including the ports got too large to easily see meaning, so at this point I'm going to analyze each top talker separately. (I will only be examining the top 3 talkers since they make up almost all of the alerts, 762 of 789, or 96.58%.)

A.7.7.2. Top source #1: 193.197.70.61

We want to take a look at the ports involved in the alerts from this source:

srcip	srcport	dstip	dstport	count(*)
193.197.70.61	1626	MY.NET.163.15	22	55
193.197.70.61	1627	MY.NET.163.15	22	53

```
| 193.197.70.61 | 1628    | MY.NET.163.15 | 22      |        59 |
| 193.197.70.61 | 1629    | MY.NET.163.15 | 22      |        52 |
| 193.197.70.61 | 1630    | MY.NET.163.15 | 22      |        49 |
| 193.197.70.61 | 1631    | MY.NET.163.15 | 22      |        53 |
| 193.197.70.61 | 1632    | MY.NET.163.15 | 22      |        57 |
| 193.197.70.61 | 1633    | MY.NET.163.15 | 22      |        59 |
+---------------+---------+---------------+---------+-----------+
```

That's a pretty interesting pattern…
We see that all of the alerts from 193.197.70.61 are to MY.NET.163.15.
We see that all of the alerts are to the ssh port.
We see that there are about 50 alerts each time.
We also see that the source port incremented by one on each successive group.
That looks to me like a scripted probe or exploit. I wrote up the CRC32 integer buffer overflow for ssh in the Net Detects section, is this it? I went back to Dittrich's paper, after explaining what you'd see on each end, Dittrich has a section for snort signatures, one of which included searching for x86 NOOPs:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22     \
   (msg:"EXPLOIT ssh CRC32 overflow NOOP";     \
   flags:A+; content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; \
   reference:bugtraq,2347; reference:cve,CVE-2001-0144; \
   classtype:shellcode-detect;) (Dittrich)
```

That is probably almost exactly what the "EXPLOIT x86 NOOP" alert is looking for.

Now I'm interested in why there are eight separate sessions. To try to find out more, I'll bring times into the picture.

```
+---------------+---------+---------------+---------+---------------------+---------------------
+----------+
| srcip         | srcport | dstip         | dstport | min(time)           | max(time)           |
count(*) |
+---------------+---------+---------------+---------+---------------------+---------------------
+----------+
| 193.197.70.61 | 1626    | MY.NET.163.15 | 22      | 2001-12-20 00:57:16 | 2001-12-20 00:57:17
|       55 |
| 193.197.70.61 | 1627    | MY.NET.163.15 | 22      | 2001-12-20 00:57:18 | 2001-12-20 00:57:19
|       53 |
| 193.197.70.61 | 1628    | MY.NET.163.15 | 22      | 2001-12-20 00:57:20 | 2001-12-20 00:57:20
|       59 |
| 193.197.70.61 | 1629    | MY.NET.163.15 | 22      | 2001-12-20 00:57:21 | 2001-12-20 00:57:22
|       52 |
| 193.197.70.61 | 1630    | MY.NET.163.15 | 22      | 2001-12-20 00:57:23 | 2001-12-20 00:57:24
|       49 |
| 193.197.70.61 | 1631    | MY.NET.163.15 | 22      | 2001-12-20 00:57:25 | 2001-12-20 00:57:26
|       53 |
| 193.197.70.61 | 1632    | MY.NET.163.15 | 22      | 2001-12-20 00:57:27 | 2001-12-20 00:57:27
|       57 |
| 193.197.70.61 | 1633    | MY.NET.163.15 | 22      | 2001-12-20 00:57:29 | 2001-12-20 00:57:29
|       59 |
+---------------+---------+---------------+---------+---------------------+---------------------
+----------+
```

The attacks occurred very close to each other, time-wise. Each cluster of attacks took no more than one second, with the next cluster coming in within a second or two. The total time for all of

the attacks was 13 seconds. This attack was definitely scripted, but I cannot tell whether these are eight probes for specific overflow exploits, or all a part of a single exploit. To get this information, we would have to look at the contents of the alert packets.

To take a guess, I'll took a look to see if we have any other alerts or scans from that source. There were no other alerts or scans with the source address or destination address of 193.197.70.61. Without further information, I cannot tell if the x86 NOOP attacks worked or not.

Lastly, we'll want to find out to whom this source belongs:

```
Trying whois -h whois.ripe.net 193.197.70.61
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      193.197.68.0 - 193.197.70.255
netname:      BN-ULM
descr:        Traegerverein Buergernetz Ulm/Neu-Ulm e.V.
country:      DE
admin-c:      PAUL3-RIPE
tech-c:       AFB13-RIPE
tech-c:       BEL553-RIPE
status:       ASSIGNED PA
remarks:      please report spam and other abuse to abuse@bn-ulm.de
mnt-by:       BELWUE-NTFY
changed:      hoechel@belwue.de 19991230
changed:      hoechel@belwue.de 20000830
source:       RIPE

route:        193.197.0.0/16
descr:        BELWUE-AGG-197
origin:       AS553
mnt-by:       BELWUE-MNT
changed:      georgi@belwue.de 19991014
source:       RIPE

role:         BelWue Network Operation Center
address:      Universitaet Stuttgart
address:      Rechenzentrum
address:      BelWue-Koordination
address:      Allmandring 30
address:      D-70550 Stuttgart
address:      Germany
phone:        +49 711 685 5804
phone:        +49 711 685 5807
```

fax-no: +49 711 678 8363
e-mail: noc@belwue.de
admin-c: PM178
tech-c: PM178
tech-c: WH657-RIPE
tech-c: JG5-RIPE
tech-c: JF1236-RIPE
tech-c: IH353-RIPE
nic-hdl: BEL553-RIPE
remarks: http://www.belwue.de
notify: hm-dbm-msgs@ripe.net
mnt-by: BELWUE-MNT
changed: hoechel@belwue.de 19980910
changed: hoechel@belwue.de 19981119
changed: hoechel@belwue.de 19990121
source: RIPE

person: Markus Klenk
address: Internet Ulm/Neu-Ulm e.V.
address: M.-Dietrich-Str. 5
address: D-89231 Neu-Ulm
phone: +49 731 9217439
e-mail: klenk@bn-ulm.de
nic-hdl: PAUL3-RIPE
notify: hostmaster@bn-ulm.de
mnt-by: BN-ULM
changed: hostmaster@bn-ulm.de 20000727
changed: hostmaster@bn-ulm.de 20010904
source: RIPE

person: Andreas Franz Borchert
address: Am Nohl 33
address: 89173 Lonsee, Germany
phone: +49 7336 5896
e-mail: hostmaster@bn-ulm.de
nic-hdl: AFB13-RIPE
notify: hostmaster@bn-ulm.de
mnt-by: BN-ULM
changed: hostmaster@bn-ulm.de 20000104
changed: hostmaster@bn-ulm.de 20000106
source: RIPE

Belwue.de is a German ISP that provides high speed Internet access. Belwue stands for "Baden-Württembergs extended LAN." This is very much like our @Home ISP, so it's not surprising that the attack came from there.

A.7.7.3. Top source #2: 24.3.48.41:

We want to take a look at the ports involved in the alerts from this source:

```
+------------+---------+---------------+---------+----------+
| srcip      | srcport | dstip         | dstport | count(*) |
+------------+---------+---------------+---------+----------+
| 24.3.48.41 | 64798   | MY.NET.162.36 | 13696   |      145 |
| 24.3.48.41 | 64998   | MY.NET.162.36 | 42652   |      136 |
+------------+---------+---------------+---------+----------+
```

Those ports don't immediately draw my attention like the top source #1. In fact I couldn't find any information about any of the ports, so I cannot tell which is the server and more importantly, I cannot tell why they connected in the first place. If I had to guess, I'd say 24.3.48.41 is the client, since the source ports are relatively close together.

Bringing the times into the query to see if that helps:

```
+------------+---------+---------------+---------+---------------------+---------------------+---
-------+
| srcip      | srcport | dstip         | dstport | min(time)           | max(time)           |
count(*) |
+------------+---------+---------------+---------+---------------------+---------------------+---
-------+
| 24.3.48.41 | 64798   | MY.NET.162.36 | 13696   | 2001-12-20 18:03:30 | 2001-12-20 18:05:09
|      145 |
| 24.3.48.41 | 64998   | MY.NET.162.36 | 42652   | 2001-12-20 18:46:10 | 2001-12-20 18:47:44
|      136 |
+------------+---------+---------------+---------+---------------------+---------------------+---
-------+
```

Each group is pretty tightly clustered at a little more than a minute and a half for each. This shows us that the clusters certainly are separate attacks. However, this still doesn't tell me what is going on.

The address 24.3.48.41 does not occur as a source or destination for any other alerts or scans. A whois lookup shows that this address is from the @Home network:

> Trying whois -h whois.arin.net 24.3.48.41
> @Home Network (NETBLK-ATHOME) ATHOME 24.0.0.0 - 24.23.255.255
> @Home Network (NETBLK-MD-COMCAST-ESSX-1) MD-COMCAST-ESSX-1
> 24.3.48.0 - 24.3.55.255

To single out one record, look it up with "!xxx", where xxx is the
handle, shown in parenthesis following the name, which comes first.

The ARIN Registration Services Host contains ONLY Internet
Network Information: Networks, ASN's, and related POC's.
Please use the whois server at rs.internic.net for DOMAIN related
Information and whois.nic.mil for NIPRNET Information.

All of my findings are inconclusive on what was going on between 24.3.48.41 and MY.NET.162.36.
There were two attacks, each attack raised the "EXPLOIT x86 NOOP" alert around 140 times

under 3 minutes.
The second attack took place some 41 minutes after the first attack ended.
The source is from a cable modem ISP, known to be the source of many attacks.
The source and destination ports are unregistered, ephemeral ports.
Neither the source nor destination addresses show any other significant alerts or scans (there were 5 total scans of MY.NET.162.36).
Given the above, I cannot say that this is definitely an attack, but it looks awfully suspicious. I would certainly like to see the contents of those alerts and I would also like to take a look at MY.NET.162.36 to find out what is running on those ports, 13696 and 42652. Since I couldn't find anything about those ports, I have to suggest that they represent a backdoor for which the port can be set by the attacker. I would definitely consider MY.NET.162.36 as a compromised computer until it can be proven otherwise.

A.7.7.4. Top source #3: 205.138.230.234:

We want to take a look at the ports involved in the alerts from this source:

srcip	srcport	dstip	dstport	count(*)
205.138.230.234	80	MY.NET.177.18	2430	1
205.138.230.234	80	MY.NET.177.18	3165	1
205.138.230.234	80	MY.NET.97.216	1148	1
205.138.230.234	80	MY.NET.97.216	1150	1
205.138.230.234	80	MY.NET.97.216	1176	6
205.138.230.234	80	MY.NET.97.216	1177	9
205.138.230.234	80	MY.NET.97.216	1178	12
205.138.230.234	80	MY.NET.97.216	1179	8
205.138.230.234	80	MY.NET.97.216	1190	1
205.138.230.234	80	MY.NET.97.94	1088	1
205.138.230.234	80	MY.NET.98.148	1156	2
205.138.230.234	80	MY.NET.98.148	1158	1

-+

This looks like a bunch of false positives, where 205.138.230.234 is not an attacker, but a web server that happens to be returning content that includes strings of NOOP characters.

Just to make sure this isn't a clever attacker who is crafting his packets to use a source port of 80, which he probably knows many of us will dismiss as non-harmful web-traffic false positive, I did some research into the 205.138.230.234 address.

The whois lookup shows that this is from Cable & Wireless (ISP), and American Express:

> Trying whois -h whois.arin.net 205.138.230.234
> Cable & Wireless USA (NETBLK-CW-03BLK) CW-03BLK 205.138.0.0 - 205.140.255.255
> AMERICAN EXPRESS (NETBLK-CW-205-138-230) CW-205-138-230
> 205.138.230.0 - 205.138.230.255

To single out one record, look it up with "!xxx", where xxx is the handle, shown in parenthesis following the name, which comes first.

The ARIN Registration Services Host contains ONLY Internet Network Information: Networks, ASN's, and related POC's.
Please use the whois server at rs.internic.net for DOMAIN related Information and whois.nic.mil for NIPRNET Information.

To make sure the 205.138.230.234 was a web server, I plugged the address into my browser. It didn't bring up a web page to that IP address, but I was automatically forwarded to: http://www.americanexpress.com/homepage/mt_personal.shtml

So it looks like someone was only doing some personal finance when this alert tripped. This is good enough for me to call this a false positive.

A7.8. Severity:

The severity formula is not helpful in this case because we need more information for three of the four factors that determine the severity. I do not know the target criticality, system countermeasures, or network countermeasures.
As such, I'll grade only the attack lethality factor.

Lethality = 5, these alerts are for buffer overflows that can give root privileges to the attacker. Of the three sources I investigated, one was a false positive, another was almost certainly trying to run ssh overflow exploits, and the third is completely unknown. We have no reason to think that either of the attacks failed, so must assume that they succeeded.

A7.9. Defensive recommendation:

I would strongly urge that the attacked computers be taken off the network until they can be investigated.

We'll want to update the ssh daemons for any computers that need to be running ssh, disabling sshd for any computers that do not need to be running sshd. Router ACLs and firewall rules should allow ssh traffic only to the computers that should allow ssh connections from external networks.

If the computer MY.NET.162.36 has been compromised, make sure the attack is published, so others will be able to spot similar attacks. We particularly want to let others in the community know what backdoor was running on those previously unknown ports.

A.8. Appendix 8: Scan analysis details

A8.1. Top 10 source addresses for scans: (total 1665 unique source addresses)

srcip	count
MY.NET.162.233	926049
MY.NET.163.15	685164
MY.NET.87.50	520160
MY.NET.98.203	27085
209.190.237.123	20606
205.188.233.121	15308
205.188.228.33	12818
62.211.247.3	12251
205.188.246.121	11413
24.205.153.114	9895

A8.2. Top 10 destinations of scans:

dstip	count
24.164.41.210	21952
MY.NET.70.134	20614
MY.NET.190.15	15144
216.33.98.254	11066
194.251.249.182	7144
24.157.184.117	7080
MY.NET.70.148	6598
24.23.140.185	5574
128.8.240.71	5500
MY.NET.106.178	5195

A8.3. Further research on the top source.

It appears that the top scanner (MY.NET.162.233) was compromised by a worm or virus that automatically scans ranges of IP addresses to search for vulnerable computers in order to spread itself. Of the 926,049 scans that had MY.NET.162.233 as the source IP address, there were 925,391 unique destination IP addresses scanned.

MY.NET.162.233 scanned approximately half of the following Class B networks (925,391 of 2,097,152 possible addresses):

124.1.0.0/16	124.2.0.0/16	125.1.0.0/16	130.86.0.0/16
130.87.0.0/16	132.1.0.0/16	132.2.0.0/16	132.3.0.0/16
132.4.0.0/16	132.74.0.0/16	132.75.0.0/16	132.76.0.0/16
132.77.0.0/16	133.1.0.0/16	133.2.0.0/16	133.3.0.0/16
133.4.0.0/16	133.5.0.0/16	134.1.0.0/16	134.2.0.0/16
148.1.0.0/16	148.2.0.0/16	148.3.0.0/16	148.4.0.0/16
148.5.0.0/16	148.87.0.0/16	208.120.0.0/16	208.121.0.0/16
208.122.0.0/16	208.124.0.0/16	208.125.0.0/16	MY.NET.0.0/16

Now that I know who it was scanning, I want to know what it was looking to find :

```
+----------------+---------+----------+----------+
| srcip          | dstport | protocol | count(*) |
+----------------+---------+----------+----------+
| MY.NET.162.233 | 113     | T        |       11 |
| MY.NET.162.233 | 22      | T        |   926038 |
+----------------+---------+----------+----------+
```

I also want to know when the scans occurred, which may help in correlating to attacks:

```
+----------------+---------------------+---------------------+
| srcip          | min(time)           | max(time)           |
+----------------+---------------------+---------------------+
| MY.NET.162.233 | 2001-12-20 02:32:24 | 2001-12-21 01:30:17 |
+----------------+---------------------+---------------------+
```

We find that nearly all of the scans from this address were to TCP port 22, the port which ssh runs on.
The destination port of TCP 22 tells us this computer is not infected by Code Red II or the Nimda worm, which are known to scan for other vulnerable computers upon being infected. This made me more curious – now I wanted to know whether there were alerts that followed these scans. Unfortunately, there were no alerts with the source address of MY.NET.162.233. However, that does not mean that there were no attacks that followed. Although there are five rules related to ssh alerts in the current set of snort rules, these rules may not have been implemented in these captures – I confirmed that of the 149 unique alerts that were raised, none are among the five ssh rules. Since the timing of the scans takes place fairly early in the captured time period, I would have expected to see the attacking alerts. Furthermore, an attack may have come from a different computer. An attacker may use separate computers for reconnaissance and attacking to try to confuse security countermeasures.

A8.4. Top sources per destination port

The purpose of this section is to help the security personnel to understand which computers are the biggest offenders for each type of scan, for the ten most frequent scans.

The top 10 destination ports:

dstport	protocol	count
22	T	1645747
27005	U	261949
6970	U	76742
1214	T	61333
21	T	50331
6112	U	42212
27500	U	21624
25	T	19289
0	U	15722
6346	T	11141

A8.4.1. TCP 22, ssh

Top 10 sources:

srcip	count
MY.NET.162.233	926038
MY.NET.163.15	685162
211.248.231.10	9876
208.3.248.2	7673
24.0.28.234	5072
200.57.36.68	4160
213.219.17.147	3191
192.87.154.59	1668
24.14.23.109	1151
131.95.97.8	982

A8.4.2. UDP 27005, Half Life client (gaming):

Top 5 list:

srcip	count
MY.NET.87.50	261854
MY.NET.153.184	31

```
| MY.NET.87.44     |      31 |
| 209.190.237.123  |      13 |
| MY.NET.97.159    |       6 |
+------------------+---------+
```

It's pretty safe to predict that the MY.NET.87.50 computer was used by someone to play Half Life on the Internet, which means we'll probably want to curb that activity.

A8.4.3. UDP 6970, streaming ads in Real Player:

Top 10 list:

```
+------------------+-------+
| srcip            | count |
+------------------+-------+
| 205.188.233.121  | 15215 |
| 205.188.228.33   | 12760 |
| 205.188.246.121  | 11384 |
| 205.188.228.17   |  7612 |
| 205.188.244.57   |  7309 |
| 205.188.244.121  |  6399 |
| 205.188.233.185  |  5337 |
| 205.188.233.153  |  4462 |
| 205.188.228.1    |  3559 |
| 205.188.228.65   |  2498 |
+------------------+-------+
```

We can see that these are all external addresses, which helps to confirm that these probably are streaming ads, which also tells us that the source addresses are not what we want in this case. We want to see the destination addresses so we can tell which computers are running audio/video:

Top 10 list:

```
+-----------------+-------+
| dstip           | count |
+-----------------+-------+
| MY.NET.106.178  |  5144 |
| MY.NET.178.146  |  4961 |
| MY.NET.83.72    |  4777 |
| MY.NET.91.138   |  4191 |
| MY.NET.151.17   |  4046 |
| MY.NET.178.115  |  4029 |
| MY.NET.181.76   |  3648 |
| MY.NET.151.98   |  3582 |
| MY.NET.85.59    |  3419 |
| MY.NET.181.163  |  3067 |
+-----------------+-------+
```

That's definitely more like what we wanted to see.

A8.4.4. TCP 1214, KaZaA peer-to-peer file sharing

There were 953 unique source addresses, so I cannot list them all here. Instead I captured the top 10 in number of scans:

srcip	count
MY.NET.98.175	4142
MY.NET.115.115	2677
MY.NET.97.170	2520
MY.NET.97.177	2191
MY.NET.97.213	2101
MY.NET.98.120	1906
MY.NET.75.145	1784
MY.NET.97.237	1665
MY.NET.98.115	1603
MY.NET.97.167	1347

A8.4.5. TCP 21, FTP

Top 10 list:

srcip	count
62.211.247.3	12251
24.205.153.114	9895
210.61.63.66	9802
210.58.102.86	7680
24.44.21.206	5412
65.165.14.43	4548
217.36.8.58	194
217.136.153.154	159
MY.NET.84.185	159
24.150.0.194	122

These are the type of scan numbers I expected overall. Where there are far more scans initiated from the outside than the inside.
I was curious about the top scanner here, 62.211.247.3 turns out to be from an Italian DSL provider's pool of addresses.

A8.4.6. UDP 6112, Battle.net game servers

Top 10 list:

```
+---------------+-------+
| srcip         | count |
+---------------+-------+
| MY.NET.97.219 |  3548 |
| MY.NET.97.220 |  3128 |
| MY.NET.98.244 |  1718 |
| MY.NET.98.223 |  1477 |
| MY.NET.98.198 |  1340 |
| MY.NET.97.30  |  1300 |
| MY.NET.97.196 |  1269 |
| MY.NET.98.230 |  1251 |
| MY.NET.98.240 |  1242 |
| MY.NET.88.167 |  1164 |
+---------------+-------+
```

A8.4.7. UDP 27500, Quake game server

```
+---------------+----------+
| srcip         | count(*) |
+---------------+----------+
| MY.NET.87.50  |    21618 |
| MY.NET.153.184|        2 |
| MY.NET.97.159 |        2 |
| MY.NET.97.242 |        2 |
+---------------+----------+
```

It appears that our Half Life player also plays Quake (MY.NET.87.50).

A8.4.8. TCP 25, SMTP

Top 10 list:

```
+---------------+-------+
| srcip         | count |
+---------------+-------+
| MY.NET.253.24 |  9014 |
| MY.NET.6.47   |  3055 |
| MY.NET.6.35   |  1253 |
| MY.NET.253.52 |   869 |
| MY.NET.162.252|   807 |
| MY.NET.6.7    |   760 |
| MY.NET.100.230|   713 |
| MY.NET.6.34   |   540 |
| 199.183.24.194|   533 |
| MY.NET.97.208 |   485 |
+---------------+-------+
```

A8.4.9. UDP 0, fingerprinting?

Top 10 list:

srcip	count
209.190.237.123	5558
216.106.172.146	1269
200.69.193.177	1248
216.106.173.146	864
216.106.172.147	646
61.132.222.12	529
216.106.173.147	523
200.42.92.235	469
216.106.172.69	442
66.77.13.108	393

A8.4.10. TCP 6346, Gnutella

Top 10 list:

srcip	count
MY.NET.156.115	2919
MY.NET.97.186	2161
MY.NET.75.125	2126
MY.NET.110.224	381
MY.NET.97.36	374
MY.NET.98.228	364
MY.NET.98.147	319
MY.NET.98.175	305
MY.NET.106.139	303
MY.NET.105.247	292

A9. Appendix 9: OOS analysis details:

A9.1. Listing of source addresses, top 10 listed (82 unique addresses):

srcip	count
24.0.28.234	7931
210.125.178.52	167
199.183.24.194	80
64.172.24.155	40
24.36.185.188	15

```
| 141.157.92.22   |    12 |
| 211.39.150.91   |    11 |
| 65.165.238.50   |     9 |
| 202.168.254.178 |     7 |
| 213.84.157.192  |     7 |
+-----------------+-------+
```

A9.2. Top 10 destination addresses (7954 total unique):

```
+----------------+-------+
| dstip          | count |
+----------------+-------+
| MY.NET.163.15  |   168 |
| MY.NET.253.43  |    89 |
| MY.NET.70.70   |    44 |
| MY.NET.253.114 |    17 |
| MY.NET.253.125 |    16 |
| MY.NET.70.49   |    16 |
| MY.NET.253.41  |    14 |
| MY.NET.1.6     |    12 |
| MY.NET.60.14   |    10 |
| MY.NET.100.165 |     9 |
+----------------+-------+
```

A9.3. Further research into the top source (24.0.28.234):

A query too long to print here shows that all of the OOS packets with source address of 24.0.28.234 are sent to unique destination addresses. There are 7,931 packets and 7,931 unique destinations. This appears to be a scan of the entire MY.NET class B network.
Every packet from that source was scanning for the ssh service:

```
+-------------+---------+----------+
| srcip       | dstport | count(*) |
+-------------+---------+----------+
| 24.0.28.234 | 22      |     7931 |
+-------------+---------+----------+
```

A look at the scans detailed analysis shows this source as the 30th highest scanner, having caused 5,072 scan alerts.
If we included these scans, this source would vault up to number seven on the top scanners list.
The following shows that the scans occurred in a rather short time period:

```
+-------------+---------+--------------------+----------------
----+
| srcip       | dstport | min(time)          | max
(time)          |
+-------------+---------+--------------------+----------------
----+
```

```
| 24.0.28.234 | 22      | 2001-12-25 21:50:46 | 2001-12-25
22:12:22 |
+-------------+---------+---------------------+-----------------
----+
```

It took less than 22 minutes to scan 7,931 hosts for the ssh service.
Now that we know what this attacker was doing, when he was doing it and to whom, we want to take a look at where this attacker is coming from:

> Trying whois -h whois.arin.net 24.0.28.234
> @Home Network (NETBLK-ATHOME) ATHOME 24.0.0.0 - 24.23.255.255
> @Home Network (NETBLK-HOME-CORP-1) HOME-CORP-1 24.0.16.0 - 24.0.31.255

Another attacker from @Home, not surprising.

A9.4. Further research on top destination address (MY.NET.163.15):

There were two different sources that sent OOS packets to MY.NET.163.15.

```
+----------------+---------------+----------+
| srcip          | dstip         | count(*) |
+----------------+---------------+----------+
| 210.125.178.52 | MY.NET.163.15 |      167 |
| 24.0.28.234    | MY.NET.163.15 |        1 |
+----------------+---------------+----------+
```

We already know that 24.0.28.234 was a scan for the ssh port. Since all except that one packet are from 210.125.178.52, we should figure out what was going on there.

To find out, I ran a query that brought in the source and destination ports. Of the 167 occurrences, they were to 128 unique destination ports.

This tells us this was a port scan as well, except instead of scanning the whole class B for a single port, this scan was to a single host, looking for a response from TCP ports 0 through 183. The 0 through 183 range is strange, I would have expected it to go to 1023. I don't know if there is any significance in that.

```
+----------------+---------------+---------------------+---------------------+----------+
| srcip          | dstip         | min(time)           | max(time)           | count(*) |
+----------------+---------------+---------------------+---------------------+----------+
| 210.125.178.52 | MY.NET.163.15 | 2001-12-22 02:48:31 | 2001-12-22 03:09:46 |      167 |
+----------------+---------------+---------------------+---------------------+----------+
```

The scans took place within a span of 21 minutes, not very fast.

Looking back at the 12/22/2001 log, these packets appear to have been labeled OOS because both reserved TCP flags are set on each packet:

=+

12/22-02:48:31.500692 210.125.178.52:41989 -> MY.NET.163.15:0
TCP TTL:39 TOS:0x0 ID:27815 DF
21S***** Seq: 0xECDFE61E Ack: 0x0 Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 56199865 0 EOL EOL EOL EOL

=+
12/22-02:48:34.509093 210.125.178.52:41989 -> MY.NET.163.15:0
TCP TTL:39 TOS:0x0 ID:27816 DF
21S***** Seq: 0xECDFE61E Ack: 0x0 Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 56200165 0 EOL EOL EOL EOL

=+
12/22-02:48:38.504416 210.125.178.52:41990 -> MY.NET.163.15:1
TCP TTL:39 TOS:0x0 ID:13755 DF
21S***** Seq: 0xED10B8CE Ack: 0x0 Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 56200565 0 EOL EOL EOL EOL

A9.5. Destination Port analysis:

Top 10 destination ports:

```
+---------+-------+
| dstport | count |
+---------+-------+
| 22      |  7932 |
| 25      |   116 |
|         |    42 |
| 80      |    42 |
| 1214    |    34 |
| 21536   |    19 |
| 563     |    12 |
| 0       |    10 |
| 113     |     7 |
| 6346    |     6 |
+---------+-------+
```

A9.5.1. Top sources per destination port :

A9.5.1.1. Top destination port #1: TCP 22:

We've already researched 7,931 of the 7,932, so I'm skipping this.

A9.5.1.2. Top destination port #2: TCP 25, SMTP:

```
+-----------------+----------------+---------+----------+
| srcip           | dstip          | dstport | count(*) |
+-----------------+----------------+---------+----------+
| 199.183.24.194  | MY.NET.253.43  | 25      |       80 |
```

```
| 204.228.228.145 | MY.NET.253.41  | 25       |          2 |
| 204.228.228.145 | MY.NET.253.42  | 25       |          2 |
| 207.228.236.26  | MY.NET.100.230 | 25       |          2 |
| 210.125.178.52  | MY.NET.163.15  | 25       |          2 |
| 211.39.150.91   | MY.NET.253.41  | 25       |         11 |
| 217.106.234.13  | MY.NET.253.41  | 25       |          1 |
| 65.105.159.22   | MY.NET.253.42  | 25       |          2 |
| 65.105.159.22   | MY.NET.6.35    | 25       |          2 |
| 65.105.159.22   | MY.NET.6.40    | 25       |          1 |
| 65.105.159.22   | MY.NET.6.47    | 25       |          1 |
| 65.165.238.50   | MY.NET.253.43  | 25       |          9 |
| 65.184.132.241  | MY.NET.100.230 | 25       |          1 |
+-----------------+----------------+----------+------------+
```

A9.5.1.2.1. Top source 199.183.24.194:

Whois shows this as an unlikely suspect:

> Trying whois -h whois.arin.net 199.183.24.194
> ICG NetAhead, Inc. (NET-ICG-BLK-BLK4-C) ICG-BLK-BLK4-C
> 199.183.16.0 - 199.183.143.255
> Red Hat Software (NET-REDHAT) REDHAT 199.183.24.0 - 199.183.24.255

The time is pretty wide spread:

srcip	dstip	dstport	min(time)	max(time)
199.183.24.194	MY.NET.253.43	25	2001-12-22 01:07:48	2001-12-25 23:41:14

```
12/22-01:07:48.738261 199.183.24.194:46184 -> MY.NET.253.43:25
TCP TTL:52 TOS:0x0 ID:53975  DF
21S***** Seq: 0x722C52F3   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 192512036 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/22-03:09:23.255156 199.183.24.194:33723 -> MY.NET.253.43:25
TCP TTL:52 TOS:0x0 ID:49564  DF
21S***** Seq: 0x3EA8655C   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 193242304 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/22-03:31:16.072293 199.183.24.194:48513 -> MY.NET.253.43:25
TCP TTL:52 TOS:0x0 ID:26502  DF
21S***** Seq: 0x9049C2F9   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 193373567 0 EOL EOL EOL EOL

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

Manually looked through the logs for 12/22/2001 and 12/25/2001, they all appear to be OOS because they have the reserved TCP flags set.

Since there's no pattern and it keeps coming, I wouldn't guess this was a scan. You don't usually send the same scan eighty times over four days to the same computer, same port.

A9.5.1.3. Top destination port #3: <blank> (no port number):

```
+---------------+---------+-------------+---------+--------------------+--------------------+-
---------+
| srcip         | srcport | dstip       | dstport | min(time)          | max(time)          |
count(*) |
+---------------+---------+-------------+---------+--------------------+--------------------+-
---------+
| 64.172.24.155 |         | MY.NET.70.70 |         | 2001-12-22 12:55:56 | 2001-12-22 13:04:47
|       40 |
| 65.164.16.201 |         | MY.NET.70.70 |         | 2001-12-22 22:07:22 | 2001-12-22 22:07:22
|        2 |
+---------------+---------+-------------+---------+--------------------+--------------------+-
---------+
```

Manually looked through the logs for these packets, they all appear to be fragments with both the DF (do not fragment) and MF (more fragments) flags set:

```
12/22-12:55:56.078126 64.172.24.155 -> MY.NET.70.70
TCP TTL:108 TOS:0x0 ID:22235  DF MF
Frag Offset: 0x0   Frag Size: 0x22
70 DB C9 3A CB 40 7A 48 D2 FC 2F 75 42 FD 57 A3
p..:.@zH../uB.W.
6C 6E 05 47 4F A0 E1 B4 48 78 5C F5 FF 8B F9 76
ln.GO...Hx\....v
20 05                                            .

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
12/22-12:55:56.309531 64.172.24.155 -> MY.NET.70.70
TCP TTL:108 TOS:0x0 ID:22237  DF MF
Frag Offset: 0x0   Frag Size: 0x22
74 23 C3 CA 86 A1 45 C4 17 F7 04 05 71 0F 23 49
t#....E.....q.#I
83 B6 6D 9D 1C 14 66 F0 21 40 C7 FD DF 49 EE 0A  ..m...f.!
@...I..
01 B8                                           ..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
12/22-12:56:05.276093 64.172.24.155 -> MY.NET.70.70
TCP TTL:108 TOS:0x0 ID:22326  DF MF
Frag Offset: 0x0   Frag Size: 0x22
D1 65 6F BE A6 8D 1C D1 8C 3C 83 7B EF 2F C1 46  .eo......<.
{./.F
AF 30 98 A2 2A 5A 1F EF 3E FF 6A 1F AB C4 E1
32  .0..*Z..>.j....2
D1 D5                                           ..
```

Other than the conflicting fragmentation flags, the other thing that looks suspicious is the Frag offset are all set to 0x0, so each packet is claiming to be the first fragment. These packets certainly do look like they are crafted.

The source address looks like it belongs to an ISP's dialup pool:

> Trying whois -h whois.arin.net 64.172.24.155
> Pac Bell Internet Services (NETBLK-PBI-NET-8) PBI-NET-8
> 64.160.0.0 - 64.175.255.255
> PPPOX POOL - RBACK3 SNTC01 (NETBLK-SBCIS-101227-191152) SBCIS-101227-191152
> 64.172.24.0 - 64.172.25.255

The target, MY.NET.70.70 is a computer that we've previously determined to have been compromised and is running a KaZaA file sharing server. The packets may be evidence of the attack that compromised the MY.NET.70.70 computer. These packets are from 12/22/2001, the alerts we researched earlier (in Appendix 4: Watchlist 000220 IL-ISDNNET-990517) occurred between 12/25/2001 and 12/26/2001.

A9.5.1.4. Top destination port #4: TCP 80, http:

srcip	dstip	dstport	count(*)
12.230.253.9	MY.NET.253.125	80	4
12.234.167.15	MY.NET.6.7	80	1
193.11.234.215	MY.NET.100.165	80	1
193.120.224.170	MY.NET.253.125	80	5
193.231.20.2	MY.NET.60.14	80	3
193.251.49.8	MY.NET.6.7	80	2
195.96.106.109	MY.NET.100.165	80	1
202.130.239.149	MY.NET.100.165	80	4
202.168.254.178	MY.NET.253.125	80	7
202.75.185.186	MY.NET.100.165	80	1
212.187.171.150	MY.NET.253.114	80	2
213.47.247.120	MY.NET.253.125	80	2
24.28.134.6	MY.NET.60.14	80	3
61.152.210.129	MY.NET.253.125	80	1
62.4.18.79	MY.NET.100.165	80	1
65.113.91.99	MY.NET.60.14	80	2
66.92.13.71	MY.NET.60.14	80	1
67.160.235.105	MY.NET.253.125	80	1

These alerts seem to be fairly well spread out, so I'll dismiss them as wrong numbers.

A10. Appendix 10: Description of the Analysis Process:

Stephen Northcutt made a statement that Christmas Day is a great day to attack computers. No one is actively monitoring them, yet they're up and running. Therefore, I chose to do my analysis on the days surrounding Christmas Day. However, I didn't want the judges to think I chose those dates because they happen to have much less alerts, scans and OOS data than the typical week. To prove that wasn't the case, I added on two more days, causing my analysis to include seven consecutive days instead of the required five.

An important factor in my methodology was that I'm working with extremely limited resources. I could see that previous practicals were most often done by running SnortSnarf on the collected data, but the machine I'm working on is a Pentium 150 with 64 MB RAM, running Solaris 8. The limitation of processing power made it impractical to use perl scripts on such a large amount of data. I tried running SnortSnarf on just the data for 12/25/2001, the script ran for over an hour, so I decided I needed to do something else. I understand that to keep in the spirit of the exercise, I should be building upon what my predecessors have done, but I decided that it was more important to keep in mind that this exercise should be as close to real-life as possible, meaning: If I was the real analyst for this University and those were my hardware limitations, I would have to do something more practical that run SnortSnarf, then wait for a few hours for output.

Given the above, I used sed and awk to format the collected data into files that could be inserted into a MySQL database. This proved to be much more practical, as I could now load the entire seven days of alerts into the database in under a minute and run queries that returned information in seconds rather than hours.

My other option was to put the records directly into MS Excel, but the limitations of Excel are that only 64K (65535) records can be entered onto a single spreadsheet – which was not sufficient, since I had over 570,000 alert records.

Analyzing the Alerts:

The first step was to concatenate the seven days of collected data:
Example:
```
> cat alert.011220 alert.011221 alert.011222 alert.011223 alert.011224 alert.011225 alert.011226
> bigalert
```

First of all, the collected alert data is not very easy to put into MySQL, because the field delimiter is "[**]", which awk cannot use. Further difficulties came with the type of alert field, which is of variable length and usually contains spaces (meaning you can't use spaces as a field delimiter).

To work around the field delimiter problem, I first ran the collected data through a sed command to replace the " [**] " with a single pipe character "|". (Note that I also got rid of a single space on either side of the field delimiter.)
Example:
```
> sed 's/ \[\*\*\] /|/g' bigalert > fixed_bigalert
```

This left the collected data looking like this:
```
> head fixed_bigalert
**************************************************
```

```
            Snort Alert Report at Fri Dec 21 00:06:13 2001
**********************************************************
12/20-00:16:08.460970 |spp_portscan: PORTSCAN DETECTED from MY.NET.87.50
(THRESHOLD 4 connections exceeded in 0 seconds)|
12/20-00:00:07.191190 |CS WEBSERVER - external web traffic|216.45.66.241:1137 ->
MY.NET.100.165:80
12/20-00:00:07.209293 |MISC traceroute|128.195.186.5:45144 -> MY.NET.140.9:33479
12/20-00:00:09.570242 |MISC source port 53 to <1024|137.49.1.100:53 -> MY.NET.1.4:53
12/20-00:16:12.872120 |spp_portscan: PORTSCAN DETECTED from MY.NET.97.129
(THRESHOLD 4 connections exceeded in 2 seconds)|
12/20-00:16:13.263525 |spp_portscan: portscan status from MY.NET.87.50: 10 connections
across 10 hosts: TCP(0), UDP(10)|
12/20-00:16:18.873634 |spp_portscan: portscan status from MY.NET.97.129: 7 connections
across 7 hosts: TCP(7), UDP(0)|
```

I then wrote an awk script to put the data into a format that would facilitate inserting the data into a MySQL table.
To get the data into that format, we need to:

- Get rid of the title lines
- Get rid of the spp_portscan lines (they are already represented in the scans data, so are redundant here)
- Parse the data such that there are tabs between the fields and "\N" occupies any field that is empty (like port numbers for ICMP alerts).

Here is the awk script:
(Note: the awk script is still what I'd call "quick and dirty", meaning it is very specific to the data I used. If I were the actual analyst for this University, I'd make the script not dependent on the given variables of monthno and year, figure out some better way to decide whether a record is a scan or alert, and make it more consistent [specifically the use of continue for title lines, but the else after a scan record]. Also, I'd look for a way to do the sed command and the awk script in the same command – either by combining the two or writing a wrapper script to run the sed, then the awk.)

```
# alert.awk -- takes the "analyze this" alert files, puts them into the
#            tab delimited format for entering into MySQL
#
BEGIN {
      titles = 0
      year = "2001"
      monthno = "12"
      OFS = "\t"
      scans = 0
      alerts = 0
      skip = "spp_portscan"
      print > FILENAME".txt"
      print > FILENAME".scans.txt"
      print > FILENAME".titles.txt"
      FS = "|"
```

```
# the print above is to create the output file.
}

{

# get rid of any lines that don't start with the date (title lines)
if ( substr($1,1,2) != monthno ) {
	print $0 >> FILENAME".titles.txt"
	++titles
	continue
}

if ( substr($2,1,12) == skip ) {
	print $0 >> FILENAME".scans.txt"
	++scans
}
else {
	# split up the date/time to get it into the format MySQL wants
	split($1,datetime,"-")  # split up date and time
	split(datetime[1],date,"/")  # split up the day and month of date
	split(datetime[2],time,".")  # split the seconds fraction out
	split(time[2],timefrac," ")  # remove the trailing spaces

	# split up the IP info
	split($3,tuples," ")  # split the Source IP from Dest IP
	split(tuples[1],source,":")  # split the Source IP from port
	split(tuples[3],dest,":") # split the Dest IP from port

	if ( source[2] == "" ) {
		source[2] = "\N"  # if no Source Port, set it to NULL
	}

	if ( dest[2] == "" ) {
		dest[2] = "\N"  # if no Dest Port, set it to NULL
	}

	print year"-"date[1]"-"date[2]" "time[1], timefrac[1], $2, source[1], source[2], dest[1], dest
[2] >> FILENAME".txt"
	++alerts

}

}

END {
	print "there were " titles " Title lines"
	print "there were " scans" scans"
	print "there were " alerts " alerts"
```

```
        print "there were " NR " total records"
}
```

In addition to the requirements in the bullets above, I also put in counters to keep track of how many of each type of record occurred, and put each type of record into a separate file. Lastly, I also outputted the total number of records so I could manually add up the types of each record to confirm that every record was used.

It turned out to be a very good thing I did this extra check, because I found that the alert data from 12/20/2001 had a rogue new line character. Here is the contents of the file for title lines:

```
**********************************************************
        Snort Alert Report at Fri Dec 21 00:06:13 2001
**********************************************************
:1087 -> MY.NET.253.114:80

**********************************************************
        Snort Alert Report at Sat Dec 22 00:05:30 2001
**********************************************************
**********************************************************
        Snort Alert Report at Sun Dec 23 00:05:04 2001
**********************************************************
**********************************************************
        Snort Alert Report at Mon Dec 24 00:05:08 2001
**********************************************************
**********************************************************
        Snort Alert Report at Tue Dec 25 00:05:04 2001
**********************************************************
**********************************************************
        Snort Alert Report at Wed Dec 26 00:05:06 2001
**********************************************************
**********************************************************
        Snort Alert Report at Thu Dec 27 00:05:21 2001
**********************************************************
```

As you can see, the ":1087 -> MY.NET.253.114:80" should actually be part of the record before it. I used vi to open the alert file, search for that line, and fix the problem.

After that, I re-ran everything from the concatenation through the awk script.
The result was:

```
> awk -f alert.awk fixed_bigalert
there were 22 Title lines
there were 195341 scans
there were 574157 alerts
there were 769520 total records
```

(Note: 22 Title lines is still unexpected, the problem is that there is another rogue new line character in the alert output for 12/20/2001, but that is a blank line with no data, so I didn't fix it)

The resulting alert file (fixed_bigalert.txt) looks something like this:
> head fixed_bigalert.txt

2001-12-20 00:00:07 191190 CS WEBSERVER - external web traffic 216.45.66.241 1137 MY.NET.100.165 80
2001-12-20 00:00:07 209293 MISC traceroute 128.195.186.5 45144 MY.NET.140.9 33479
2001-12-20 00:00:09 570242 MISC source port 53 to <1024 137.49.1.100 53 MY.NET.1.4 53
2001-12-20 00:00:14 970793 MISC source port 53 to <1024 128.8.10.60 53 MY.NET.1.5 53
2001-12-20 00:00:15 253606 MISC source port 53 to <1024 199.234.180.10 53 MY.NET.1.5 53
2001-12-20 00:00:22 674347 MISC source port 53 to <1024 216.136.95.2 53 MY.NET.1.5 53
2001-12-20 00:00:24 610014 CS WEBSERVER - external web traffic 66.77.74.235 3783 MY.NET.100.165 80
2001-12-20 00:00:25 089898 ICMP Source Quench MY.NET.5.13 \N MY.NET.200.150 \N
2001-12-20 00:00:25 089990 ICMP Source Quench MY.NET.5.13 \N MY.NET.200.148 \N

I then created a table in MySQL to house the data:

```
mysql> create table alerts (time DATETIME, timefrac INTEGER(6),
alert VARCHAR(50), srcip VARCHAR(15), srcport VARCHAR(5), dstip
VARCHAR(15), dstport VARCHAR(5));
Query OK, 0 rows affected (0.08 sec)

mysql> desc alerts;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| time     | datetime    | YES  |     | NULL    |       |
| timefrac | int(6)      | YES  |     | NULL    |       |
| alert    | varchar(50) | YES  |     | NULL    |       |
| srcip    | varchar(15) | YES  |     | NULL    |       |
| srcport  | varchar(5)  | YES  |     | NULL    |       |
| dstip    | varchar(15) | YES  |     | NULL    |       |
| dstport  | varchar(5)  | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
7 rows in set (0.17 sec)
```

(Note: MySQL doesn't support fractions of a second, so I put the fractions of a second into a separate field.)

Then I used the "load infile" capability to insert the data into the table:
mysql> load data local infile "/var2/mark/fixed_bigalert.txt" into table alerts;Query OK, 574158 rows affected (53.10 sec)

Once I'd gotten the data into the table, it was pretty simple to do select queries to do the analysis. Example: To find count how many of each type of alert and sort them (the first page on SnortSnarf output):

```
mysql> create temporary table tmp (alert VARCHAR(50), count INT
(8));
Query OK, 0 rows affected (0.02 sec)

mysql> insert into tmp select alert, count(*) from alerts group
by alert;
Query OK, 149 rows affected (28.90 sec)
Records: 149  Duplicates: 0  Warnings: 0
```

The analysis for the scans and OOS packets were based on the same awk script and similar mysql tables.

A11. Appendix 11: Works Cited:

Allerdice, Bryan. "RE: Packets destined for ports 6970 and 6972." Security Incidents Reporting mailing list archive. 18 July 2001. URL: http://lists.insecure.org/incidents/2001/Jul/0109.html (7 February 2002).

Anderson, Jason. "An Analysis of Fragmentation Attacks." SANS Institute Information Security Reading Room. 15 March 2001. URL: http://rr.sans.org/threats/frag_attacks.php (23 January 2002).

Arkin, Ofir. "Foundry Networks Networking Devices Padded Bytes with ICMP Port Unreachable(s) - The 12 Bytes from No Where." nmap-hackers (free security scanner) web archive. 6 December 2000. URL: http://lists.insecure.org/nmap-hackers/2000/Oct-Dec/0040.html (7 February 2002).

Audio-Video Transport Working Group, H. Schulzrinne, GMD Fokus, S. Casner, Precept Software, Inc., R. Frederick, Xerox Palo Alto Research Center, V. Jacobson, Lawrence Berkeley National Laboratory. "RTP: A Transport Protocol for Real-Time Applications." Request for Comments: 1889. January 1996. URL: http://www.rfc-editor.org/rfc/rfc1889.txt (7 February 2002).

Baker, Andrew R. "Chapter 6. Preprocessor Plugins." Snort Documentation. 2000. URL: http://www.dpo.uab.edu/~andrewb/snort/snortdoc/preplugin.html (22 January 2002).

"CVE-2000-0884." 22 January 2001. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0884 (30 January 2002).

"CVE-2001-0144." 7 May 2001. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0144 (4 February 2002).

"CVE-2001-0154." 7 May 2001. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0154 (30 January 2002).

"CVE-2001-0333." 18 September 2001. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0333 (30 January 2002).

Danyliw, R., C. Dougherty, A. Householder, and R. Ruefle. "CERT® Advisory CA-2001-26 Nimda Worm." 25 September 2001. URL: http://www.cert.org/advisories/CA-2001-26.html (28 January 2002).

Decker, Nicole LaRock. "Buffer Overflows: Why, How and Prevention." SANS Institute Information Security Reading Room, Threats and Vulnerabilities. 13 November 2000. URL: http://rr.sans.org/threats/buffer_overflow.php (11 February 2002).

Dittrich, David A. "Analysis of SSH crc32 compensation attack detector exploit." 15 November 2001. URL: http://staff.washington.edu/dittrich/misc/ssh-analysis.txt (4 February 2002).

Fulton, Russell. "slowish ssh scan from 149.69.85.65." Neohapsis Archives Incidents List. 4 December 2001. URL: http://archives.neohapsis.com/archives/incidents/2001-12/0047.html (4 February 2002).

Gregoire, Paul. "Winfreeze EXPLOIT Win9x/NT." Windows NT Bugtraq Mailing List. 8 March 1999. URL: http://www.ntbugtraq.com/default.asp?pid=36&sid=1&A2=ind9903&L=ntbugtraq&F=P&S=&P=2580 (6 February 2002).

Hagino, Jun-ichiro. "udp with dst port 0." tech-net@netbsd.org. 8 January 2000. URL: http://mail-index.netbsd.org/tech-net/2000/01/08/0000.html (7 February 2002).

Hernan, Shawn. "CERT® Advisory CA-2001-12 Superfluous Decoding Vulnerability in IIS." 15 May 2001. URL: http://www.cert.org/advisories/CA-2001-12.html (29 January 2002).

Hernan, Shawn. "Vulnerability Note VU#111677
Microsoft IIS 4.0 / 5.0 vulnerable to directory traversal via extended unicode in url (MS00-078)." 18 September 2001. URL: http://www.kb.cert.org/vuls/id/111677 (29 January 2002).

"ICMP Redirects Against Embedded Controllers." ISS Security Advisory. 10 December 1998. URL: http://www.infowar.com/iwftp/xforce/advise14.html (6 February 2002).

Larratt, Glenn. "Re: slowish ssh scan from 149.69.85.65." Neohapsis Archives Incidents List. 5 December 2001. URL: http://archives.neohapsis.com/archives/incidents/2001-12/0056.html (4 February 2002).

Lemos, Robert. "Hacker had WorldCom in his hands." 6 December 2001. URL: http://news.com.com/2100-1001-276711.html?legacy=cnet (31 January 2002).

Mackie, A., J. Roculan, R. Russell, and M. Van Velzen. "Nimda Worm Analysis." 21 September 2001. URL: http://aris.securityfocus.com/alerts/nimda/010919-Analysis-Nimda.pdf (29 January 2002).

McClure, S. and Scambray, J. "How hackers cover their tracks." 25 January 1999. URL:

http://www.cnn.com/TECH/computing/9901/25/hacktracts.idg/ (31 January 2002).

"Microsoft Security Bulletin (MS01-020) Incorrect MIME Header Can Cause IE to Execute E-mail Attachment." 29 March 2001. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-020.asp (29 January 2002).

"Net Metropolitan 1.00." Dark Eclipse Software Archive Trojans. 2000. URL: http://www.dark-e.com/archive/trojans/NetMetro/100/index.shtml (11 February 2002).

"NIMDA Worm/Virus Report – Final." 3 October 2001. URL: http://www.incidents.org/react/nimda.pdf (28 January 2002).

Northcutt, S. and J. Novak. Network Intrusion Detection An Analyst's Handbook, Second Edition. Indianapolis: New Riders Publishing. September 2000.

"OpenSSH Security." 20 January 2002. URL: http://www.openssh.org/security.html

Ostling, Andreas. "Re: slowish ssh scan from 149.69.85.65." Neohapsis Archives Incidents List. 5 December 2001. URL: http://archives.neohapsis.com/archives/incidents/2001-12/0060.html (4 February 2002).

Poulsen, Kevin. "Lamo's Adventures in WorldCom." 5 December 2001. URL: http://www.securityfocus.com/news/296 (31 January 2002).

Roesch, Marty. "Re: [snort] Tiny Fragments." Neohapsis Archives Snort Discussion. 14 May 2000. URL: http://archives.neohapsis.com/archives/snort/2000-05/0103.html (22 January 2002).

Rubio, Reuben. "GCIA Practical Assignment Version 2.9." June 2001. URL: http://www.giac.org/practical/REUBEN_RUBIO_GCIA.doc (31 January 2002).

Sage, John. "port 22 ssh probe from your host 209.61.148.63 - dev.plussize.com." 21 December 2001. URL: http://www.incidents.org/archives/intrusions/msg03009.html (4 February 2002).

Spriggs, Dwayne. "Version 2.9 The GCIA Practical." August 2001. http://www.giac.org/practical/Dwayne_Spriggs_GCIA.doc (31 January 2002).

Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, Inc. 1994.

Webb, Lloyd. "Practical Assignment 2.9 for GCIA Intrusion Analyst Certification." June 2001. URL: http://www.giac.org/practical/Lloyd_Webb_GCIA.doc (31 January 2002).

Yago, Cliff. "Report Date: April 04, 2001 – 0900." SANS Global Incident Analysis Center. 4 April 2001. URL: http://www.sans.org/y2k/040401.htm (11 February 2002).

Yuen, Rick. "GIAC Intrusion Detection, Practical Assignment." 11 October 2001. URL:

http://www.giac.org/practical/Rick_Yuen_GCIA.doc (28 January 2002).

Zenomorph. "Fingerprinting Port 80 Attacks: A look into web server, and web application attack signatures." November 2001. URL: http://www.cgisecurity.com/papers/fingerprint-port80.txt (30 January 2002).