Global Information Assurance Certification Paper

Todd Chapman

SANS GIAC Level Two – Intrusion Detection In Depth

SANS GCIA Practical Assignment

Version 3.0

SANS San Diego, October 2001

Table of Contents

Assignment 1 – Intrusion Detection, Active Response, and LaBrea

Intrusion Detection Systems (IDS) have been continuously evolving over the past 20 years. From the original host based systems, to network based and hybrid systems, the nature and capabilities of intrusion detection systems have continuously improved. In this paper we will briefly review the history of IDS's, discuss active response and its problems, and take a close look at LaBrea as an attack detection and active response tool.

The Evolution of Intrusion Detection

The concept of intrusion detection was born with James Anderson's 1980 paper, Computer Security Threat Modeling and Surveillance. Anderson recognized the value of audit trail data in tracking user activity and identifying misuse. The use of audit data means the first intrusion detection concepts were host centric, and in 1984 Dr. Dorothy Denning created the first functional host based intrusion detection system (HIDS), IDES. That same year Denning published, An Intrusion Detection Model, which served as a kind of blueprint for commercial IDS's to come.

In 1998 the Haystack project at Lawrence Livermore Labs developed a Distributed Intrusion Detection System (DIDS). This system monitored audit data on both servers and clients. Members of the Haystack project formed Haystack Labs, becoming the first commercial IDS vendor.

Then in 1990, Todd Heberlein developed Network Security Monitor, the first Network Intrusion Detection System (NIDS). Not only did Heberlein introduce the concept of NIDS, he also introduced the Hybrid Intrusion Detection System. Hybrid IDS's centralize intrusion detection management, integrating information from both network based and host based systems. Heberlein's work accelerated investment in the development of intrusion detection systems. The number of companies offering IDS's and sales of IDS's flourished in the mid to late '90s, becoming an essential piece of the network security puzzle today.

Regardless of their host or network based architecture, one thing all intrusion detection systems have had in common for most of their history is their passive nature. A passive IDS takes no defensive action based on attacks that it detects, other than notifying security administrators through logs, e-mail, pages, or other means. Even if the IDS monitors activity in real time, defensive actions are left to a security staff. With the frequent occurrence of false positives, how willing would a over worked security administrator be to respond to a pager in the middle of the night, analyze an alert, and remotely make configuration changes to defensive systems in response? Using a time based security model: $P > D + R$[1]. That is, how long our defensive measures remain

[1] Northcutt, Stephen. "Time Based Security." Presented at SANS conference. October 2001.

intact (P) should be greater than the sum of how long it takes to detect an attack (D) and how long it takes to respond to an attack (R). One approach to this problem is to reduce reaction time, R.

Active Response

This problem was the motivation behind the development of active response features in intrusion detection systems. The idea of an active IDS is to react to IDS alerts by automatically modifying the configuration of defensive systems. For example, if a network based IDS detects that an Internet host is port scanning the network, the IDS may respond by alerting the network's firewall rules to block all connections from the scanning host. There are several ways in which active response may be implemented.

As previously mentioned, an active IDS can respond to attacks by modifying firewall rules to block the attacker. Guardian (http://www2.chaotic.org:81/~astevens/Guardian/) is an add-on for the Snort IDS (http://www.snort.org/) that can respond to Snort alerts by updating firewall rules to block the attacker. Guardian currently supports Linux ipchains, Linux iptables, and FreeBSD IPFW firewalls.

Another method for actively stopping an attack is to kill the connection. This is sometimes referred to as, 'connection sniping', and has the added benefit of being independent of the type of firewall in use. For TCP, which is a connection oriented protocol, a connection can be killed by sending a TCP reset to the attacker, to the target, or both. For UDP which is connectionless, the same thing can be accomplished by sending an ICMP port unreachable messages to the sender.

The Snort IDS implements these features where they are referred to as, "flexible response." In addition to sending ICMP port unreachable to the attacker, Snort can also send ICMP host unreachable and ICMP net unreachable messages to the attacker. Snort's flexible response features are used by making modifications to Snort signature rules. These Snort features are currently in development and testing phase so use them at your own risk.

Of course, all active IDS features should be used with care. If a security administrator is not careful, an attacker can use the target's active response features against him. If the attacker spoofs her IP address, she can trigger alerts in the target's IDS which will cause the spoofed source address to be blocked. By spoofing the addresses of a large number of hosts the attacker could cause the IDS to DoS (denial of service) it's own network. Or the attacker could might spoof only a few critical addresses, such as those of DNS servers or other critical services. This is why an IDS with active response features must be carefully configured not to block the addresses of critical servers. Even if you could detect and ignore spoofed source addresses, occasionally legitimate network traffic is going to trigger a false positive in the IDS. With active response the result would be to deny service to legitimate users. These types of problems will probably get the active response features turned off.

Now imagine if spoofed source addresses were detectable and false positives were eliminated. In this idealized situation it sounds as if active response would be a silver bullet for stopping those attacks that are detectable. Unfortunately it's not that simple. Not only can the motivated attacker use spoofed source addresses to cause a DoS, but the attacker can also use techniques to defeat the actions of the active response mechanism.

Session sniping as an active response technique requires fast timing. To optimize transmission speed some TCP/IP stacks send more than one packet at a time and only re-send packets for which no acknowledgement is received. If the second packet in a series of three packets triggers an IDS alert, the active response mechanism must respond with a TCP reset to the destination before the third packet in the stream arrives. If it doesn't the destination may ignore the reset. If the attacker sends the packets out of order the third packet can be easily made to reach the target before the second packet triggers the alert. And if the attacker has gone to this much trouble you can bet that any TCP resets sent to the source address will be ignored.

If session sniping is so timing critical, can we use the firewall update method instead? Yes and no. Firewall updates are don't suffer the same timing problems but they have there own issues. Do you know if an update to your vendor's firewall rules will effect an already active connection? It's probably a safe bet that some won't. You'll need to test it to be sure. Also, if an attacker successfully execute a buffer overflow or similar attack, these is nothing to prevent the attacker taking advantage of a remote shell from a different IP address. Regardless of the active response methods used and their problems, perhaps the biggest problem is the fallacy of the name. 'active response.' The name *re*active response would be more appropriate. By the time active response techniques have kicked in a target is under attack. What the world needs is something that can prevent attacks before they occur.

LaBrea

In an July 31, 2001 posting to the Intrusions mailing list on Incidents.org, Tom Liston posted a message with the subject, "Can we make life difficult for Code Red?" In this posting Tom discussed his initial ideas for detecting and slowing Code Red and similar worms. Then on August 8, 2001 Tom announced LaBrea, "A 'Tarpit' That Traps Worms"[2].

A worm works by scanning networks for systems that a susceptible to a particular vulnerability. When a vulnerable system is found the worm installs itself on the target and begins scanning from the new location in addition to the original location. This

[2] Delio, Michael. "A 'Tarpit' That Traps Worms." September 19, 2001. http://www.wired.com/news/technology/0,1282,46964,00.html (Febuary 2002)

process allows worms to propagate very quickly. In the case of Code Red, up to 2,000 hosts per minute were infected[3].

The concept behind LaBrea is simple. Machines infected with worms scan thousands of IP addresses – in the case of Code Red, TCP port 80. Many of these addresses are not currently active. When a worm sends the SYN packet of the TCP 3-way handshake to a targeted IP address, it expects one of three responses:

1. If the IP address exists and offers a service on the destination report, the target will reply with a SYN/ACK. The worm then completes the handshake and tries the exploit on the target.
2. If the IP address exists and does not offer a service on the destination port, the target will respond with a TCP reset. The worm then moves on to the next address.
3. If the IP address does not exist then the connection attempt will eventually timeout. After several retries the worm moves on to the next target.

LaBrea monitors all unused IP addresses on a network, including every TCP port on those addresses. When a TCP connection attempt is made on one of those address, LaBrea accepts the connection is accepted and logged. At this point LaBrea's data transfer rate limiting features kick in.

LaBrea determines if an IP address is by listening to ARP requests and replies. If a router broadcasts and ARP request more than once without a reply, LaBrea will respond with an ARP reply containing its own MAC address. In a switched network LaBrea will not see ARP replies. LaBrea's '-s' switch can be used to get around this problem. In this case when LaBrea sees an ARP request it will issue a similar ARP request and if it gets no ARP reply it will issue its own reply to the original request.

TCP being a connection oriented protocol, tries hard to keep connections going, including modifying packet properties to accommodate restrictions advertised by either end of the connection. LaBrea advertises a TCP window size of 0 to the attacker after receiving the first data packet, limiting the number of bytes LaBrea is willing to accept to 0 bytes.

> "Many newcomers to TCP/IP are surprised to learn that no data whatsoever flows across an idle TCP connection. That is, if neither process at the ends of the TCP connection is sending data to the other, nothing is exchanged between the two TCP modules. There is no polling, for example, as you might find in other networking protocols. … This assumes that neither application – the client or

[3] CAIDA. "CAIDA Analysis of Code Red." November 28, 2001. http://www.caida.org/analysis/security/code-red/ (February 2002)

server – has application-level timers to detect inactivity, causing either application to terminate."[4] (Stevens, p. 331)

So depending on the implementation of the worm, the connection can be help open indefinitely.

Because the window size is set to zero, the attacker will periodically query the target to check for increases in window size. The cost in bytes per second of keeping a worm occupied varies based on the timer used to query for window size changes. This value varies for each operating system. The longer the timer the less expensive it is to hold onto a worm. According to Liston, the cost of holding a connection from an NT machine in a persistent state is 1,215 bytes per hour.

Other features of LaBrea include:

- LaBrea's virtual IP addresses can be pinged.
- LaBrea responds to SYN/ACK packets with a RST, limiting the effect of attacks which have spoofed the source address using an address from a network protected by LaBrea.
- LaBrea can run on modest hardware and can run completely in RAM on a diskless system by running from a Linux boot disk.

Although LaBrea was named the "MOST USEFUL APPLICATION of 2001" by eWEEK labs, other have questioned its usefulness. Rob Rosenberg of the vMyths virus information website (http://www.vmyths.com) speculated that LaBrea would not make a big impact because for LaBrea to be effective it needs to be run by people with large chunks of IP address space, and that not enough people cared about computer security to make an impact. Other have countered that LaBrea finally gives administrators an ethical way to fight back. [2]

Perhaps in response to this problem, LaBrea@Home (http://www.hackbusters.net/LaBrea/lbathome.html) has been developed. Unlike the original LaBrea, which runs on Linux and BSD machines, LaBrea@Home runs on Windows machines and was developed to run on single IP machines on cable and DSL connections that aren't already running web servers. The goal is the same though – tying up worms that attempt to connect to non-existent web servers.

Conclusion

Intrusion Detection Systems have undergone an interesting evolution in the past 20 years. From passive host based and then network based devices, to distributed and hybrid systems with active response capabilities, IDSs have become a powerful and

[4] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, Inc, 1994.

important component of a layered security strategy. The development of active response techniques has helped to reduce the exposure time for network based attacks, while at the same time introducing problems of their own. The development of LaBrea has added another weapon to the defense arsenal, slowing attackers to prevent them from reaching real targets, making it more difficult for hackers to implement effective attack tools, and making the term 'active response' more meaningful.

Additional References:

Innella, Paul. "The Evolution of Intrusion Detection Systems." November 16, 2001. http://www.securityfocus.com/infocus/1514

Kipp, James. "Using Snort as an IDS and Network Monitor In Linux." June 13, 2001. http://rr.sans.org/intrusion/monitor.php

Larsen, Jason and Haile, Jed. "Understanding IDS Active Response Mechanisms." January 29, 2002. http://www.securityfocus.com/infocus/1540

Liston, Tom. "Welcome to My Tarpit, The Tactical and Strategic Use of LaBrea." http://www.hackbusters.net/LaBrea/LaBrea.txt

Assignment 2 – Network Detects

Detect #1 – TCP Port 515 Probes

```
Dec 28 11:03:20 4.61.109.246:1869 -> a.b.c.20:515 SYN ******S*
Dec 28 11:03:20 4.61.109.246:1882 -> a.b.c.33:515 SYN ******S*
Dec 28 11:03:20 4.61.109.246:1900 -> a.b.c.51:515 SYN ******S*
Dec 28 11:03:20 4.61.109.246:1911 -> a.b.c.62:515 SYN ******S*
Dec 28 11:03:20 4.61.109.246:1915 -> a.b.c.66:515 SYN ******S*
Dec 28 11:03:20 4.61.109.246:2031 -> a.b.c.182:515 SYN ******S*
Dec 28 11:03:21 4.61.109.246:2061 -> a.b.c.212:515 SYN ******S*
Dec 28 11:03:21 4.61.109.246:2074 -> a.b.c.225:515 SYN ******S*
Dec 28 11:03:21 4.61.109.246:2349 -> a.b.d.245:515 SYN ******S*
Dec 28 11:03:21 4.61.109.246:2461 -> a.b.e.102:515 SYN ******S*
Dec 28 11:03:23 4.61.109.246:1931 -> a.b.c.82:515 SYN ******S*
Dec 28 11:03:24 4.61.109.246:2302 -> a.b.d.198:515 SYN ******S*
Dec 28 11:03:24 4.61.109.246:2307 -> a.b.d.203:515 SYN ******S*
Dec 28 11:03:24 4.61.109.246:2344 -> a.b.d.240:515 SYN ******S*
Dec 28 11:03:24 4.61.109.246:2347 -> a.b.d.243:515 SYN ******S*
Dec 28 11:03:24 4.61.109.246:2354 -> a.b.d.250:515 SYN ******S*
Dec 28 11:03:27 4.61.109.246:2537 -> a.b.e.176:515 SYN ******S*
Dec 28 11:03:27 4.61.109.246:2540 -> a.b.e.179:515 SYN ******S*
Dec 28 11:03:28 4.61.109.246:2545 -> a.b.e.184:515 SYN ******S*
Dec 28 11:03:30 4.61.109.246:2550 -> a.b.e.189:515 SYN ******S*
Dec 28 11:03:28 4.61.109.246:2578 -> a.b.e.217:515 SYN ******S*
Dec 28 11:03:28 4.61.109.246:2582 -> a.b.e.221:515 SYN ******S*
Dec 28 11:03:28 4.61.109.246:2590 -> a.b.e.229:515 SYN ******S*
Dec 28 11:03:28 4.61.109.246:2671 -> a.b.f.55:515 SYN ******S*
Dec 28 11:03:28 4.61.109.246:2690 -> a.b.f.74:515 SYN ******S*
Dec 28 11:03:28 4.61.109.246:2782 -> a.b.f.166:515 SYN ******S*
Dec 28 11:03:28 4.61.109.246:2784 -> a.b.f.168:515 SYN ******S*
Dec 28 11:03:31 4.61.109.246:2792 -> a.b.f.176:515 SYN ******S*
Dec 28 11:03:29 4.61.109.246:2820 -> a.b.f.204:515 SYN ******S*
Dec 28 11:03:29 4.61.109.246:2821 -> a.b.f.205:515 SYN ******S*
Dec 28 11:03:30 4.61.109.246:2630 -> a.b.f.14:515 SYN ******S*
Dec 28 11:03:30 4.61.109.246:2632 -> a.b.f.16:515 SYN ******S*
Dec 28 11:03:30 4.61.109.246:2636 -> a.b.f.20:515 SYN ******S*
Dec 28 11:03:31 4.61.109.246:2758 -> a.b.f.142:515 SYN ******S*
Dec 28 11:03:31 4.61.109.246:2761 -> a.b.f.145:515 SYN ******S*
```

1. Source of Trace

Laurie Zirkle, Incidents mailing list archive.
http://www.incidents.org/archives/intrusions/msg03090.html

2. Source of Detect

The detect is from an unknown source. There is no indication that the connections were denied or allowed so it is probably not a firewall log. The detect may be from an IDS with rule to record all connection attempts to TCP port 515. The log format is:

```
Date Stamp | Source IP:Port -> Destination IP:Port | State | TCP Flags Set
Dec 28 11:03:20 | 4.61.109.246:1869 -> a.b.c.20:515 | SYN | ******S*
```

3. Probability the source address was spoofed

This appears to be an attempt to find or exploit lpd servers. To find lpd servers the attacker needs to attempt a 3-way TCP handshake with the target. If there is an active lpd server then the handshake should complete. If there is no active lpd server then the attacker will expect a TCP reset.

To exploit a vulnerable lpd server the attacker must complete the handshake with the server so that it can then send the exploit code in the data. Either way spoofing the source address would defeat the attackers goal. Therefore the source address is probably not spoofed.

4. Description of Attack

The attacker is scanning the target network to either:

a) Build a list of all lpd servers, or
b) Attempt to exploit a previously built list of lpd servers

Because not every host on the target network is contacted the attacker may have a list of existing hosts from previous reconnaissance. lpd is the Unix printing service. Several vendors lpd implementations are susceptible to remote buffer overflow which allows the attacker to execute arbitrary code with the privileges of the server.

5. Attack Mechanism

The attack is either reconnaissance or an exploit attempt and is executed by scanning a list of hosts, targeting lpd servers listening on TCP port 515. Once an lpd server is found the attacker attempts to execute a buffer overflow attack against the server. The attack works by sending more data to the server than the server was programmed to accept. The extra data overwrites portions of memory reserved for program instructions. When formatted properly the data can include instructions to start a remote shell or perform other malicious activity. If the attack is successful the attacker will execute programs with the privileges of the lpd server, usually root.

Information about buffer overflow attack can be found at:

- http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2605669,00.html
- http://news.com.com/2100-1001-233483.html?legacy=cnet

The details of lpd print service vulnerabilities can be found in the following CERT and CVE postings:

- http://www.cert.org/advisories/CA-2001-15.html
- http://www.cert.org/advisories/CA-2001-30.html
- http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=LPD

6. Correlation

Many posts to the Intrusions mailing list have documented port 515 scans to Laurie's network. This trace is the first trace from source address 4.61.109.246 and standard Internet searches have found no other reports of malicious activity from this address.

- http://www.incidents.org/archives/intrusions/msg03681.html
  OS fingerprinting with destination port 515. Source address 216.232.46.223
- http://www.incidents.org/archives/intrusions/msg03595.html
  SYN scans of target network, destination port 515. Source address 208.41.56.210
- http://www.incidents.org/archives/intrusions/msg03491.html
  SYN scans of target network, destination port 515. Source address 65.31.25.19
- http://www.incidents.org/archives/intrusions/msg03439.html
  SYN scans of target network, destination port 515. Source address 218.63.192.35
- http://www.incidents.org/archives/intrusions/msg03276.html
  SYN scans of target network, destination port 515. Source address 66.149.34.37
- http://www.incidents.org/archives/intrusions/msg03100.html
  SYN scans of target network, destination port 515. Source address 210.121.195.156
  SYN scans of target network, destination port 515. Source address 140.109.245.4, source port 515
- http://www.incidents.org/archives/intrusions/msg03098.html
  SYN scans of target network, destination port 515. Source address 195.2.116.134

Also, this post at dsheild.org notes a sudden rise in port 515 scans in October of 2001, right before CERT advisory CA-2001-30 in November, 2001.
http://www1.dshield.org/pipermail/list/2001-October/001709.html

The following article discusses the Adore worm and reports that one of the vulnerabilities it exploits is lpd related. http://lwn.net/2001/0405/a/adore-ARIS.php3

7. Evidence of active targeting

Not every address in the target network was scanned. The addresses that were scanned appear to be randomly selected, which may indicate that they were actually selected from previous reconnaissance. Because the target network is frequently scanned for lpd servers it is not likely that the scan is a wrong number. With just this information, active targeting should be assumed.

8. Severity

- **Criticality: 4**
  The targeted systems may have been chosen as a result of previous reconnaissance, but there is no evidence that they are critical systems.
- **Lethality: 5**
  If a vulnerable system is located and exploited it will be under complete control of the attacker.
- **Severity, System Countermeasures: 1**
  We don't know what countermeasures are employed on the system so we must assume the worst.
- **Severity, Network Countermeasures: 1**
  We don't know what countermeasures are employed on the network so we must assume the worst.

(4 + 5) – (1 + 1) = 7

9. Defensive Recommendations

The following actions should eliminate risks of exploitation of lpd servers.

- Disable lpd servers on systems where they are not needed.
- Install the latest vendor security patches.
- Block inbound TCP port 515 at border routers or firewalls.
- Use TCPwrappers or similar host based security to only provide lpd services to trusted hosts.
- Monitor all networks for lpd service activity.

10. Multiple choice test question

In October of 2001 a surge in scans of TCP port 515 was reported. What service was the target of these scans?

a) ftp
b) telnet
c) lpd
d) dns
e) SubSeven

Answer C: The lpd service listens on TCP port 515 and was the subject of CERT advisories in November, 2001.

Detect #2 – Trin00 Detect

```
[**] [1:237:1] DDOS Trin00:MastertoDaemon(defaultpassdetected!) [**]
[Classification: Attempted Denial of Service] [Priority: 2]
02/07-10:55:13.413350 68.40.179.13:62924 -> A.NET.34.155:27444
UDP TTL:235 TOS:0x0 ID:9 IpLen:20 DgmLen:39
Len: 19
[Xref => http://www.whitehats.com/info/IDS197]
```

1. Source of Trace

My home network.

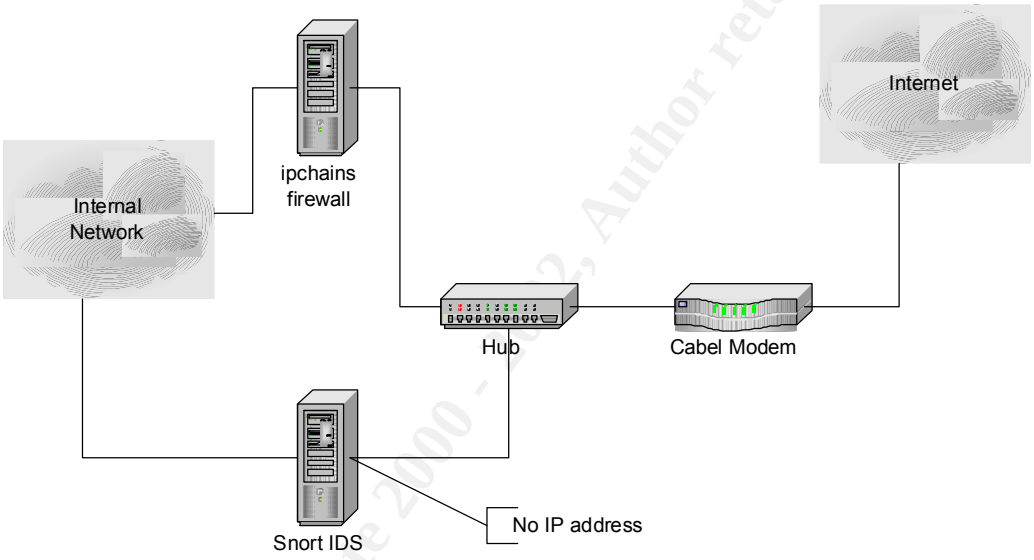


Figure 1 - Diagram of Home Network

2. Source of Detect

The detect was generated by a Snort IDS with the Snort 1.8.1 rule set. (https://www.snort.org/) Specifically, the detect was generated by the following Snort rule:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 27444 (msg:"DDOS
Trin00\:MastertoDaemon(defaultpassdetected!)"; content:"l44adsl";
reference:arachnids,197; classtype:attempted-dos; sid:237; rev:1;)
```

3. Probability the source address was spoofed

The attacker is sending Trin00 PNG commands to the target and expects a PONG response to indicated that the daemon is ready and waiting for other commands. If the

source address was spoofed the attacker would not receive any confirmation that the daemon is ready.

4. Description of Attack

Trin00 is a distributed denial of service of service tool (DDoS). An attacker sends commands to Trin00 master servers, which in turn send commands to Trin00 daemon clients. The purpose is to coordinated the clients to execute packet based attacks against one or more victims, causing the victim's network resources to be exhausted.

The Trin00 master server communicates with daemon clients on UDP port 27444. The master server sends commands to the daemon clients along with a password. The password is meant to keep others from taking over the daemon client. Here is the detailed packet log produced by Snort for this alert:

```
[**] DDOS Trin00:MastertoDaemon(defaultpassdetected!) [**]
02/07-10:55:13.413350 68.40.179.13:62924 -> A.NET.34.155:27444
UDP TTL:235 TOS:0x0 ID:9 IpLen:20 DgmLen:39
Len: 19
70 6E 67 20 6C 34 34 61 64 73 6C              png l44adsl
```

In this alert the master server sent the command 'png' with the password '144adsl' to the daemon client. 144adsl is the default password for connections from the master server to the daemon client. png is a test to see if the daemon client is ready to receive commands from the master server. If the daemon client is ready it will respond with the string 'PONG' to UDP port 31335 on the master server.

A detailed analysis of Trin00 can be found at:
http://staff.washington.edu/dittrich/misc/trinoo.analysis.

A candidate CVE entry can be found at:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0138.

5. Attack Mechanism

This alert does not indicate an attack, but an attempt of a Trin00 master server to check for the status of, or existence of a Trin00 daemon client. If the daemon client exists it is usually the result of a previous attack the target. Attackers scan networks for systems that are vulnerable to remote buffer overrun exploits and use those exploits to set up Trin00 systems.

The ipchains firewall log of the target indicates that the connection from the master server to the daemon client was denied by policy:

```
Feb  7 10:55:05 pengy kernel: Packet log: input DENY eth1 PROTO=17
68.40.179.13:63047 A.NET.34.155:27444 L=39 S=0x00 I=9 F=0x0000 T=235
```

6. Correlation

No reports of Trin00 activity originating from host `68.40.179.13` were found using Internet search engines. Use of Trin00 for DDoS attacks has been widely reported and has been described in CERT incident note IN-99-07.
(http://www.cert.org/incident_notes/IN-99-07.html)

Distributed Denial of Service attacks have been responsible for the interruption for a number of well known networks including, the Yahoo[1] network in February of 2000 and whitehouse.gov[2] website in May of 2001.

7. Evidence of active targeting

There is no prior history of Trin00 activity to or from this host, and the firewall policy for the target has always denied connection attempts of this type. This makes it highly unlikely that this host was specifically targeted. More likely, the attacking host is scanning networks for pre-existing Trin00 daemons to take over.

8. Severity

- **Criticality: 5**
  The target system is a firewall. If the firewall is compromised the internal network is open to attack.
- **Lethality: 1**
  This is not really an attack. Even if the port was accessible no harm would be done unless the system was already compromised, and even then the system would be used to attack other systems, not to harm itself.
- **Severity, System Countermeasures: 5**
  The system is up to date, offers minimal services, and is protected by its own firewall rules.
- **Severity, Network Countermeasures: 5**
  The system is a firewall which is protected by its own rules. The network of the system's external interface is monitored by an IDS.

(5 + 1) – (5 + 5) = -4

9. Defensive Recommendations

Since this was not really an attack there are no defenses to deploy. However a number of actions that can be taken to help prevent the use of systems in DDoS attacks.

- Keep systems up to date with security patches to prevent compromises that will allow DDoS tools from being installed.
- DDoS attacks usually send large amounts of traffic to the victim, with spoofed source addresses in the packets. Preventing network traffic with spoofed source

addresses from leaving your network will help prevent systems on your network from being used in DDoS attacks.

- As in this case, using an IDS with signatures to recognize DDoS traffic will help detect DDoS activity so that it can be stopped and infected systems located and cleaned.

10. Multiple choice test question

Traffic to or from UDP port 27444 may indicate the presence of which hacker tool?

a) Tribe Flood Network
b) Trin00
c) Code Red
d) BackOrifice

Answer B: By default, Trin00 master servers use UDP port 27444 to communicate with the daemon client.

Detect #3 – Back Orifice Access

```
[**] [105:1:1] spp_bo: Back Orifice Traffic detected (key: 31337) [**]
02/11-11:13:26.005583 24.2.97.31:63015 -> A.NET.34.155:31337
UDP TTL:44 TOS:0x0 ID:0 IpLen:20 DgmLen:46 DF
Len: 26

[**] [1:116:3] BACKDOOR BackOrifice access [**]
[Classification: Misc activity] [Priority: 3]
02/11-11:13:26.005583 24.2.97.31:63015 -> A.NET.34.155:31337
UDP TTL:44 TOS:0x0 ID:0 IpLen:20 DgmLen:46 DF
Len: 26
[Xref => http://www.whitehats.com/info/IDS399]
```

1. Source of Trace

My home network. (See Figure 1 - Diagram of Home Network)

2. Source of Detect

The detect was generated by a Snort IDS with the Snort 1.8.1 rule set. (https://www.snort.org/) The first detect was generated by the following Snort rule:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 31337 (msg:"BACKDOOR BackOrifice
access"; content: "|ce63 d1d2 16e7 13cf 39a5 a586|";
reference:arachnids,399; sid:116;  classtype:misc-activity; rev:3;)
```

The second alert was generated by the Snort BackOrifice preprocessor…

3. Probability the source address was spoofed

The attacker is trying to access a BackOrifice server on the target. Spoofing the source address would prevent the attacker from receiving replies from the connection attempts, making in unlikely that the source address was spoofed.

4. Description of Attack

Back Orifice is a remote administration utility created and released by the Cult of the Dead Cow (http://www.cultdeadcow.com/) in August of 1998. It is frequently installed by hackers on compromised Windows 95/98 systems to allow them easy control of the system is the future. Some of the features of Back Orifice are:

- The ability to get detailed system information.
- Complete files system and registry control.
- The ability to list, kill, and spawn processes.
- Control of network resources, shares, etc.
- An integrated packet sniffer.
- Packet redirection to any TCP or UDP address and port.
- Extensible plug-in architecture.

Communication with the Back Orifice server use the UDP protocol, server port 31337 (hacker speak for elite) by default. Back Orifice communications use a simple encryption scheme but can be detected using the content in the Snort signature above.

Comprehensive information about Back Orifice can be found at the following locations:

- NorthWest Internet website, http://www.nwinternet.com/~pchelp/bo/bo.html.
- ISS website, http://www.iss.net/security_center/alerts/advise8.php.
- CERT website, http://www.iss.net/security_center/alerts/advise8.php.

Back Orifice has a candidate entry for the CVE list, CAN-1999-0660, which can be found at: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660.

5. Attack Mechanism

A 'BackOrifice Access' alert is triggered when a Back Orifice server is contacted by a Back Orifice client. This might be considered an attack if the Back Orifice server was installed by an authorized person for legitimate use and an attacker is attempting to use the server without authorization. More likely the attacker has already compromised the target, installed the Back Orifice server, and is now attempting to use the installed server.

6. Correlation

The Snort IDS also detected a number of port scans and attempts to use other non-existent services by the attacking host in the same time frame. No reports of malicious activity from this host have been found in Internet searches.

The attacking host is part of the @Home network, which is the broadband provider to many home computers users who are unaware of good security practices, and thus contains many compromised machines and is a frequent source of malicious traffic.

```
# whois 24.2.97.31@whois.arin.net
[whois.arin.net]
@Home Network (NETBLK-ATHOME)   ATHOME                 24.0.0.0 -
24.23.255.255
@Home Network (NETBLK-TN-IMEDIA-200-1) TN-IMEDIA-200-124.2.96.0 -
24.2.103.255
```

7. Evidence of active targeting

Back Orifice runs on Windows 95/98 systems. The target system in a Linux computer. This means that either the attacker is just fishing for systems with Back Orifice installed, or the target may get its address by DHCP and its current address formerly belonged to a system that had Back Orifice installed.

The additional Snort alerts showing port scan and attempts to exploit non-existent services indicate that the attacker is just fishing for exposed services and backdoors, possibly with an automated scanning tool that the attacker doesn't understand how to use efficiently. This is demonstrated when the attacker does a port scan and then tries exploit against ports that are blocked by the firewall rules.

8. Severity

- **Criticality: 5**
  The target system is a firewall. If the firewall is compromised the internal network is open to attack.
- **Lethality: 5**
  If the attacker successfully connects to a Back Orifice server the system is under complete control of the attacker.
- **Severity, System Countermeasures: 5**
  The system is up to date, offers minimal services, and is protected by its own firewall rules. This ipchains log entry demonstrates that the UDP port is blocked:

  ```
  Feb 11 11:13:16 pengy kernel: Packet log: input DENY eth1 PROTO=17
  24.2.97.31:63111 A.NET.34.155:31337 L=46 S=0x00 I=0 F=0x4000 T=44
  ```

  The system in a Linux system and not compatible with Back Orifice.
- **Severity, Network Countermeasures: 5**
  The system is a firewall which is protected by its own rules. The network of the system's external interface is monitored by an IDS.

(5 + 5) – (5 + 5) = 0

9. Defensive Recommendations

The most common way that systems are infected with Back Orifice is by users executing unknown e-mail attachments or downloading and installing programs from unknown sources. Several Windows anti-virus programs detect Back Orifice but do not remove it when it is running. The BODetect program can be run continuously and when kill Back Orifice whenever it runs.

http://www.cbsoftsolutions.com/Products/bodetect.htm

10. Multiple choice test question

Which of the following choices is the best way to protect systems from becoming infected with Back Orifice?

a) Block UDP port 31337 at the border router of firewall.
b) Install anti-virus software on all systems.
c) Install the latest Microsoft security patches.
d) Educate users not to execute e-mail attachments of programs of unknown origin.

Answer D: User education is the best method of protection against Back Orifice. Blocking a specific port will not work if the port has been changed from the default. Anti-virus packages do not reliably detect and remove Back Orifice. Microsoft's position is that Back Orifice does not exploit any security problems in its operating systems.

Detect #4 – TCP Port 0 Traffic

```
[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:56:58.604613 199.6.47.245:62931 -> A.NET.34.155:0
TCP TTL:235 TOS:0x0 ID:60748 IpLen:20 DgmLen:40
******S* Seq: 0x4B019277  Ack: 0x0  Win: 0x1234  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:56:58.617324 199.6.47.245:62932 -> A.NET.34.155:0
TCP TTL:235 TOS:0x0 ID:60749 IpLen:20 DgmLen:40
***A**S* Seq: 0x4B019277  Ack: 0x0  Win: 0x1234  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:56:58.617577 A.NET.34.155:0 -> 199.6.47.245:62932
TCP TTL:255 TOS:0x0 ID:22675 IpLen:20 DgmLen:40
*****R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:56:58.637755 A.NET.34.155:0 -> 199.6.47.245:62933
TCP TTL:255 TOS:0x0 ID:22676 IpLen:20 DgmLen:40
***A*R** Seq: 0x0  Ack: 0x4B019277  Win: 0x0  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
```

```
02/09-01:56:58.672051 199.6.47.245:62934 -> A.NET.34.155:0
TCP TTL:235 TOS:0x0 ID:60751 IpLen:20 DgmLen:40
***A***F Seq: 0x4B019277  Ack: 0x0  Win: 0x1234  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:56:58.672251 A.NET.34.155:0 -> 199.6.47.245:62934
TCP TTL:255 TOS:0x0 ID:22677 IpLen:20 DgmLen:40
*****R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:56:58.722698 A.NET.34.155:0 -> 199.6.47.245:62936
TCP TTL:255 TOS:0x0 ID:22678 IpLen:20 DgmLen:40
***A*R** Seq: 0x0  Ack: 0x4B019277  Win: 0x0  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:56:58.736313 199.6.47.245:62937 -> A.NET.34.155:0
TCP TTL:235 TOS:0x0 ID:60754 IpLen:20 DgmLen:40
12****S* Seq: 0x4B019277  Ack: 0x0  Win: 0x1234  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:57:16.005521 199.6.47.245:62940 -> A.NET.34.155:0
TCP TTL:44 TOS:0x0 ID:43524 IpLen:20 DgmLen:60 DF
******S* Seq: 0x201128F7  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 180781811 0 NOP WS: 0

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:57:19.006096 199.6.47.245:62940 -> A.NET.34.155:0
TCP TTL:44 TOS:0x0 ID:43525 IpLen:20 DgmLen:60 DF
******S* Seq: 0x201128F7  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 180782111 0 NOP WS: 0

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:57:24.997361 199.6.47.245:62940 -> A.NET.34.155:0
TCP TTL:44 TOS:0x0 ID:43526 IpLen:20 DgmLen:60 DF
******S* Seq: 0x201128F7  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 180782711 0 NOP WS: 0

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-02:29:19.407436 199.6.47.245:63055 -> A.NET.34.155:0
TCP TTL:235 TOS:0x0 ID:47826 IpLen:20 DgmLen:40
******S* Seq: 0x6ECC8C65  Ack: 0x0  Win: 0x1234  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-02:29:19.428761 199.6.47.245:63056 -> A.NET.34.155:0
TCP TTL:235 TOS:0x0 ID:47827 IpLen:20 DgmLen:40
***A**S* Seq: 0x6ECC8C65  Ack: 0x0  Win: 0x1234  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
```

```
02/09-02:29:19.429019 A.NET.34.155:0 -> 199.6.47.245:63056
TCP TTL:255 TOS:0x0 ID:24464 IpLen:20 DgmLen:40
*****R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-02:29:19.450000 A.NET.34.155:0 -> 199.6.47.245:63057
TCP TTL:255 TOS:0x0 ID:24465 IpLen:20 DgmLen:40
***A*R** Seq: 0x0  Ack: 0x6ECC8C65  Win: 0x0  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-02:29:19.467208 199.6.47.245:63058 -> A.NET.34.155:0
TCP TTL:235 TOS:0x0 ID:47829 IpLen:20 DgmLen:40
***A***F Seq: 0x6ECC8C65  Ack: 0x0  Win: 0x1234  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-02:29:19.467428 A.NET.34.155:0 -> 199.6.47.245:63058
TCP TTL:255 TOS:0x0 ID:24466 IpLen:20 DgmLen:40
*****R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-02:29:19.506651 A.NET.34.155:0 -> 199.6.47.245:63060
TCP TTL:255 TOS:0x0 ID:24467 IpLen:20 DgmLen:40
***A*R** Seq: 0x0  Ack: 0x6ECC8C65  Win: 0x0  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-02:29:19.543083 199.6.47.245:63061 -> A.NET.34.155:0
TCP TTL:235 TOS:0x0 ID:47832 IpLen:20 DgmLen:40
12****S* Seq: 0x6ECC8C65  Ack: 0x0  Win: 0x1234  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-02:29:36.971889 199.6.47.245:63063 -> A.NET.34.155:0
TCP TTL:44 TOS:0x0 ID:22460 IpLen:20 DgmLen:60 DF
******S* Seq: 0x9AAA2E92  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 180975918 0 NOP WS: 0

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-02:29:39.970335 199.6.47.245:63063 -> A.NET.34.155:0
TCP TTL:44 TOS:0x0 ID:22461 IpLen:20 DgmLen:60 DF
******S* Seq: 0x9AAA2E92  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 180976218 0 NOP WS: 0

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-02:29:45.966749 199.6.47.245:63063 -> A.NET.34.155:0
TCP TTL:44 TOS:0x0 ID:22462 IpLen:20 DgmLen:60 DF
******S* Seq: 0x9AAA2E92  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 180976818 0 NOP WS: 0
```

1. Source of Trace

My home network. (See Figure 1 - Diagram of Home Network)

2. Source of Detect

The detect was generated by a Snort IDS with the Snort 1.8.1 rule set.

Snort rule:

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD TRAFFIC tcp port 0
traffic"; sid:524;  classtype:misc-activity; rev:3;)
```

3. Probability the source address was spoofed

It appears that the attacker is trying to use TCP flags to fingerprint the operating system of the target. To do that any responses from the target must be able to reach the attacker. The source address is probably not spoofed.

4. Description of Attack

This attack appears to be operating system fingerprinting. OS fingerprinting is a reconnaissance technique used by hackers to determine what operating system a target is running. The hacker can then use this information to determine what vulnerabilities on the target could be exploited.

5. Attack Mechanism

To determine the OS version of the target, the attacker crafts a series of TCP packets and send them to the target. By varying settings in the TCP header a unique response can be stimulated from the target. The response is unique of each OS because the TCP/IP stack implementation for each OS is slightly different, usually as the result of differing interpretations of the TCP/IP specification.

The following two packets taken from the trace above show a stimulus packet followed by a response packet.

```
[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:56:58.672051 199.6.47.245:62934 -> A.NET.34.155:0
TCP TTL:235 TOS:0x0 ID:60751 IpLen:20 DgmLen:40
***A***F Seq: 0x4B019277  Ack: 0x0  Win: 0x1234  TcpLen: 20

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:56:58.672251 A.NET.34.155:0 -> 199.6.47.245:62934
TCP TTL:255 TOS:0x0 ID:22677 IpLen:20 DgmLen:40
*****R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
```

The stimulus packet has the ACK and FIN bits set and has a destination port of zero. This packets is unexpected by the target because there is no service on port zero and having both FIN and ACK bits set is unusual. The target responds with a TCP reset.

The next packet from the trace shows a stimulus packet with the two reserved bits set. The target did not respond to this stimulus.

```
[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
02/09-01:56:58.736313 199.6.47.245:62937 -> A.NET.34.155:0
TCP TTL:235 TOS:0x0 ID:60754 IpLen:20 DgmLen:40
12****S* Seq: 0x4B019277  Ack: 0x0  Win: 0x1234  TcpLen: 20
```

With enough variations in TCP flags and options the attacker can determine with a high degree of accuracy which OS version the target is running.

In this attack no particular service is being targeted but once the attacker has determined the OS version it is likely that the attacker will try to exploit known vulnerabilities in the target.

More information about OS fingerprinting can found at the following resources:

- http://www.insecure.org/nmap/nmap-fingerprinting-article.html
- http://www.sans.org/newlook/resources/IDFAQ/TCP_fingerprinting.htm

6. Correlation

No evidence of the attacking IP targeting any other hosts could be found, but OS fingerprinting is a widely used hacking technique. The most popular fingerprinting tool is Nmap (http://www.insecure.org/), which is a port scanner with fingerprinting capabilities.

Fingerprinting with TCP reserved bits set was reported by Laurie@edu in the following reports:

- http://www.incidents.org/archives/y2k/020601-1000.htm
- http://www.incidents.org/archives/y2k/011601-1430.htm

7. Evidence of active targeting

The packets in the trace are targeted against a specific host, but the IDS only monitors one IP address which would not detect if the stimulus packets are a part of a larger scan. Given the source IP address there is no indication that this is a wrong number. It is unusual for OS fingerprinting to be conducted on a large scale, so it is likely that this is a case of active targeting.

8. Severity

- **Criticality: 5**
  The target system is a firewall. If the firewall is compromised the internal network is open to attack.
- **Lethality: 2**
  Successful reconnaissance will not result in a compromise but may lead to one.
- **Severity, System Countermeasures: 5**
  The system is up to date, offers minimal services, and is protected by its own firewall rules. This ipchains log entry demonstrates that the UDP port is blocked:

```
Feb  9 01:56:58 pengy kernel: Packet log: input DENY eth1 PROTO=6
199.6.47.245:62937 A.NET.34.155:0 L=40 S=0x00 I=60754 F=0x0000 T=235
```

- **Severity, Network Countermeasures: 5**
  The system is a firewall which is protected by its own rules. The network of the system's external interface is monitored by an IDS.

(5 + 2) – (5 + 5) = -3

9. Defensive Recommendations

If a target is directly reachable from the Internet there is little that can be done to prevent OS fingerprinting techniques from being used. Using a stateful firewall can prevent stimulus pockets from reaching the target. Using an application proxy will help protect the target and any response packets will represent the proxies TCP/IP stack and not the stack of the target.

Some countermeasures are discussed by Rik Farrow at:
http://www.spirit.com/Network/net0900.txt.

10. Multiple choice test question

A TCP packet with the reserved bits set captured by an IDS may indicate what?

a) An attempt to exploit a vulnerability on the target.
b) An indication of packets being mangled by a malfunctioning router.
c) An attempt to fingerprint the target's operating system.
d) The use of TCP options.

Answer C: TCP flags, options, and reserved bits are used to fingerprint a target's operating system.

Detect #5 – TCP Port 23 Probes

```
Dec 12 03:37:50 24.36.182.39:2077 -> a.b.c.20:23 SYN ******S*
Dec 12 03:37:50 24.36.182.39:2090 -> a.b.c.33:23 SYN ******S*
Dec 12 03:37:50 24.36.182.39:2108 -> a.b.c.51:23 SYN ******S*
Dec 12 03:37:50 24.36.182.39:2116 -> a.b.c.59:23 SYN ******S*
Dec 12 03:37:50 24.36.182.39:2119 -> a.b.c.62:23 SYN ******S*
Dec 12 03:37:53 24.36.182.39:2139 -> a.b.c.82:23 SYN ******S*
Dec 12 03:37:53 24.36.182.39:2140 -> a.b.c.83:23 SYN ******S*
Dec 12 03:37:53 24.36.182.39:2158 -> a.b.c.101:23 SYN ******S*
Dec 12 03:37:54 24.36.182.39:2204 -> a.b.c.147:23 SYN ******S*
Dec 12 03:38:00 24.36.182.39:2240 -> a.b.c.183:23 SYN ******S*
Dec 12 03:38:00 24.36.182.39:2836 -> a.b.f.14:23 SYN ******S*
Dec 12 03:38:03 24.36.182.39:3015 -> a.b.f.190:23 SYN ******S*
Dec 12 03:38:03 24.36.182.39:3030 -> a.b.f.205:23 SYN ******S*
Dec 12 03:38:04 24.36.182.39:3077 -> a.b.f.252:23 SYN ******S*
```

1. Source of Trace

Laurie Zirkle, Incidents mailing list archive.
http://www.incidents.org/archives/intrusions/msg02889.html

2. Source of Detect

The detect is from an unknown source. There is no indication that the connections were denied or allowed so it is probably not a firewall log. The detect may be from an IDS with rule to record all connection attempts to TCP port 23 (telnet). The log format is:

```
Date Stamp | Source IP:Port -> Destination IP:Port | State | TCP Flags Set
Dec 12 03:37:50 | 24.36.182.39:2077 -> a.b.c.20:23 | SYN | ******S*
```

3. Probability the source address was spoofed

This appears to be an attempt to find or exploit telnet servers. One way to find telnet servers is for the attacker to attempt a 3-way TCP handshake with the target. If there is an active telnet server then the handshake should complete. If there is no active telnet server then the attacker will expect a TCP reset.

To exploit a vulnerable telnet server the attacker must complete the handshake with the server so that it can then send data to execute the exploit. Either way spoofing the source address would defeat the attacker's goal.

Additionally this source IP address has been recorded in other probes against the target network, which means that the attacker is using the same spoofed IP address, or the source address is not spoofed. Based on this evidence the source address is probably not spoofed.

4. Description of Attack

Many vulnerabilities in telnet server implementations have been identified in the past three years. These vulnerabilities can be group into three areas: denial of service, buffer overflow, and information disclosure. Only attacks against more common systems and attacks that don't require user level system access will be explored.

Buffer Overflow:

- CERT Advisory CA-2001-21 Buffer Overflow in telnetd
- CVE-1999-0230 Buffer overflow in Cisco 7xx routers through the telnet service. http://www.cert.org/advisories/CA-2001-21.html
- CVE-2000-0166 Buffer overflow in the InterAccess telnet server TelnetD allows remote attackers to execute commands via a long login name.
- CAN-2001-0554 ** CANDIDATE (under review) ** Buffer overflow in BSD-based telnetd telnet daemon on various operating systems allows remote attackers to execute arbitrary commands via a set of options including AYT (Are You There), which is not properly handled by the telrcv function.
- CAN-2001-0797 ** CANDIDATE (under review) ** Buffer overflow in login in various System V based operating systems allows remote attackers to execute arbitrary commands via a large number of arguments through services such as telnet and rlogin.
- CAN-2002-0020 ** CANDIDATE (under review) ** Buffer overflow in telnet server in Windows 2000 and Interix 2.2 allows remote attackers to execute arbitrary code via malformed protocol options.

Denial of Service:

- CVE-1999-0087 Denial of service in AIX telnet can freeze a system and prevent users from accessing the server.
- CVE-1999-0273 Denial of service through Solaris 2.5.1 telnet by sending ^D characters.
- CVE-1999-0416 Vulnerability in Cisco 7xx series routers allows a remote attacker to cause a system reload via a TCP connection to the router's TELNET port.
- CVE-1999-0740 Remote attackers can cause a denial of service on Linux in.telnetd telnet daemon through a malformed TERM environmental variable.
- CVE-2000-0268 Cisco IOS 11.x and 12.x allows remote attackers to cause a denial of service by sending the ENVIRON option to the Telnet daemon before it is ready to accept it, which causes the system to reboot.
- CVE-2001-0041 Memory leak in Cisco Catalyst 4000, 5000, and 6000 series switches allows remote attackers to cause a denial of service via a series of failed telnet authentication attempts.
- CVE-2001-0345 Microsoft Windows 2000 telnet service allows attackers to prevent idle Telnet sessions from timing out, causing a denial of service by creating a large number of idle sessions.

- CVE-2001-0346 Handle leak in Microsoft Windows 2000 telnet service allows attackers to cause a denial of service by starting a large number of sessions and terminating them.
- CVE-2001-0348 Microsoft Windows 2000 telnet service allows attackers to cause a denial of service (crash) via a long logon command that contains a backspace.
- CVE-2001-0351 Microsoft Windows 2000 telnet service allows a local user to make a certain system call that allows the user to terminate a Telnet session and cause a denial of service.
- CVE-2001-0427 Cisco VPN 3000 series concentrators before 2.5.2(F) allow remote attackers to cause a denial of service via a flood of invalid login requests to (1) the SSL service, or (2) the telnet service, which do not properly disconnect the user after several failed login attempts.

Information Disclosure:

- CVE-2001-0347 Information disclosure vulnerability in Microsoft Windows 2000 telnet service allows remote attackers to determine the existence of user accounts such as Guest, or log in to the server without specifying the domain name, via a malformed user ID.

Authentication Bypass:

- CVE-1999-0889 Cisco 675 routers running CBOS allow remote attackers to establish telnet sessions if an exec or superuser password has not been set.
- CAN-2000-1195 ** CANDIDATE (under review) ** telnet daemon (telnetd) from the Linux netkit package before netkit-telnet-0.16 allows remote attackers to bypass authentication when telnetd is running with the -L command line option.
- CERT Advisory CA-1995-14 Telnetd Environment Vulnerability (CVE-1999-0073) http://www.cert.org/advisories/CA-1995-14.html

5. Attack Mechanism

Buffer Overflow:

Buffer overflows are probably the most dangerous vulnerability, allowing the attacker to execute arbitrary commands on the target with the end result being complete control of the target. The mechanism for buffer overflows is described in detect #1.

Denial of Service:

The purpose of a denial of service (DoS) attack is to stimulate a condition in the target which legitimate use of the target or a service on the target is prevented[8]. Ways in which this is accomplished include:

- Flooding a network with traffic to prevent legitimate network traffic.
- Consuming limited resources on the target.

- Disrupting the operation of the target, such as a system freeze.

For example, in CVE entry, CVE-1999-0740 and attacker can take advantage of the Linux telnetd servers support of RFC's 1408 and 1572. These RFC's define a method for telnet clients to specify environment variables to a telnet server, allowing the ability to transfer environment variables from one system to another. An improperly formatted TERM environment variable could cause the system to perform actions that would result in a system crash.

Information Disclosure:

The purpose of an information disclosure attack is to trick a target into revealing information that could be useful in future attacks on the target. In CVE-2001-0347 an information disclosure attack on the Windows 200 telnet service is documented. Under normal conditions a user authenticating to the Windows 2000 telnet server must specify which domain the account belongs to. The user must know a valid domain name and a valid account name within that domain[10].

The vulnerability involves appending special characters to the account name in the authentication process. This causes the telnet services to search it's domain and all trusted domains for a matching account name. Once a matching account has been found the authentication process must be completed. At this point the attacker can use brute force to obtain the password of the account.

Authentication Bypass:

Authentication bypass attacks involve tricking the target into authenticating the attacker without the attacker providing proper credentials. Documented in CVE-1999-0073, the same environment variable setting feature described in the Denial of Service section above is used to set the environment variable that specifies the path that operating system searches for shared libraries. By altering the variable to use alternative libraries, possibly installed by the attacker through another mechanism, the authentication process may be bypassed.

CVE entry CAN-2000-1195 documents a Linux telnetd vulnerability. When the '-L' option to telnetd is specified by the administrator so that an alternate login program may be used, the attacker may overwrite this information. The result is that in most cases telnetd will stop working, but sometimes the authentication process may be bypassed[9].

6. Correlation

Two other incidents of host 24.36.182.39 were reported by Laurie Zirkle. In both cases a select set of hosts on the target network were scanned for portmap servers (TCP port 111). Some hosts were scanned on both 11/9 and 11/12. On 11/12 host a.b.c.62 was scanned on both UDP and TCP ports 111. Snort and portmap logs also recorded that event, indicating that host a.b.c.62 was possibly the only host offering portmap services.

11/9/2001
http://www.incidents.org/archives/intrusions/msg02419.html

```
Nov  9 06:15:20 24.36.182.39:1779 -> a.b.c.14:111 SYN ******S*
Nov  9 06:15:17 24.36.182.39:1816 -> a.b.c.51:111 SYN ******S*
Nov  9 06:15:17 24.36.182.39:1827 -> a.b.c.62:111 SYN ******S*
Nov  9 06:15:17 24.36.182.39:1836 -> a.b.c.71:111 SYN ******S*
Nov  9 06:15:17 24.36.182.39:1861 -> a.b.c.96:111 SYN ******S*
Nov  9 06:15:17 24.36.182.39:1866 -> a.b.c.101:111 SYN ******S*
Nov  9 06:15:17 24.36.182.39:2093 -> a.b.d.73:111 SYN ******S*
Nov  9 06:15:20 24.36.182.39:1936 -> a.b.c.171:111 SYN ******S*
Nov  9 06:15:20 24.36.182.39:1946 -> a.b.c.181:111 SYN ******S*
Nov  9 06:15:20 24.36.182.39:1947 -> a.b.c.182:111 SYN ******S*
Nov  9 06:15:20 24.36.182.39:1960 -> a.b.c.195:111 SYN ******S*
Nov  9 06:15:20 24.36.182.39:2248 -> a.b.d.228:111 SYN ******S*
Nov  9 06:15:20 24.36.182.39:2338 -> a.b.e.63:111 SYN ******S*
Nov  9 06:15:21 24.36.182.39:2752 -> a.b.f.190:111 SYN ******S*
```

11/12/2001
http://www.incidents.org/archives/intrusions/msg02431.html

```
Nov 12 09:23:17 24.36.182.39:3631 -> a.b.c.12:111 SYN ******S*
Nov 12 09:23:15 24.36.182.39:3641 -> a.b.c.22:111 SYN ******S*
Nov 12 09:23:15 24.36.182.39:3655 -> a.b.c.36:111 SYN ******S*
Nov 12 09:23:15 24.36.182.39:3670 -> a.b.c.51:111 SYN ******S*
Nov 12 09:23:18 24.36.182.39:3676 -> a.b.c.59:111 SYN ******S*
Nov 12 09:23:18 24.36.182.39:3679 -> a.b.c.62:111 SYN ******S*
Nov 12 09:23:19 24.36.182.39:640 -> a.b.c.62:111 UDP
Nov 12 09:23:18 24.36.182.39:3688 -> a.b.c.71:111 SYN ******S*
Nov 12 09:23:18 24.36.182.39:3744 -> a.b.c.127:111 SYN ******S*
Nov 12 09:23:18 24.36.182.39:3800 -> a.b.c.183:111 SYN ******S*
Nov 12 09:23:19 hosthu portmap[3399]: connect from 24.36.182.39 to
getport(status): request from unauthorized host
Nov 12 09:23:19 hosthu snort: [1:583:2] RPC portmap request rstatd
[Classification: Decode of an RPC Query] [Priority: 2]: {UDP}
24.36.182.39:640 -> a.b.c.62:111
```

7. Evidence of active targeting

Hosts on the target network have been scanned on three separate occasions, indicating that the network has been actively targeted. Not all possible IP addresses on the target network were scanned indicating that these hosts may have been targeted based on previous reconnaissance. There is very little chance that this is a wrong number.

8. Severity

- **Criticality: 4**
  The targeted systems may have been chosen as a result of previous reconnaissance and have been attacked multiple times, but there is no evidence that they are critical systems.
- **Lethality: 5**
  If a vulnerable system is located and exploited it will be under complete control of the attacker.

- **Severity, System Countermeasures: 2**
  We don't know what countermeasures are employed on all the systems but the logs from 11/12 indicate that at least on one system portmap is configured with some security restrictions.
- **Severity, Network Countermeasures: 2**
  We know that Snort is monitoring at least some of the systems on the network but have limited knowledge of all defensive systems.

(5 + 5) – (2 + 2) = 7

9. Defensive Recommendations

The best defense against telnet vulnerabilities is to not use telnet at all. The SSH protocol is a much more secure alternative to SSH. A freely available implementation of the SSH protocol is available at http://www.openssh.org/. Some of the security features of SSH are:

- All communication, including authentication is encrypted using strong encryption.
- Protections from DNS spoofing and connection hijacking.
- Support for digital certificates for authentication.
- Never trusts the network.
- Support for tunneling other protocols through encrypted SSH connections.

In addition to end user and server systems, SSH is supported on some network equipment including recent version of Cisco IOS.

If eliminating telnet isn't an option, the following defensive measures should be taken:

- Make sure to install the latest vendor patches.
- Do not use telnet over un-trusted network links.
- Use firewall's and IDS's to monitor all telnet connections from external networks.
- Configure systems to disallow telnet access for the root account.
- Enforce the use of strong passwords on systems with telnet services.
- If possible one time passwords or two factor authentication for telnet access.

10. Multiple choice test question

Which option below is the best defense against telnet vulnerabilities?

a) Install latest vendor patches.
b) Use TCPwrappers to limit connections to telnet daemon.
c) Replace all use of telnet with ssh.
d) Block telnet connections at a border router or firewall.

Answer C: All the options above are good but eliminating the use of telnet is the best option. SSH is a much more secure alternative to telnet. All SSH communications are encrypted in addition to other protections built into the SSH protocol.

Assignment 3 – Analyze This

Executive Summary

- False Positives:

  The IDS rule set is generating a large number of alerts. Some of the most common alerts appear to be false positives. These should be investigated and if they are false positives the IDS rules should be modified to eliminate these alerts. This will make future analyses less complex by reducing the haystack to needle ratio.

- SNMP Passwords:

  Simple Network Management Protocol (SNMP) is being used with the default community strings, the equivalent of passwords. This puts SNMP managed devices at risk for information disclosure and modification. SNMP community strings should be changed from the defaults and SNMP version 3, with encrypted password support should be used if possible.

- Perimeter Protection:

  Filters should be installed on border routers or perimeter firewalls should be installed to eliminate some malicious traffic and block unsafe protocols that should not enter or leave the network. Out-of-spec packets with illegal flag combination or reserved bits set should be blocked by perimeter devices. Some known unfriendly networks should also be blocked.

- Tune IDS Rules:

  Some IDS rules are triggered solely on source or destination port numbers associated with well know trojans and attacks, such a SubSeven. If possible the rules should be tuned to alert based frequency of the port being detected per host. This will help to eliminate false positives that ephemeral ports will occasionally generate.

- Anti-virus updates appear to be working:

  The e-mail virus MYPARTY was release on January 29, 2002 and on that day 525 associated alerts were generated. The following 3 days no MYPARTY alerts were generated indicating that anti-virus software is being used and automatic updates of virus signatures are working.

Data Analyzed

The following Snort log files were collected from http://www.research.umbc.edu/~andy:

Alert Logs	Scan Logs	OOS Logs
alert.020126	scans.020126	oos_Jan.26.2002
alert.020127	scans.020127	oos_Jan.27.2002
alert.020128	scans.020128	oos_Jan.28.2002
alert.020129	scans.020129	oos_Jan.29.2002
alert.020130	scans.020130	oos_Jan.30.2002

Table 1- Data Files Analyzed

Alert Summary

# of Alerts	% of all Alerts	# of distinct SRC IPs	# of distinct DST IPs	Alert Type / Name
81575	23.33%	0	0	PORTSCAN DETECTED
70513	20.17%	136	615	spp_http_decode: IIS Unicode attack detected
68835	19.69%	112	1	connect to 515 from inside
32753	9.37%	13	10	MISC Large UDP Packet
28080	8.03%	193	202	SMB Name Wildcard
23815	6.81%	18	143	SNMP public access
13884	3.97%	101	16	ICMP Echo Request L3retriever Ping
5744	1.64%	11	13	spp_http_decode: CGI Null Byte attack detected
5545	1.59%	80	80	INFO MSN IM Chat data
5262	1.50%	113	148	High port 65535 udp - possible Red Worm - traffic
3409	0.97%	46	10	Watchlist 000220 IL-ISDNNET-990517
2067	0.59%	64	4	ICMP Echo Request Nmap or HPING2
1776	0.51%	144	1	ICMP Router Selection
1017	0.29%	24	1	WEB-IIS view source via translate header
900	0.26%	6	2	FTP DoS ftpd globbing
820	0.23%	34	49	ICMP Fragment Reassembly Time Exceeded
600	0.17%	123	6	Null scan!
525	0.15%	4	1	MYPARTY - Possible My Party infection
288	0.08%	18	16	ICMP Echo Request Windows
278	0.08%	83	1	WEB-IIS _vti_inf access
257	0.07%	81	1	WEB-FRONTPAGE _vti_rpc access
251	0.07%	4	7	ICMP Echo Request BSDtype
140	0.04%	17	7	WEB-MISC Attempt to execute cmd
137	0.04%	12	8	EXPLOIT NTPDX buffer overflow
113	0.03%	2	2	TFTP - Internal TCP connection to external tftp server
113	0.03%	1	1	ICMP Destination Unreachable (Communication Administratively Prohibited)
110	0.03%	14	5	MISC traceroute
75	0.02%	13	21	INFO Possible IRC Access

# of Alerts	% of all Alerts	# of distinct SRC IPs	# of distinct DST IPs	Alert Type / Name
54	0.02%	12	5	NMAP TCP ping!
53	0.02%	9	9	WEB-MISC compaq nsight directory traversal
51	0.01%	9	1	WEB-CGI scriptalias access
49	0.01%	49	3	INFO Outbound GNUTella Connect accept
46	0.01%	46	5	SCAN Synscan Portscan ID 19104
46	0.01%	2	20	INFO FTP anonymous FTP
44	0.01%	12	12	Possible trojan server activity
35	0.01%	3	16	WEB-MISC 403 Forbidden
32	0.01%	14	7	ICMP traceroute
31	0.01%	12	12	EXPLOIT x86 NOOP
29	0.01%	11	10	SCAN Proxy attempt
26	0.01%	1	18	INFO Napster Client Data
24	0.01%	3	5	ICMP Destination Unreachable (Protocol Unreachable)
24	0.01%	11	14	Attempted Sun RPC high port access
21	0.01%	6	6	Port 55850 tcp - Possible myserver activity - ref. 010313-1
19	0.01%	13	3	Incomplete Packet Fragments Discarded
16	0.00%	2	2	Russia Dynamo - SANS Flash 28-jul-00
16	0.00%	1	1	WEB-MISC whisker head
13	0.00%	1	3	INFO - ICQ Access
11	0.00%	3	3	High port 65535 tcp - possible Red Worm - traffic
9	0.00%	5	6	INFO - Possible Squid Scan
9	0.00%	4	8	Back Orifice
8	0.00%	2	2	SMB CD...
8	0.00%	2	1	WEB-MISC http directory traversal
7	0.00%	7	2	WEB-CGI formmail access
7	0.00%	7	2	MISC source port 53 to <1024
6	0.00%	3	2	Queso fingerprint
6	0.00%	2	2	ICMP Address Mask Reply
6	0.00%	2	1	SUNRPC highport access!
6	0.00%	1	6	WEB-IIS 5 Printer-beavuh
5	0.00%	3	3	Port 55850 udp - Possible myserver activity - ref. 010313-1
5	0.00%	2	1	TFTP - External UDP connection to internal tftp server
5	0.00%	1	3	ICMP Echo Request CyberKit 2.2 Windows
4	0.00%	4	4	EXPLOIT x86 setgid 0
4	0.00%	3	3	EXPLOIT x86 setuid 0
4	0.00%	3	1	INFO Inbound GNUTella Connect request
4	0.00%	1	1	Probable NMAP fingerprint attempt
2	0.00%	2	1	WEB-IIS encoding access

# of Alerts	% of all Alerts	# of distinct SRC IPs	# of distinct DST IPs	Alert Type / Name
2	0.00%	1	2	WEB-IIS Unauthorized IP Access Attempt
2	0.00%	1	1	Watchlist 000222 NET-NCFC
2	0.00%	1	1	RPC udp traffic contains bin sh
2	0.00%	1	1	NIMDA - Attempt to execute cmd from campus host
2	0.00%	1	1	ICMP SRC and DST outside network
2	0.00%	1	1	ICMP Address Mask Request
1	0.00%	1	1	WEB-MISC webdav search access
1	0.00%	1	1	WEB-IIS 5 .printer isapi
1	0.00%	1	1	Tiny Fragments - Possible Hostile Activity
1	0.00%	1	1	TFTP - Internal UDP connection to external tftp server
1	0.00%	1	1	MISC PCAnywhere Startup
1	0.00%	1	1	ICMP Echo Request Delphi-Piette Windows
1	0.00%	1	1	FTP EXPLOIT aix overflow
1	0.00%	1	1	BACKDOOR NetMetro File List
349,647				**Total Alerts**

Table 2 – Summary of Alert Frequency

In the 5 day period analyzed, 80 different alert types were logged and 349,647 individual alerts were logged. Clearly there is a large amount of malicious activity on the university network. Not every alert type can be investigated because of the large volume. The best approach is rank the alert types using severity and frequency, which is a judgment call of the analyst.

For instance, alerts such as *spp_http_decode: IIS Unicode attack detected* are very frequently but not very interesting because large number of automated scans that trigger this alert. More interesting are the alert types that receive relatively few alerts which indicates two likely scenarios:

1. The alert is a false positive, generated by normal traffic that happens to match an IDS rule.
2. The alert is specifically targeted to hosts based on previous reconnaissance of an attacker.

The alert which is generated as a result of previous reconnaissance indicates that the attacker is being more methodological in testing for weaknesses and choosing hosts to attack. Those hosts targeted for attack should be inspected for signs of intrusion and monitored for changes in alert generation before and after the attack. If the targeted host starts generating alerts after it has been specifically targeted for attack there is good probability that the attack was successful and the host has been compromised.

A total of 1,045 individual IP addresses generated alerts in the 5 day period. Of these 1,045 IP addresses, 41% of them belonged to MY.NET. The following table show the top ten IP addresses generating alerts from inside the MY.NET network and the percentage of all alerts generated.

Rank	# of Alerts	% of Alerts	Source IP
1	11807	4.40	MY.NET.70.177
2	10684	3.99	MY.NET.153.119
3	9481	3.54	MY.NET.153.111
4	7485	2.79	MY.NET.153.114
5	6983	2.60	MY.NET.153.122
6	6053	2.26	MY.NET.11.6
7	5911	2.21	MY.NET.153.123
8	5496	2.05	MY.NET.11.7
9	5414	2.02	MY.NET.153.118
10	4875	1.82	MY.NET.153.126
	74,189	**27.68**	**Total**

Table 3 – Top Ten Alert Sources On MY.NET

The following table show the top ten IP addresses generating alerts from outside the MY.NET network and the percentage of all alerts generated.

Rank	# of Alerts	% of Alerts	Source IP
1	10739	4.01	63.250.209.34
2	9859	3.68	63.250.211.165
3	7246	2.70	63.250.210.50
4	3109	1.16	203.231.232.15
5	1419	0.53	212.179.35.8
6	1104	0.41	212.179.35.118
7	841	0.31	202.58.33.70
8	677	0.25	64.152.216.77
9	421	0.16	64.152.108.142
10	404	0.15	64.161.36.66
	35,819	**13.36**	**Total**

Table 4 – Top Ten Alert Sources From Other Networks

A total of 1,221 individual IP addresses were the targets of alerts in the 5 day period. Of these 1,221 IP addresses, 36% of them belonged to MY.NET. The following table show the top ten targeted IP addresses inside the MY.NET network and the percentage of all alerts generated.

Rank	# of Alerts	% of Alerts	Destination IP
1	68836	25.68	MY.NET.150.198

2	27629	10.31	MY.NET.151.63
3	13018	4.86	MY.NET.11.6
4	12045	4.49	MY.NET.11.7
5	6445	2.40	MY.NET.152.109
6	3811	1.42	MY.NET.5.96
7	3181	1.19	MY.NET.153.195
8	2971	1.11	MY.NET.11.5
9	2014	0.75	MY.NET.151.114
10	1890	0.71	MY.NET.5.128
	141,840	**52.92**	**Total**

Table 5 – Top Ten Alert Destinations On MY.NET

The following table show the top ten IP targeted addresses outside the MY.NET network and the percentage of all alerts generated.

Rank	# of Alerts	% of Alerts	Destination IP
1	15788	5.89	211.115.213.202
2	5444	2.03	209.10.239.135
3	2719	1.01	64.12.184.141
4	2696	1.01	211.115.213.207
5	1776	0.66	224.0.0.2
6	1669	0.62	211.32.117.31
7	1495	0.56	211.32.116.112
8	1332	0.50	211.32.117.228
9	1318	0.49	64.12.180.21
10	1080	0.40	211.233.29.219
	35,317	**13.17**	**Total**

Table 6 – Top Ten Alert Destinations On Other Networks

The following table show the top ten targeted ports and the percentage of all alerts generated.

Rank	# of Alerts	% of Alerts	Destination Port
1	77528	28.92	80
2	68835	25.68	515
3	28080	10.48	137
4	23815	8.88	161
5	19272	7.19	-1*
6	4692	1.75	65535
7	3166	1.18	1863
8	1548	0.58	2368
9	1539	0.57	2327
10	1350	0.50	2833

	229,825	**85.73**	**Total**

Table 7 – Top Ten Targeted Ports

* Indicates alerts that don't have associated ports, such as ICMP packets.

Portscan Summary

Over the 5 day period 1,972,299 individual ports were scanned. The following table list the top 10 port scanners on MY.NET.

Rank	# of ports scanned	% of ports scanned	Source IP
1	455893	23.11	MY.NET.60.43
2	91544	4.64	MY.NET.6.49
3	91115	4.62	MY.NET.6.45
4	88902	4.51	MY.NET.6.48
5	86823	4.44	MY.NET.6.52
6	75547	3.83	MY.NET.6.50
7	42138	2.14	MY.NET.6.53
8	37832	1.92	MY.NET.6.60
9	21185	1.07	MY.NET.153.154
10	14404	0.73	MY.NET.153.211
	1,005,383	**51.01**	**Total**

Table 8 – Top Ten Port Scanners On MY.NET

The following table list the top 10 port scanners on outside the MY.NET network.

Rank	# of ports scanned	% of ports scanned	Source IP
1	45757	2.32	64.152.108.141
2	35987	1.82	64.152.108.142
3	26700	1.35	63.210.134.141
4	16851	0.85	205.188.228.65
5	13645	0.69	205.188.228.17
6	12983	0.66	205.188.228.33
7	8537	0.43	205.188.228.1
8	7614	0.39	66.38.185.143
9	5951	0.30	140.142.8.72
10	3863	0.20	216.106.172.157
	177,888	**9.01**	**Total**

Table 9 – Top Ten Port Scanners On Other Networks

Alert Analysis

The following alert types and hosts, ranked from most important to least important were chosen for further analysis based on number of alerts, criticality of alerts, activity of the hosts, and the judgment of the analyst.

Rank	Alert Type / Host
1	connect to 515 from inside
2	Host: MY.NET.70.177
3	Host: MY.NET.11.6
4	Host: MY.NET.11.7
5	BACKDOOR NetMetro File List
6	FTP EXPLOIT aix overflow
7	Hosts: 63.250.209.34. 63.250.211.165, 63.250.210.50
8	Host: 203.231.232.15
9	Hosts: 202.179.35.8, 202.179.35.118
10	Network: 64.152

Table 10 – Prioritized Table of Alerts and Hosts for Analysis

Snort rule format:

Information on writing Snort rules can be found at http://www.snort.org/docs/writing_rules/chap2.html#tth_chAp2.

"Snort rules are divided into two logical sections, the rule header and the rule options. The rule header contains the rule's action, protocol, source and destination IP addresses and netmasks, and the source and destination ports information. The rule option section contains alert messages and information on which parts of the packet should be inspected to determine if the rule action should be taken. "

In the rule above, the action is *alert*. The source is specified as any UDP packet (any IP, and port) and the destination is anywhere on $HOME_NET, port 161, which is the SNMP port. $HOME_NET is a variable containing a specification of hosts on the network the IDS is monitoring.

Inside the parentheses are the rule options. The *msg* option specifies the text to print when an alert is triggered. The *content* option causes an alert to trigger if the packet header matches the rule header and the packet payload includes the content text, in this case, "public."

Analysis: *connect to 515 from inside*

Log Sample:

```
01/26-12:52:53.144400  [**] connect to 515 from inside [**]
MY.NET.153.119:1648 -> MY.NET.150.198:515
01/26-12:52:53.146130  [**] connect to 515 from inside [**]
MY.NET.153.119:1648 -> MY.NET.150.198:515
01/26-12:52:53.147581  [**] connect to 515 from inside [**]
MY.NET.153.119:1648 -> MY.NET.150.198:515
01/26-12:52:53.149003  [**] connect to 515 from inside [**]
MY.NET.153.119:1648 -> MY.NET.150.198:515
```

Snort Rule:

```
alert tcp $HOME_NET any -> $HOME_NET 515 (msg:"connect to 515 from inside";
classtype:misc-activity;)
```

This alert has the interesting property of being the 3rd most frequent alert but having only 1 distinct destination addresses (MY.NET.150.198). As illustrated below, all but 2 hosts are on the MY.NET.152 and MY.NET.153 networks.

```
# ./alert_stats.pl '515 from inside' alert.all

Source IP       -> Destination IP       # of alerts
MY.NET.153.210  -> MY.NET.150.198       alerts: 47
MY.NET.153.105  -> MY.NET.150.198       alerts: 428
MY.NET.153.211  -> MY.NET.150.198       alerts: 539
MY.NET.153.106  -> MY.NET.150.198       alerts: 941
MY.NET.152.160  -> MY.NET.150.198       alerts: 48
      .
      .     Data Pruned
      .
MY.NET.153.189  -> MY.NET.150.198       alerts: 112
MY.NET.153.110  -> MY.NET.150.198       alerts: 1461
MY.NET.153.111  -> MY.NET.150.198       alerts: 9136
MY.NET.153.112  -> MY.NET.150.198       alerts: 1122
MY.NET.153.113  -> MY.NET.150.198       alerts: 429
MY.NET.153.114  -> MY.NET.150.198       alerts: 7018
MY.NET.153.115  -> MY.NET.150.198       alerts: 2206
MY.NET.152.170  -> MY.NET.150.198       alerts: 32
MY.NET.152.171  -> MY.NET.150.198       alerts: 16
MY.NET.152.244  -> MY.NET.150.198       alerts: 38
MY.NET.153.150  -> MY.NET.150.198       alerts: 12
MY.NET.153.118  -> MY.NET.150.198       alerts: 4678
MY.NET.152.172  -> MY.NET.150.198       alerts: 10
MY.NET.153.119  -> MY.NET.150.198       alerts: 7506
MY.NET.153.152  -> MY.NET.150.198       alerts: 118
MY.NET.153.153  -> MY.NET.150.198       alerts: 120
      .
      .     Data Pruned
      .
MY.NET.153.120  -> MY.NET.150.198       alerts: 1316
MY.NET.152.46   -> MY.NET.150.198       alerts: 467
MY.NET.153.121  -> MY.NET.150.198       alerts: 668
MY.NET.152.216  -> MY.NET.150.198       alerts: 7
MY.NET.153.122  -> MY.NET.150.198       alerts: 3778
MY.NET.153.123  -> MY.NET.150.198       alerts: 693
MY.NET.152.250  -> MY.NET.150.198       alerts: 9
```

```
MY.NET.153.124  -> MY.NET.150.198        alerts: 944
MY.NET.152.251  -> MY.NET.150.198        alerts: 118
MY.NET.153.125  -> MY.NET.150.198        alerts: 593
MY.NET.153.126  -> MY.NET.150.198        alerts: 4722
MY.NET.153.127  -> MY.NET.150.198        alerts: 1292
MY.NET.152.180  -> MY.NET.150.198        alerts: 116
MY.NET.152.181  -> MY.NET.150.198        alerts: 52
MY.NET.153.160  -> MY.NET.150.198        alerts: 453
      .
      .     Data Pruned
      .
MY.NET.88.148   -> MY.NET.150.198        alerts: 2175
MY.NET.88.181   -> MY.NET.150.198        alerts: 83
MY.NET.152.21   -> MY.NET.150.198        alerts: 22
MY.NET.152.22   -> MY.NET.150.198        alerts: 513
```

(Hosts on the top talkers list are in red.)

Description of Alert:

Port 515 is the well known port of the Unix lpd print service. Some lpd servers are susceptible to a remote buffer overflow which allows the attacker to execute arbitrary code with the privileges of the server. Details of the attack can be found in CERT advisory CA-2001-15 (http://www.cert.org/advisories/CA-2001-15.html), CA-2001-30 (http://www.cert.org/advisories/CA-2001-30.html), and in the Common Vulnerabilities and Exposures database – CVE (http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=LPD).

Defensive Recommendations:

The following actions will reduce exposure to this attack:

- Disable lpd service on this host if it is not needed.
- Install latest vendor patches.
- Use TCPwrappers or a host based firewall to limit access to the lpd service to only trusted hosts.

Also MY.NET.150.198 should be analyzed for signs of intrusion and cleaned if necessary.

Correlation:

In **Table 3** the network MY.NET.153 has 7 of the top 10 addresses generating alerts on MY.NET.

```
# grep --  "MY\.NET\.153.*->" alert.all | cut -b 23-53 | sort | uniq -c
      1  [**] Attempted Sun RPC high po
  62432  [**] connect to 515 from insid
     55  [**] High port 65535 udp - pos
      8  [**] ICMP Destination Unreacha
```

```
     1  [**] ICMP Echo Request Delphi-
    36  [**] ICMP Echo Request L3retri
    12  [**] ICMP Echo Request Windows
   754  [**] ICMP Fragment Reassembly
   583  [**] ICMP Router Selection [**
     3  [**] ICMP traceroute  [**] MY.
  1847  [**] INFO MSN IM Chat data [**
    26  [**] INFO Napster Client Data
    35  [**] INFO Possible IRC Access
   525  [**] MYPARTY - Possible My Par
   101  [**] SMB Name Wildcard [**] MY
  1901  [**] SNMP public access [**] M
  5552  [**] spp_http_decode: CGI Null
 62379  [**] spp_http_decode: IIS Unic
```

Using the following command it was determined that the only alerts generate by MY.NET.150.198 are SNMP Public Access (3698) and SMB Name Wildcard (1), which are not critical alerts.

```
# grep "MY.NET.150.198.*->" alert.all | cut -b 23-43 | sort | uniq -c
```

The lack of critical alerts generated from MY.NET.150.198 and absence of critical alerts destined for MY.NET.150.198 indicate that it is probably a departmental print server generating false positives in the IDS. This reduces the concern of MY.NET.150.198 having the top position in **Table 5**. Not knowing the true network address of MY.NET limits our ability to correlate with outside sources.

⇒ The analysis above suggest that the 7 hosts in **Table 3** on the MY.NET.153 network are probably using a departmental print server. Removing those alerts would cause the hosts to fall out of the top 10. The next 3 analyses will concentrate on the remaining 3 hosts in **Table 3**.

Tally Analysis

To confirm the theory that the port 515 connects are false positives generated by the use of a departmental print server we will perform a tally analysis. The goal is to show that port 515 connections a lower on weekends than on week days which would be consistent with printer usage in a university, except possibly in student labs.

For this exercise 24 days of alert logs were analyzed using this perl script:

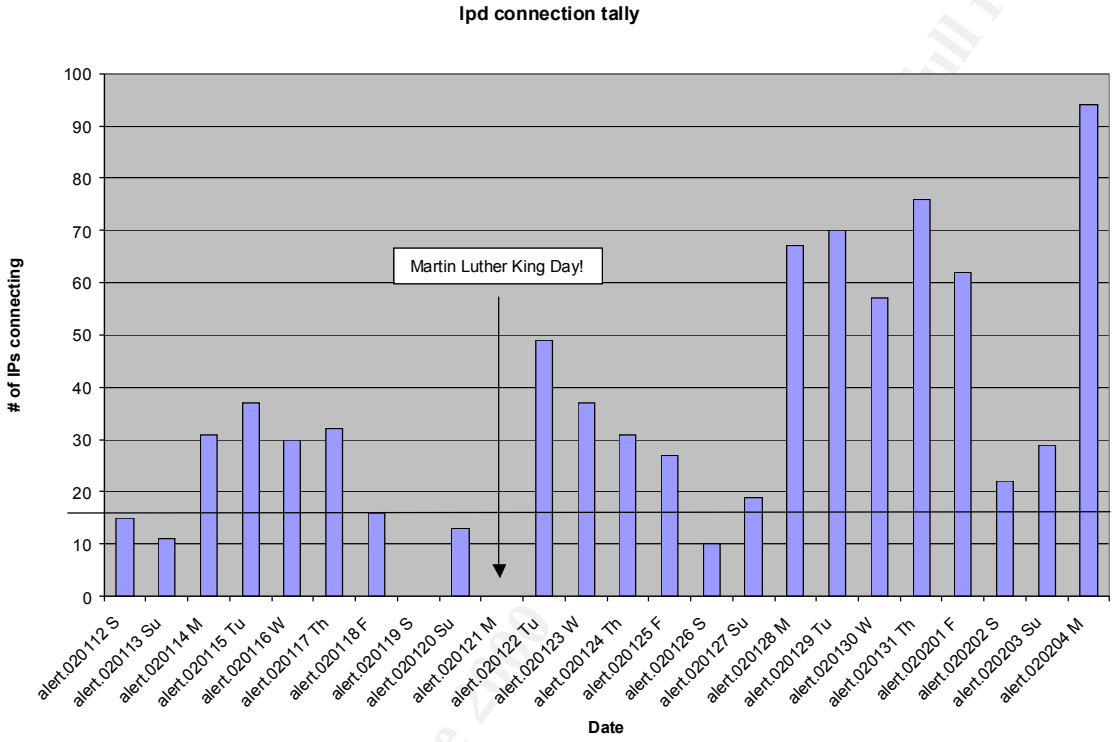
```
while (<>) {

        next unless /connect to 515 from inside/;

        /\s+(\S+)\s+->/;
        $combo = $1;
        ($ip, $port) = split(/:/, $combo);
        if (! $seen{$ip} ) {
                $seen{$ip} = 1;
                $hits++;
```

```
        }
}
print "$hits distinct hits\n";
```

Each unique IP address that connects to the print is counted once each day.



The graph above shows that lpd connections drop significantly during the weekends and on national holidays which is consistent with usage of a departmental print server.

Analysis: *Host MY.NET.70.177*

Using the Unix grep command to obtain a quick summary of the alerts generated by MY.NET.70.177 reveals:

```
# grep  "MY.NET\.70\.177.*->" alert.all | cut -b 23-53 | sort | uniq -c
     20  [**] SMB Name Wildcard [**] MY
  11787  [**] SNMP public access [**] M
```

which reveals that almost all alerts this host generates are the result of the following Snort rule:

```
alert udp any any - $HOME_NET 161 (msg: "SNMP public access";
content:"public";)
```

Description of Alert:

SNMP (Simple Network Management Protocol) is used to monitor and configure devices that have SNMP capabilities. The SNMP community string servers as a simple password to protect access to SNMP devices. The word 'public' is often set as the default SNMP community string. Attackers can get and set information on your devices by using this community string. For additional information on this alert see the following sources:

- http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0516
- http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0517
- http://www.sans.org/newlook/resources/IDFAQ/SNMP.htm

Defensive Recommendations:

Changing the SNMP community strings to something other than the common defaults of 'public' and 'private' will make it more difficult for hackers to exploit your SNMP services. However the community string may still be easily discovered by hackers with local access to your network. SNMP version 3 has the ability to transfer community string is an encrypted form and is recommend if your devices support it.

Correlation:

By inspection of **Table 2** we see that 18 host have generated all SNMP alerts which account for 6.81% of all alerts. Digging a little deeper:

```
# ./alert_stats.pl 'SNMP public' alert.all > SNMP
# grep "MY.NET.70.177" SNMP | cut -f 3 | cut -b 10- | perl -n -e '$i += $_;
END {print "$i\n";}'
11787
# grep -v "MY.NET.70.177" SNMP | cut -f 3 | cut -b 10- | perl -n -e '$i +=
$_; END {print "$i\n";}'
12028
# grep "MY.NET.70.177" SNMP
MY.NET.70.177   -> MY.NET.5.104  alerts: 15
MY.NET.70.177   -> MY.NET.5.105  alerts: 17
MY.NET.70.177   -> MY.NET.5.106  alerts: 16
MY.NET.70.177   -> MY.NET.5.204  alerts: 17
MY.NET.70.177   -> MY.NET.5.107  alerts: 15
MY.NET.70.177   -> MY.NET.5.108  alerts: 15
MY.NET.70.177   -> MY.NET.5.109  alerts: 15
MY.NET.70.177   -> MY.NET.5.141  alerts: 1029
MY.NET.70.177   -> MY.NET.5.127  alerts: 1848
MY.NET.70.177   -> MY.NET.5.37   alerts: 1717
MY.NET.70.177   -> MY.NET.5.128  alerts: 1890
MY.NET.70.177   -> MY.NET.5.238  alerts: 18
MY.NET.70.177   -> MY.NET.5.90   alerts: 16
MY.NET.70.177   -> MY.NET.5.83   alerts: 363
MY.NET.70.177   -> MY.NET.5.92   alerts: 688
MY.NET.70.177   -> MY.NET.5.249  alerts: 1413
MY.NET.70.177   -> MY.NET.5.85   alerts: 17
MY.NET.70.177   -> MY.NET.5.79   alerts: 463
```

```
MY.NET.70.177   -> MY.NET.5.95   alerts: 58
MY.NET.70.177   -> MY.NET.5.87   alerts: 18
MY.NET.70.177   -> MY.NET.5.96   alerts: 1863
MY.NET.70.177   -> MY.NET.5.97   alerts: 212
MY.NET.70.177   -> MY.NET.5.100  alerts: 14
MY.NET.70.177   -> MY.NET.5.101  alerts: 14
MY.NET.70.177   -> MY.NET.5.110  alerts: 20
MY.NET.70.177   -> MY.NET.5.103  alerts: 16
```

We find that MY.NET.70.177 generated half of all SNMP alerts and that all of those alerts are to systems on the MY.NET.5 network. This would indicate that MY.NET.70.177 is most likely an SNMP management station and not an attacker. For the listing above it is reasonable to suspect that the MY.NET.5 network is a server network. Alerts for machines on this network should be closely monitored.

Analysis: *Hosts MY.NET.11.6 and MY.NET.11.7*

Using the Unix grep command to obtain a quick summary of the alerts generated by MY.NET.11.6 reveals:

```
# grep  "MY.NET\.11\.6.*->" alert.all | cut -b 23-63 | sort | uniq -c
[**] SMB Name Wildcard [**] MY.NET.11.6:
# grep  "MY.NET\.11\.7.*->" alert.all | cut -b 23-53 | sort | uniq -c
[**] SMB Name Wildcard [**] MY
```

all alerts generated by the hosts are SMB Name Wildcard, and were generated by the following Snort rule:

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

Description of Alert:

The signature for this alert is designed to catch the use of wildcards in the Microsoft SMB protocol. This wildcard is normally used to enumerate network services on a Windows host. Hackers often use SMB wildcards as part of reconnaissance. The Windows command 'nbtstat –A <ip address>' can be used to enumerate services on the given IP address.

This alert is not critical if the source of the alert is a Windows system on your own network. If the source of the alert is a system outside of your network or from a non-Windows hosts then the source IP should be monitor for other hostile activity.

More information on SMB wildcards can be found at the SANS website.[3]

Defensive Recommendations:

Firewalls should be used to limit SMB traffic to only desired networks.

Correlation:

```
./alert_stats.pl 'SMB Name Wildcard' alert.all > SMB
grep 'MY.NET\.11\.6.*->' SMB
MY.NET.11.6     -> MY.NET.152.214        alerts: 200
MY.NET.11.6     -> MY.NET.152.215        alerts: 204
MY.NET.11.6     -> MY.NET.152.216        alerts: 98
      .
      .     Data Pruned
      .
MY.NET.11.6     -> MY.NET.152.173        alerts: 123
MY.NET.11.6     -> MY.NET.152.157        alerts: 48
MY.NET.11.6     -> MY.NET.152.174        alerts: 68
MY.NET.11.6     -> MY.NET.152.247        alerts: 194
MY.NET.11.6     -> MY.NET.152.158        alerts: 236
MY.NET.11.6     -> MY.NET.152.248        alerts: 8
MY.NET.11.6     -> MY.NET.152.159        alerts: 141
MY.NET.11.6     -> MY.NET.152.175        alerts: 170
MY.NET.11.6     -> MY.NET.152.249        alerts: 1
MY.NET.11.6     -> MY.NET.152.176        alerts: 38
MY.NET.11.6     -> MY.NET.152.177        alerts: 70
MY.NET.11.6     -> MY.NET.152.178        alerts: 101
MY.NET.11.6     -> MY.NET.152.179        alerts: 232
MY.NET.11.6     -> MY.NET.152.21         alerts: 174
MY.NET.11.6     -> MY.NET.152.22         alerts: 207
MY.NET.11.6     -> MY.NET.152.44         alerts: 72
MY.NET.11.6     -> MY.NET.152.213        alerts: 199
grep 'MY.NET\.11\.7.*->' SMB | more
MY.NET.11.7     -> MY.NET.152.214        alerts: 5
MY.NET.11.7     -> MY.NET.152.46         alerts: 207
MY.NET.11.7     -> MY.NET.152.215        alerts: 5
MY.NET.11.7     -> MY.NET.152.216        alerts: 128
MY.NET.11.7     -> MY.NET.152.160        alerts: 205
MY.NET.11.7     -> MY.NET.152.250        alerts: 93
MY.NET.11.7     -> MY.NET.152.161        alerts: 169
MY.NET.11.7     -> MY.NET.152.162        alerts: 196
MY.NET.11.7     -> MY.NET.152.251        alerts: 169
      .
      .     Data Pruned
      .
MY.NET.11.7     -> MY.NET.152.176        alerts: 200
MY.NET.11.7     -> MY.NET.152.177        alerts: 109
MY.NET.11.7     -> MY.NET.152.178        alerts: 23
MY.NET.11.7     -> MY.NET.152.179        alerts: 10
MY.NET.11.7     -> MY.NET.152.21         alerts: 4
MY.NET.11.7     -> MY.NET.152.22         alerts: 12
MY.NET.11.7     -> MY.NET.152.44         alerts: 143
MY.NET.11.7     -> MY.NET.152.213        alerts: 8
```

All SMB wildcard alerts are to hosts on the MY.NET.152 network. MY.NET.11.7 is probably the browse master for a windows network and hosts on network MY.NET.152 are members of the Windows domain. If this is not the case then these costs should be inspected for signs of intrusion.

⇒ At this point all hosts in Table 3 have had their alerts tentatively labeled as false positives. The analysis will continue by looking at alerts that occurred very few times or once. By doing this it is hoped that we will find attacks specifically targeted through prior reconnaissance.

Analysis: *BACKDOOR NetMetro File List*

By grepping alert.all we find that the target is MY.NET.5.96 and the potential attacker is 141.157.96.84.

```
# grep "BACKDOOR NetMetro File" alert.all
01/28-21:19:56.466715  [**] BACKDOOR NetMetro File List [**] MY.NET.5.96:80 -
> 141.157.96.84:5032
```

Snort alert file format:

The entry for the alert file above is an example of the Snort fast file format, which is composed of 4 elements:

1. The first field is the date a timestamp from when the packet was logged.
2. The second field contains the alert name between [**].
3. The third field is the source IP address and port separated by a colon.
4. The fourth field is the destination IP address and port separated by a colon.

```
<Date Stamp> [**] <Alert Name> [**] <Source_IP>:<Source_port> ->
<Destination_IP>:<Destination_Port>
```

Here is the Snort rule that generated the alert:

```
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;)
```

Snort rule options not covered previously in this document are:

- flags: Specifies what TCP flags must be set for the rule to match. In this case 'A+' means that the ACK flag must be set and any other flag may be set.
- reference: Allows rules to include references to external attack identification systems.
- classtype: Allows rules to include references to external attack identification systems.
- sid: Allows rules to include references to external attack identification systems.
- rev: The rev keyword is used to identify rule revisions.

This alert looks like a false positive for the following reasons:

- The target appears to be a web server and since the alert only happened once it is reasonable that content rules will occasionally be triggered by normal activity.

- The ephemeral port 5032 will be encountered every once in a while.

- MY.NET.5.96 doesn't generate many interesting alerts, but does appear to have an SNMP relationship with local address MY.NET.70.177.

```
# grep 'MY\.NET\.5\.96' alert.all | cut -b '23-' | sort | uniq -c
```

The following alert was generated 1,863 times:

```
[**] SNMP public access [**] MY.NET.70.177:1070 -> MY.NET.5.96:161
```

MY.NET.70.177 is a known SNMP management station.

Description of Alert:

NetMetro is a Trojan that runs on NT systems and commonly listens to port 5032.

Defensive recommendations (expand):

- Check to make sure MY.NET.5.96 is a web server.
- If it is a web server review for evidence of compromise and make sure current patches are installed.

Correlation:

MY.NET.70.177 is a known SNMP management station.

Analysis: *FTP EXPLOIT aix overflow*

There is only one occurrence of this alert in the log allowing simple inspection with grep.

```
# grep "FTP EXPLOIT aix overflow" alert.all
01/30-16:56:06.228940  [**] FTP EXPLOIT aix overflow [**]
137.142.181.128:2201 -> MY.NET.153.152:21
```

The alert was generated by the following Snort rule:

```
# grep "FTP EXPLOIT aix overflow" /etc/snort/*
/etc/snort/ftp.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP
EXPLOIT aix overflow";flags: A+;dsize:>1300; content:"CEL ";
reference:bugtraq,679; reference:cve,CVE-1999-0789; reference:arachnids,257;
classtype:attempted-admin; sid:337; rev:2;)
```

Snort rule options not covered previously in this document are:

- dsize: The dsize option is used to test the packet payload size.

Description of Alert:

'FTP EXPLOIT aix overflow', is an attempt to exploit a bug in the AIX FTP server. This bug allows a remote attacker to run arbitrary command with the privileges of the root user.

This alert is an FTP buffer overflow with MY.NET.152.152 as the target. Noting the date stamp in the alert above, it would be interesting to see if MY.NET.153.152 show signs of exploit, especially after the alert. Looking at all alerts with MY.NET.153.152 as the source:

```
# grep "MY.NET.153.152.* ->" alert.all | cut –b 23-53 | sort | uniq -c
    118  [**] connect to 515 from inside [**]
      4  [**] ICMP Echo Request Windows [**]
      1  [**] SMB Name Wildcard [**]
    258  [**] spp_http_decode: IIS Unicode attack detected [**]
```

(output slightly edit to show complete alert name)

The 118 'connect to 515 from inside' alerts have been previously attributed to use of a departmental print server. The 258 unicode alerts are curious. Inspecting the alert log directly:

```
# grep "MY.NET.153.152" alert.all | grep Unicode | more
01/28-12:16:52.496144  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1278 -> 211.32.117.37:80
01/28-12:16:52.496144  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1278 -> 211.32.117.37:80
01/28-12:16:52.496144  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1278 -> 211.32.117.37:80
01/28-12:16:52.496144  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1278 -> 211.32.117.37:80
01/28-12:16:52.496144  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1278 -> 211.32.117.37:80
      .
      .     Data Pruned
      .

01/28-12:16:58.055477  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1288 -> 211.32.117.37:80
01/28-12:16:58.055477  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1288 -> 211.32.117.37:80
01/28-12:16:58.055477  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1288 -> 211.32.117.37:80
01/28-12:16:58.055477  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1288 -> 211.32.117.37:80
01/28-12:16:58.055477  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1288 -> 211.32.117.37:80
01/28-12:16:58.055477  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1288 -> 211.32.117.37:80
```

```
01/28-12:16:58.055477  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1288 -> 211.32.117.37:80
01/28-12:16:58.055477  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1288 -> 211.32.117.37:80
      .
      .     Data Pruned
      .
01/28-12:17:45.304090  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1261 -> 211.233.29.210:80
01/28-12:17:45.304090  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1261 -> 211.233.29.210:80
01/28-12:17:45.304090  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1261 -> 211.233.29.210:80
01/28-12:17:45.470490  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1283 -> 211.233.29.210:80
01/28-12:17:45.470490  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1283 -> 211.233.29.210:80
01/28-12:17:45.470490  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1283 -> 211.233.29.210:80
      .
      .     Data Pruned
      .

01/29-17:05:33.225008  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1592 -> 61.222.204.138:80
01/29-17:07:29.643089  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1625 -> 61.222.204.138:80
01/29-17:07:29.643089  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1625 -> 61.222.204.138:80
01/29-17:07:29.643089  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1625 -> 61.222.204.138:80
01/29-17:07:29.643089  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1625 -> 61.222.204.138:80
01/29-17:16:32.564375  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1663 -> 61.222.204.138:80
01/29-17:19:16.320608  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1680 -> 61.222.204.138:80
01/29-17:19:18.165684  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1681 -> 61.222.204.138:80
01/29-17:19:40.433654  [**] spp_http_decode: IIS Unicode attack detected [**]
MY.NET.153.152:1685 -> 61.222.204.138:80
```

The problem at this point is to determine if these alerts represent real Unicode attacks or are false positives. It would make sense that a host that uses a departmental print server also is used for web browsing.

Checking MY.NET.153.152 for other signs of malicious activity:

```
# grep "MY\.NET\.153\.152.* ->" scans.all | wc -l
6395
```

Observing some of the scan logs directly led me to:

```
# grep "MY\.NET\.153\.152.* ->" scans.all | grep "53 UDP" | wc -l
   1596
```

```
# grep "MY\.NET\.153\.152.* ->" scans.all | grep "53 UDP" | cut -b 40- | sort
| uniq -c
    776 MY.NET.1.3:53 UDP
    783 MY.NET.1.4:53 UDP
     37 MY.NET.1.5:53 UDP
```

So 25% of the hosts alleged scans are all to what appear to be DNS server for MY.NET. Most likely the other portscan alerts are also false positives.

Correlation:

In the alerts above it is curious that many alerts to the same host all happen in the exact same timestamp. This is a probably a bug in the Snort http_decode preprocessor that causes multiple alerts for the same packet. A discussion about this bug can be found at Incidents.org[4] and Securepoint.com[5].

Further Analysis:

Focusing our analysis on the attacker in the alert:

```
# grep "137.142.181.128" ip_src.sorted
71 137.142.181.128
```

Visual inspection of the alert log show that all 71 alerts were ftp related, had MY.NET.153.152 as the target, and occurred in a 10 minute period.

Defensive recommendations:

- Review MY.NET.153.152 for signs of intrusion or use as a reconnaissance station.
- Review the need for MY.NET.153.152 to run an FTP and shut it down or make sure that it is appropriately patched.
- If it is a web server review for evidence of compromise and make sure current patches are installed.
- The host should be watched for signs of malicious activity in the future.

⇒ At this point the analysis will focus on hosts outside MY.NET generating large numbers of alerts (**Table 4**).

Analysis: *Hosts 63.250.209.34, 63.250.211.165, 63.250.210.50*

These 3 hosts are the 3 hosts outside of MY.NET that generate the most alerts (**Table 4**).

Registration Information:

All the addresses belong to Yahoo! Broadcast Services.

```
# whois 63.250.209.34@whois.arin.net
[whois.arin.net]
Yahoo! Broadcast Services, Inc. (NETBLK-NETBLK2-YAHOOBS)
   2914 Taylor st
   Dallas, TX 75226
   US

   Netname: NETBLK2-YAHOOBS
   Netblock: 63.250.192.0 - 63.250.223.255
   Maintainer: YAHO

   Coordinator:
      Bonin, Troy  (TB501-ARIN)  netops@broadcast.com
      214.782.4278 ext. 2278

   Domain System inverse mapping provided by:

   NS.BROADCAST.COM            206.190.32.2
   NS2.BROADCAST.COM           206.190.32.3

   ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

   Record last updated on 29-Jun-2001.
   Database last updated on  4-Feb-2002 19:57:43 EDT.

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.
```

Analysis of the alert log shows that all alerts generated by these 3 hosts are of the type, 'MISC Large UDP Packet':

```
# grep "63\.250.209\.34.*->" alert.all | cut -b 23-63 | sort | uniq -c
  10739  [**] MISC Large UDP Packet [**] 63.250.2
# grep "63\.250.211\.165.*->" alert.all | cut -b 23-63 | sort | uniq -c
   9859  [**] MISC Large UDP Packet [**] 63.250.2
# grep "63\.250.210\.50.*->" alert.all | cut -b 23-63 | sort | uniq -c
   7246  [**] MISC Large UDP Packet [**] 63.250.2
```

which is detected by the following Snort rule:

```
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Large UDP Packet";
dsize: >4000; reference:arachnids,247; classtype:bad-unknown; sid:521;
rev:1;)
```

Description of Alert:

The Snort rule is matched by all UDP packets originating outside of MY.NET with a destination of MY.NET having a packet payload size greater than 4,000 bytes. Traffic originating from Yahoo! Broadcast Services that matches this rule are associated with streaming audio and other multimedia applications and are not hostile.

Defensive Recommendations:

This traffic is not malicious, but the traffic generated by streaming media applications may impact network performance or be considered wasteful. Blocking this traffic at a border router or firewall may be appropriate. Also the IDS rules should be tuned so that false positives will not be generated by these hosts.

Correlation:

This type of traffic has been analyzed in another practical with similar conclusions[6].

Analysis: *Hosts 203.231.232.15*

This host is the 4th largest generator of alerts in **Table 4**.

Registration Information:

This address belongs to PSINet in Korea. PSINet is a worldwide Internet service provider. http://www.psi.net/network/index.html

```
# whois 203.231.232.15@whois.arin.net
[whois.arin.net]
Asia Pacific Network Information Center (APNIC2)
   These addresses have been further assigned to Asia-Pacific users.
   Contact info can be found in the APNIC database,
   at WHOIS.APNIC.NET or http://www.apnic.net/
   Please do not send spam complaints to APNIC.
   AU

   Netname: APNIC-CIDR-BLK
   Netblock: 202.0.0.0 - 203.255.255.255
   Maintainer: AP

   Coordinator:
      Administrator, System  (SA90-ARIN)  [No mailbox]
      +61-7-3367-0490

   Domain System inverse mapping provided by:

   SVC00.APNIC.NET              202.12.28.131
   NS.APNIC.NET                 203.37.255.97
   NS.TELSTRA.NET               203.50.0.137
```

```
   NS.RIPE.NET                 193.0.0.193

   Regional Internet Registry for the Asia-Pacific Region.

   *** Use whois -h whois.apnic.net [object]                    ***
   *** or see http://www.apnic.net/db/ for database assistance   ***


   Record last updated on 18-Jun-1999.
   Database last updated on  4-Feb-2002 19:57:43 EDT.

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.

# whois 203.231.232.15@whois.apnic.net
[whois.apnic.net]

% Rights restricted by copyright. See
http://www.apnic.net/db/dbcopyright.html
% (whois7.apnic.net)

inetnum:     203.226.0.0 - 203.231.255.255
netname:     KRNIC-KR
descr:       KRNIC
descr:       Korea Network Information Center
country:     KR
admin-c:     HM127-AP
tech-c:      HM127-AP
remarks:     ****************************************
remarks:     KRNIC is the National Internet Registry
remarks:     in Korea under APNIC. If you would like to
remarks:     find assignment information in detail
remarks:     please refer to the KRNIC Whois DB
remarks:     http://whois.nic.or.kr/english/index.html
remarks:     ****************************************
mnt-by:      APNIC-HM
mnt-lower:   MNT-KRNIC-AP
changed:     hostmaster@apnic.net 19981001
changed:     hostmaster@apnic.net 20010606
source:      APNIC

person:      Host Master
address:     Korea Network Information Center
address:     Narajongkeum B/D 14F, 1328-3, Seocho-dong, Seocho-ku, Seoul,
137-070, Republic of Korea
country:     KR
phone:       +82-2-2186-4500
fax-no:      +82-2-2186-4496
e-mail:      hostmaster@nic.or.kr
nic-hdl:     HM127-AP
mnt-by:      MNT-KRNIC-AP
changed:     hostmaster@nic.or.kr 20010514
source:      APNIC
```

```
# whois 203.231.232.15@whois.nic.or.kr
[whois.nic.or.kr]
Korea Internet Information Service V1.0 ( created by KRNIC, 2001.6 )

20013b 7?y 2@O:NEM4B 03<15H Whois <-:q=:8& @{?kGO0m @V=@4O4Y.

query: 203.231.232.15


.
. (Korean pruned)
.

If you did not get any query result of some IP address blocks,
they are the IP address blocks each responsible ISP didnot notice KRNIC of
its assignment or still held by KRNIC.
Regarding end-user contact of the IP address,
you should contact directly a responsible person in each ISP.


[ GX4g IPAV<R0! 9hA$5H KRNIC H8?x ISP1b0| A$:8 ]

 KOREAN

.
. (Korean pruned)
.

 ENGLISH

[ ISP member ORG information ]
Org Name      : PSINet Korea Inc.
Service Name  : PSINet
Org Address   : Seoul Inet Bldg, 738-37, Yoksam-dong, Kangnam-gu

[ Admin Contact Information ]
Name          : Changseung LEE
Phone         : 02-531-7700
Fax           : 02-555-8127
E-Mail        : mgr@kr.psi.net

[ IP Manager Contact Information ]
Name          : Soojeong LEE
Phone         : 02-531-7700
Fax           : 02-555-8127
E-mail        : ipadm@kr.psi.net

[ Hacking/SPAM Contact Information ]
Name          : ABUSE
Phone         : 02-531-7900
Fax           : 02-555-8127
E-mail        : abuse@kr.psi.net
```

Analysis of the alert log shows that all alerts generated by this host are of the type, 'MISC Large UDP Packet':

```
# grep "203.231.232.15.*->" alert.all | cut -b 23-63 | sort | uniq -c
   3109  [**] MISC Large UDP Packet [**] 203.231.
```

which is detected by the Snort rule described in the previous analysis.

```
# ./alert_stats.pl 'MISC Large UDP Packet' alert.all | more
207.25.79.240   -> MY.NET.88.181         alerts: 1
207.25.79.241   -> MY.NET.150.79         alerts: 1
211.233.27.142  -> MY.NET.153.160        alerts: 199
202.58.33.70    -> MY.NET.153.191        alerts: 840
63.250.211.165  -> MY.NET.151.63         alerts: 9859
63.250.209.88   -> MY.NET.151.63         alerts: 4
63.250.208.38   -> MY.NET.153.193        alerts: 27
63.250.209.34   -> MY.NET.153.210        alerts: 219
63.250.209.34   -> MY.NET.151.63         alerts: 10520
63.250.210.50   -> MY.NET.151.63         alerts: 7246
68.55.200.56    -> MY.NET.150.143        alerts: 4
203.231.232.15  -> MY.NET.153.195        alerts: 3109
63.250.211.197  -> MY.NET.153.210        alerts: 47
64.152.216.77   -> MY.NET.153.194        alerts: 677
```

Description of Alert:

The Snort rule is matched by all UDP packets originating outside of MY.NET with a destination of MY.NET having a packet payload size greater than 4,000 bytes. This might be a streaming media application similar to the previous analysis.

Defensive Recommendations:

This traffic is probably not malicious. The only destination host for these alerts was MY.NET.153.195 which is on a network that is frequently the destination of large UDP packets usually associated with Yahoo! Broadcast Services. Regardless host 203.231.232.15 should be watched for future hostile activity.

Correlation:

I have not found any sources of information about alerts generated by 203.231.232.15.

Analysis: *Hosts 202.179.35.8 and 202.179.35.8*

Both of these host are on the same network which has the following registration information:

Registration Information:

```
# whois 212.179.35.8@whois.arin.net
[whois.arin.net]
European Regional Internet Registry/RIPE NCC (NET-RIPE-NCC-)
   These addresses have been further assigned to European users.
   Contact info can be found in the RIPE database, via the
```

```
   WHOIS and TELNET servers at whois.ripe.net, and at
   http://www.ripe.net/perl/whois/
   NL

   Netname: RIPE-NCC-212
   Netblock: 212.0.0.0 - 212.255.255.255
   Maintainer: RIPE

   Coordinator:
      Reseaux IP European Network Co-ordination Centre Singel 258  (RIPE-NCC-
ARIN)  nicdb@RIPE.NET
      +31 20 535 4444

   Domain System inverse mapping provided by:

   NS.RIPE.NET                   193.0.0.193
   NS.EU.NET                     192.16.202.11
   AUTH03.NS.UU.NET              198.6.1.83
   NS2.NIC.FR                    192.93.0.4
   SUNIC.SUNET.SE                192.36.125.2
   MUNNARI.OZ.AU                 128.250.1.21
   NS.APNIC.NET                  203.37.255.97

   To search on arbitrary strings, see the Database page on
   the RIPE NCC website at http://www.ripe.net/perl/whois/

   Record last updated on 16-Oct-1998.
   Database last updated on  4-Feb-2002 19:57:43 EDT.

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.

# whois 212.179.35.8@whois.ripe.net
[whois.ripe.net]
% 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
```

A quick analysis of the alert logs shows that these hosts, along with other hosts on the 212.179 network are responsible for all 'Watchlist 000220 IL-ISDNNET-990517' alerts.

```
# grep "212.179.35.*->" alert.all | cut -b 23-63 |sort | uniq -c | more
   2561  [**] Watchlist 000220 IL-ISDNNET-990517
# grep "212.179\..*->" alert.all | cut -b 23-63 | sort | uniq -c | more
[**] Watchlist 000220 IL-ISDNNET-990517
```

Additional analysis shows that all hosts targeted from this network are on these networks:

- MY.NET.88
- MY.NET.150
- MY.NET.151
- MY.NET.153

```
# ./alert_stats.pl 'IL-ISDNNET-990517' alert.all > ISDNNET
# cat ISDNNET
212.179.5.226   -> MY.NET.150.41        alerts: 5
212.179.43.225  -> MY.NET.88.162        alerts: 12
212.179.48.2    -> MY.NET.88.162        alerts: 11
212.179.48.2    -> MY.NET.150.220       alerts: 1
212.179.48.2    -> MY.NET.150.133       alerts: 5
212.179.56.5    -> MY.NET.150.41        alerts: 4
212.179.33.169  -> MY.NET.150.41        alerts: 12
212.179.71.214  -> MY.NET.88.162        alerts: 32
212.179.127.37  -> MY.NET.150.133       alerts: 2
212.179.127.53  -> MY.NET.150.41        alerts: 13
212.179.127.54  -> MY.NET.150.41        alerts: 16
212.179.125.79  -> MY.NET.150.41        alerts: 50
212.179.125.79  -> MY.NET.88.162        alerts: 6
212.179.127.75  -> MY.NET.88.162        alerts: 1
212.179.27.176  -> MY.NET.153.178       alerts: 96
212.179.27.176  -> MY.NET.153.148       alerts: 64
212.179.27.176  -> MY.NET.152.161       alerts: 60
212.179.27.176  -> MY.NET.153.46        alerts: 131
212.179.28.66   -> MY.NET.150.220       alerts: 4
212.179.45.196  -> MY.NET.150.41        alerts: 15
212.179.8.194   -> MY.NET.88.162        alerts: 4
212.179.19.2    -> MY.NET.88.162        alerts: 12
212.179.19.2    -> MY.NET.150.41        alerts: 12
212.179.27.6    -> MY.NET.150.41        alerts: 78
212.179.27.6    -> MY.NET.88.162        alerts: 1
212.179.27.6    -> MY.NET.150.220       alerts: 6
212.179.27.6    -> MY.NET.150.133       alerts: 25
212.179.35.8    -> MY.NET.151.85        alerts: 1419
```

```
212.179.47.87   -> MY.NET.150.133        alerts: 4
212.179.76.28   -> MY.NET.88.162         alerts: 4
212.179.127.3   -> MY.NET.88.162         alerts: 2
212.179.77.107  -> MY.NET.150.41         alerts: 2
212.179.35.118  -> MY.NET.153.178        alerts: 12
212.179.35.118  -> MY.NET.153.148        alerts: 14
212.179.35.118  -> MY.NET.88.162         alerts: 1078
212.179.49.2    -> MY.NET.150.41         alerts: 3
212.179.35.119  -> MY.NET.150.41         alerts: 2
212.179.35.119  -> MY.NET.88.162         alerts: 1
212.179.37.10   -> MY.NET.150.133        alerts: 3
212.179.127.100 -> MY.NET.88.162         alerts: 12
212.179.15.94   -> MY.NET.150.133        alerts: 12
212.179.112.100 -> MY.NET.153.203        alerts: 22
212.179.43.72   -> MY.NET.150.133        alerts: 4
212.179.127.65  -> MY.NET.150.220        alerts: 1
212.179.34.194  -> MY.NET.88.162         alerts: 3
212.179.2.220   -> MY.NET.88.162         alerts: 3
212.179.40.132  -> MY.NET.150.41         alerts: 2
212.179.40.132  -> MY.NET.88.162         alerts: 13
212.179.125.254 -> MY.NET.150.220        alerts: 2
212.179.125.254 -> MY.NET.150.133        alerts: 20
212.179.126.162 -> MY.NET.88.162         alerts: 11
212.179.45.74   -> MY.NET.88.162         alerts: 4
212.179.30.27   -> MY.NET.150.133        alerts: 3
212.179.38.226  -> MY.NET.150.133        alerts: 1
212.179.44.99   -> MY.NET.150.133        alerts: 3
212.179.29.181  -> MY.NET.88.162         alerts: 1
212.179.33.250  -> MY.NET.150.220        alerts: 3
212.179.41.246  -> MY.NET.150.41         alerts: 3
212.179.35.121  -> MY.NET.153.178        alerts: 20
212.179.35.121  -> MY.NET.153.148        alerts: 15
212.179.126.3   -> MY.NET.88.162         alerts: 9
212.179.126.3   -> MY.NET.150.41         alerts: 15
212.179.126.3   -> MY.NET.150.133        alerts: 1
212.179.95.11   -> MY.NET.150.133        alerts: 4
```

The Snort rule that matches these packets probably look like:

```
alert ip 212.179.0.0/16 -> $HOME_NET any (msg:" Watchlist 000220 IL-ISDNNET-
990517")
```

Description of Alert:

The Snort rule is matches all IP packets coming from the 212.179 network. This is a network in Israel widely know to be a source of hacking activity. All packets coming from this network should be analyzed.

Defensive Recommendations:

If possible, all traffic to or from the 212.179 network should be considered for being blocked at a border router or firewall. Also the Snort rules should be tuned to alert on more specific IDS signatures to simplify analysis of alerts from these hosts.

Further Analysis:

All 1,419 alerts to host MY.NET.151.85 are from an ephemeral port on MY.NET.151.85 to port 80 on 212.179.35.8, which may indicated that this is benign web traffic.

All 1,078 alerts to host MY.NET.88.162 are from an ephemeral port on MY.NET.88.162 to port 1214 on 212.179.35.118, which is a port often associated with the Kazaa peer-to-peer file sharing service.

Correlation:

Similar activity has been noted is the following documents:

- http://www.giac.org/practical/Garreth_jeremiah_GCIA.zip
- http://www.giac.org/practical/Thomas_Rodriguez_GCIA.doc
- http://www.giac.org/practical/Jeff_Holland_GCIA.doc

Analysis: *Network 64.162*

Two hosts from the 64.152 network are on the top ten alerts, **Table 4** and in the top ten scanners, **Table 9**.

Registration Information:

```
# whois 64.152.216.77@whois.arin.net
[whois.arin.net]
Level 3 Communications, Inc. (NETBLK-LC-ORG-ARIN)
   1025 Eldorado Boulevard
   Broomfield, CO 80021
   US

   Netname: LC-ORG-ARIN
   Netblock: 64.152.0.0 - 64.159.255.255
   Maintainer: LVLT

   Coordinator:
```

```
      level Communications  (LC-ORG-ARIN)  ipaddressing@level3.com
      +1 (877) 453-8353

   Domain System inverse mapping provided by:

   NS1.LEVEL3.NET               209.244.0.1
   NS2.LEVEL3.NET               209.244.0.2

   ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

   Record last updated on 30-May-2001.
   Database last updated on  4-Feb-2002 19:57:43 EDT.

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.
```

From the registration information we see that host 64.161.36.66 which ranks 10th in **Table 4** is also par of this net block. Turning our attention to the address with the greatest number of port scans, 64.152.108.141 we find that this system only generated 322 alerts:

```
# grep 64.152.108.141 ip_src.sorted
322     64.152.108.141
```

All connections from 64.152.108.141 are made to MY.NET.88.163.

```
# grep "64.152.108.141.*->" alert.all | perl -n -e '/(MY\.NET\.\d+\.\d+):/;
print "$1\n";' | sort | uniq -c
    322 MY.NET.88.163
```

Summarizing the alerts between these two hosts:

```
# grep "64.152.108.141.*->" alert.all | cut -b 23-63 | sort | uniq -c
      1  [**] Attempted Sun RPC high port access
     36  [**] EXPLOIT NTPDX buffer overflow [**]
      1  [**] EXPLOIT x86 NOOP [**] 64.152.108.14
    278  [**] High port 65535 udp - possible Red
      2  [**] RPC udp traffic contains bin sh [**
4  [**] TFTP - External UDP connection to I
```

The most frequently occurring alert is 'High port 65535 udp - possible Red Worm – traffic'.

Description of Alert:

This alert is generate by the following Snort rule:

```
alert udp any any <> any 65535 (msg:"High port 65535 udp - possible Red Worm
- traffic";)
```

This rule alerts on any traffic with a destination port of 65535, making occasional false positives likely. Port 65535 is the port used for the backdoor of the Red Worm, also called Adore. More information about this worm can be found here: http://www.sans.org/y2k/adore.htm. The frequency of communication with host MY.NET.88.163 on port 65535 make it unlikely that this is a false positive.

Defensive Recommendations:

Any host frequently communicating on port 65535 should be inspected for the Adore worm and cleaned. It may be possible to block connections to port 65535 on MY.NET at a border router or firewall. Only Linux systems are susceptible to attacks by the Adore worm so they should be monitored more closely for this type of traffic.

Correlation:

Discussion about malicious traffic from Level 3 netblocks can be found at Neohapsis.com[7]. No exactly matching IP addresses were mentioned.

OOS Analysis

Out of Spec was only generated for the first 3 days being analyzed and only 8 packets were captured making it impossible to include them all here.

```
01/26-12:28:28.903591 145.236.140.74:1214 -> MY.NET.150.133:1214
TCP TTL:49 TOS:0x0 ID:21505  DF
21S***** Seq: 0xEE04156C   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 2572653 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/26-12:31:26.534501 145.236.140.74:1251 -> MY.NET.150.133:1214
TCP TTL:49 TOS:0x0 ID:62090  DF
21S***** Seq: 0xF82E745A   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 2590416 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/26-12:39:41.355563 145.236.140.74:1361 -> MY.NET.150.133:1214
TCP TTL:49 TOS:0x0 ID:32438  DF
21S***** Seq: 0x189636AC   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 2639898 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/26-12:43:38.677969 145.236.140.74:1417 -> MY.NET.150.133:1214
TCP TTL:49 TOS:0x0 ID:63693  DF
21S***** Seq: 0x27AA5C02   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 2663631 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/27-06:25:02.953126 65.129.33.127:18245 -> MY.NET.5.96:21536
TCP TTL:21 TOS:0x0 ID:18458  DF
2*SFRP*U Seq: 0x2F62696E   Ack: 0x2F636F6D   Win: 0x6E2F
```

```
2E 70 6C 20 48 54 54 50 2F 31                       .pl HTTP/1

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/27-06:25:06.806710 65.129.33.127:18245 -> MY.NET.5.96:21536
TCP TTL:21 TOS:0x0 ID:20506  DF
2*SFRP*U Seq: 0x2F62696E   Ack: 0x2F636F6D   Win: 0x6E2F
2E 70 6C 3F 62 62 61 74 74 3D                       .pl?bbatt=

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/28-20:46:33.703718 64.166.209.137 -> MY.NET.88.162
TCP TTL:110 TOS:0x0 ID:33339  DF MF
Frag Offset: 0x0   Frag Size: 0x22
5A 1D 6B 5E 5B 1D 6C 99 22 37 5C 74 DD D3 2A 0C  Z.k^[.l."7\t..*.
C6 7A 15 8E E0 DC 01 2D 3E D6 87 7A D4 83 DF 32  .z.....->..z...2
3D B0                                            =.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/28-20:46:33.815778 64.166.209.137 -> MY.NET.88.162
TCP TTL:110 TOS:0x0 ID:33595  DF MF
Frag Offset: 0x0   Frag Size: 0x22
5B DC 9B FC 60 DE C0 BA E9 C4 23 F9 DA E5 95 D9  [...`.....#.....
E5 9A C7 D1 02 A6 EA 8D E4 6F 39 A3 53 B2 EB 18  .........o9.S...
75 57                                            uW

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

Hosts 65.129.33.127 and 64.166.209.137 did some minor port scanning but generated no other alerts.

Host 145.236.140.74 was the most frequent source of out of spec packets. The following alerts were generated by this address:

```
# grep "145\.236\..*->" alert.all
01/26-12:28:24.960670  [**] Queso fingerprint [**] 145.236.140.74:1214 ->
MY.NET.150.133:1214
01/26-12:31:22.573004  [**] Queso fingerprint [**] 145.236.140.74:1251 ->
MY.NET.150.133:1214
01/26-12:39:37.342370  [**] Queso fingerprint [**] 145.236.140.74:1361 ->
MY.NET.150.133:1214
01/26-12:43:34.640037  [**] Queso fingerprint [**] 145.236.140.74:1417 ->
MY.NET.150.133:1214
01/27-09:59:08.283284  [**] SCAN Synscan Portscan ID 19104 [**]
145.236.131.46:4603 -> MY.NET.150.133:1214
```

Registration Information:

```
# whois 145.236.140.74@whois.arin.net
[whois.arin.net]
European Regional Internet Registry/RIPE NCC (NETBLK-145-RIPE) RIPE-NCC-145
                                                 145.224.0.0 -
145.254.255.255
Hungarian Telecom (NET-HTC-NET) HTC-NET          145.236.0.0 -
145.236.255.255

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.

# whois 145.236.140.74@whois.ripe.net
[whois.ripe.net]
% 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:      145.236.0.0 - 145.236.255.255
netname:      MATAV
descr:        Hungarian Telecommunications Company Limited
descr:        Budapest
country:      HU
admin-c:      TS2796-RIPE
tech-c:       BAT3-RIPE
tech-c:       IC27-RIPE
rev-srv:      ns0.matav.net
rev-srv:      ns1.matav.net
rev-srv:      ns.elender.hu
status:       ASSIGNED PA
mnt-by:       AS5483-MNT
changed:      horvath@sztaki.hu 19950914
changed:      irina@mail.matav.hu 19980311
changed:      csaky@matav.net 20010123
source:       RIPE

route:        145.236.0.0/16
descr:        Hungarian Telecom
descr:        Public Internet Access Provider
descr:        Budapest, Hungary
descr:        HU
origin:       AS5483
mnt-by:       AS5483-MNT
changed:      csaky@matav.net 20010123
source:       RIPE
```

The registration information for this host indicates that it's IP address is managed by Hungarian Telecommunications Company Limited.

Description of Alert:

This alert is generate by the following Snort rule:

```
alert tcp any any -> 192.168.1.0/24 any (msg:"Queso fingerprint";flags: S12;)
```

Queso is an OS fingerprinting tool. It use unusual flag combinations and reserved bits of the TCP header to illicit unique response from the host it is fingerprinting. More information about Queso and OS fingerprinting can be found at these locations:

- http://www.securityfocus.com/columnists/57
- http://ftp.cerias.purdue.edu/pub/tools/unix/scanners/queso/
- http://www.insecure.org/nmap/nmap-fingerprinting-article.html

Defensive Recommendations:

Some routers and firewalls can be used to block TCP packets that have reserved bits set. When possible these packets should be stopped at perimeter security devices. More information about evading OS fingerprinting can be found at: http://www.linuxjournal.com/article.php?sid=4750 .

Correlation:

No correlation with any other reports could be found.

Analysis Process:

Each set of files; alert, scans, and oos were concatenated together into three large files; alert.all, scans.all, and oos.all. For all but the tally analysis, the concatenated logs were analyzed because it was not felt that time boundaries between alerts were meaningful.

The program SnortSnarf was evaluated for the data analysis task, but it was decided that it would require more computing resources than were available. Instead a few small perl scripts were developed to summarize the data and then Microsoft Excel was used to organize the data and generate some statistics.

The following perl script was written to generate the data for Table 2, and the output was augmented using Microsoft Excel.

```
#!/usr/bin/perl -w

while(<>) {

        next unless /\[\*\*\]/;
        if (/PORTSCAN DETECTED/) {
                $alert{" PORTSCAN DETECTED"}{count}++;
                $alert{" PORTSCAN DETECTED"}{src} = 'NA';
                $alert{" PORTSCAN DETECTED"}{dst} = 'NA';
                next;
        }
        /spp_portscan/ && next;
        /\[\*\*\](.*)\[\*\*\]/ || die "Improper format!\n";
        $type = $1;
        $alert{$type}{count}++;
```

```
        /\[\*\*\].*\[\*\*\]\s+(.*\..*\..*\..*)\s->\s(.*\..*\..*\..*)/ || die
"pattern not found";
        ($src_ip, $src_port) =  split(/:/, $1);
        ($dst_ip, $dst_port) =  split(/:/, $2);
        $alert{$type}{src}{$src_ip}++;
        $alert{$type}{dst}{$dst_ip}++;
}

foreach (keys %alert) {
        $num_src = scalar(keys %{$alert{$_}{src}});
        $num_dst = scalar(keys %{$alert{$_}{dst}});
        print $alert{$_}{count}, "\t $num_src\t$num_dst\t$_\n";
}
```

Data for Table 3, Table 4, Table 5, and Table 6 was generated with the following perl script and augmented with Microsoft Excel.

```
#!/usr/bin/perl -w

while(<>) {

        next unless /\[\*\*\]/;
        if (/PORTSCAN DETECTED/) {
                next;
        }
        /spp_portscan/ && next;
        /\[\*\*\].*\[\*\*\]\s+(.*\..*\..*\..*)\s->\s(.*\..*\..*\..*)/ || die
"pattern not found";
        ($src_ip, $src_port) =  split(/:/, $1);
        ($dst_ip, $dst_port) =  split(/:/, $2);
        $src{$src_ip}++;
        $dst{$dst_ip}++;
}

print "Source IP:\n\n";
foreach (keys %src) {
        print "$src{$_}\t$_\n";
}
print "\nDestination IP:\n\n";
foreach (keys %dst) {
        print "$dst{$_}\t$_\n";
}
```

Data for Table 7 was generate with the following perl script and augmented with Microsoft Excel.

```
#!/usr/bin/perl -w

while(<>) {

        next unless /\[\*\*\]/;
        if (/PORTSCAN DETECTED/) {
                next;
        }
        /spp_portscan/ && next;
```

```
        /\s+(\S+)\s->\s(\S+)/ || die "pattern not found";
        #/\[\*\*\].*\[\*\*\]\s+(.*\..*\..*\..*)\s->\s(.*\..*\..*\..*)/ || die
"pattern not found";
        ($src_ip, $src_port) =  split(/:/, $1);
        ($dst_ip, $dst_port) =  split(/:/, $2);
        unless (defined($dst_port)) { $dst_port = -1; }
        $dst{$dst_port}++;
        $dst_port = undef;
}

print "\nDestination Ports:\n\n";
foreach (keys %dst) {
        print "$dst{$_}\t$_\n";
}
```

The summary of destination ports (Table 7) was generated by the following perl script and augmented with Excel.

```
#!/usr/bin/perl -w

while(<>) {

        next unless /\[\*\*\]/;
        if (/PORTSCAN DETECTED/) {
                next;
        }
        /spp_portscan/ && next;
        /\s+(\S+)\s->\s(\S+)/ || die "pattern not found";
        #/\[\*\*\].*\[\*\*\]\s+(.*\..*\..*\..*)\s->\s(.*\..*\..*\..*)/ || die
"pattern not found";
        ($src_ip, $src_port) =  split(/:/, $1);
        ($dst_ip, $dst_port) =  split(/:/, $2);
        unless (defined($dst_port)) { $dst_port = -1; }
        $dst{$dst_port}++;
        $dst_port = undef;
}

print "\nDestination Ports:\n\n";
foreach (keys %dst) {
        print "$dst{$_}\t$_\n";
}
```

The port scanning summaries in Table 8 and Table 9 were produced by the following perl script and augmented with Excel.

```
#!/usr/bin/perl -w

while(<>) {

        /\s+(\S+)\s->\s(\S+)/ || die "pattern not found";
        ($src_ip, $src_port) =  split(/:/, $1);
        ($dst_ip, $dst_port) =  split(/:/, $2);
        $scan_src{$src_ip}++;
}
```

```
foreach (keys %scan_src) {
        print "$scan_src{$_}\t$_\n";
}
```

Using the output of these scripts, analysis subjects were identified and prioritized (Table 10). For each analysis subject, the *alert_stats.pl* script was used to generate more detailed statistics and sometimes this output was saved in a file for further processing by Unix command line tools, *grep*, *cut*, *sort*, and, *uniq, along with some one line perl programs*.

alert_stats.pl

```
#!/usr/bin/perl -w

$string = shift;

while(<>) {

        next unless /$string/;
        /\s(\S+) -> (\S+)\s/ || die "pattern not found";
        ($src_ip, $src_port) =  split(/:/, $1);
        ($dst_ip, $dst_port) =  split(/:/, $2);
        $alert{$src_ip}{$dst_ip}++;
        $src_ports{$src_port}++;
        #print "$src_ip -> $dst_ip\n";
        #print "$src_ip -> $dst_ip: $alert{$src_ip}{$dst_ip}\n";
}

foreach $src (keys %alert) {
        foreach $dst (keys %{$alert{$src}}) {
                print "$src\t-> $dst\t alerts: ", $alert{$src}{$dst}, "\n";
        }
}
```

References:

[1] MSNBC, “Yahoo! How did it happen?” February 7, 2000.
URL: http://zdnet.com.com/2100-11-518380.html?legacy=zdnn

[2] Lemos, Robert. “Hackers Cripple Whitehouse Site.” May 4, 2001.
URL: http://news.com.com/2100-1001-257068.html

[3] Alexander, Bryce. “Port 137 Scan“, May 10, 2000.
URL: http://www.sans.org/newlook/resources/IDFAQ/port_137.htm

[4] Sgtphou, “IIS Unicode attack detected : 66.90.148.161 / 66-90-148-161.grandecom.net.” June 13, 2001.
URL: http://www.incidents.org/archives/intrusions/msg00781.html

[5] Bollinger, Troy A. “Re: Fw: Remote buffer overflow exploit for ftpd from AIX 4.3.2 running on an RS6000. (power).”
URL: http://msgs.securepoint.com/cgi-bin/get/bugtraq9909/277/3.html

[6] Rodriguez, Robert. “Intrusion Detection In Depth.”
URL: http://www.giac.org/practical/Thomas_Rodriguez_GCIA.doc

[7] Bob@bangbang.org. “Strange ping activity.” May 22, 2001.
URL: http://archives.neohapsis.com/archives/snort/2001-05/0530.html

[8] CERT Coordination Center. “Denial of Service Attacks.” June 4, 2001.
URL: http://www.cert.org/tech_tips/denial_of_service.html

[9] ISS X-Force. “telnetd-login-bypass (4225).” March 13, 2000.
URL: http://www.iss.net/security_center/static/4225.php

[10] Security Focus. “Microsoft W2K Telnet Various Domain User Account Access Vulnerability.” June 8, 2001.
URL: http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=2847

Research Sources:

http://cve.mitre.org/

http://www.cert.org/

http://www.sans.org/

http://www.giac.org/

http://www.securityfocus.com/

http://www.snort.org/

http://www.incidents.org/

http://www.iss.net/

http://www.cultdeadcom.com/