Global Information Assurance Certification Paper

*** Northcutt, I only worked a "half day" today, 0830 - 2030 and all of it grading practicals. I am going to stop right now so I continue to feel like it is all worthwhile. Thank you Ron and you are indeed an analyst. 93 *

SANS GIAC

Intrusion Detection Certification

Ron Ryan
April 24, 2000.

Detect #1

Overview:

This is a detect I will almost always find in my logs for a typical day on my home Computer. I am using Signal9's personal firewall, ConSeal. Connectivity is provided by cable modem.

Trace #1 – ConSeal Personal Firewall

```
2000/04/01 12:23:38 PM GMT-0700: 3Com EtherLink 10..[0000][No matching rule] Blocking incoming TCP:
src=my.isp.94.130, dst=my.net.12.67, sport=47404, dport=119.
2000/04/01 12:23:40 PM GMT-0700: 3Com EtherLink 10..[0000][No matching rule] Blocking incoming TCP:
src=my.isp.94.130, dst=my.net.12.67, sport=47404, dport=119.

2000/04/01 12:23:40 PM GMT-0700: 3Com EtherLink 10..[0000][No matching rule] Blocking incoming TCP:
src=my.isp.94.130, dst=my.net.12.67, sport=48304, dport=119.
2000/04/01 12:23:41 PM GMT-0700: 3Com EtherLink 10..[0000][No matching rule] Blocking incoming TCP:
src=my.isp.94.130, dst=my.net.12.67, sport=48304, dport=119.

2000/04/01 12:23:41 PM GMT-0700: 3Com EtherLink 10..[0000][No matching rule] Blocking incoming TCP:
src=my.isp.94.130, dst=my.net.12.67, sport=49205, dport=119.
2000/04/01 12:23:43 PM GMT-0700: 3Com EtherLink 10..[0000][No matching rule] Blocking incoming TCP:
src=my.isp.94.130, dst=my.net.12.67, sport=49205, dport=119.
```

Analysis:

This is a port scan by my ISP. The 6 TCP packets, which are blocked by my firewall, are 3 scans for NNNTP News services (port 119). Each scan attempt is repeated twice as evidenced by the Source port used. In the first attempt a source port of 47404 is used along with a single retry two seconds later. In the second attempt source port 48304 is used, and finally source port 49205 is used for the final two scans. These scans occur daily and at random times. The ISP is searching for unauthorized News servers. The following is my ISP's response to a query on the purpose for the scan. I removed the actual location information as I am giving away too much about my configuration as it is:

> "The port scan your firewall software picked up on IP my.isp.94.130 is coming
> from 'myisp' operations in "guess where". They are monitoring the network looking
> for specific hosts.
> This is being done to maintain network integrity with regard to the proper
> use of the network. "

This is not a very good answer. It lacks any sort of technical detail and is not even a good canned response. Further queries by myself were rebuffed politely. They basically said that the information was proprietary and could be revealing.

Active Targeting: Yes, but the intent is benign. This is a cost of doing business with this ISP.

History: Definitely. This occurs daily at seemingly random time intervals.

Technique: This is a fairly slow scan of NNTP ports using a script or tool. They are scanning their entire address space and do it at a leisurely pace, probably to avoid any network congestion and perhaps because they are waiting for a response. I find it interesting that they pair attempts and do three retries.
These are most likely active open scans (SYN scans).

Intent: This is information gathering by the ISP. They are making sure that unauthorized News servers are not in place.

Severity/Criticality: 2 It's just me, on my PC.

Severity/Lethality: 1 This is information gathering only and there are no signs of any type of follow-up attempts at connecting to port 119. I don't have a news service running.

System Countermeasures: This system is Windows 98 a virtual paragon of security. File and print sharing are turned off. It has been scanned and somewhat hardened. I use first strike browsing protection, Anti-Virus Software, Back-Officer, and a personal firewall to beef up my security. I rate this as a **3**.

Network Countermeasures: I rate this as a **3.** My firewall is working and it is blocking incoming connection attempts that I don't initiate.

Severity = (2+1) – (3+3) = **-4 Non Hostile Activity**

Detect #2

Overview:
This is another log from my ConSeal personal firewall. I had to include at least one of this type of firewall log. It is just one example of the many trojan scans that occur on my home system almost every day. I have also seen BOping sweeps and netbus scans.

If you have cable modem or DSL, make sure you have some sort of personal firewall in place. I knew that bad stuff was happening on the net but I always said, "Why would anyone bother with me?" I can't stress this enough, get a firewall or IDS, it's an eye opener!

Trace #2 ConSeal Personal Firewall log:

```
2000/04/01 12:41:57 PM GMT-0700: 3Com EtherLink 10..[0000][No matching rule] Blocking incoming UDP:
src=24.19.135.26, dst=my.net.12.67, sport=31790, dport=31789.
```

Analysis:

In this particular case the bad guy is searching for UDP port 31789; the Hack´a´Tack Remote Access Trojan. This is most likely part of a much larger scan for hack'a'Tack probing the entire address space that my ISP provides. It may also be part of a smaller scan of particular sub-nets within that class B address space. I would love to have a look at the Intrusion Detect logs for my ISP, assuming they have one.

"UDP traffic on this port is currently being seen due to the "Hack-a-tack" RAT (Remote Access Trojan). This trojan includes a built-in scanner that scans from port 31790, so any packets FROM 31789 TO 31790 indicate a possible intrusion. (Port 31789 is the control connection; port 31790 is the file transfer connection)."

Ref: http://www.robertgraham.com/pubs/firewall-seen.html

Technical Information:

Hack 'a' Tack currently affects Windows 95/98 PC's. The server portion is named "expl32.exe" (236KB 5/16/99 2:49PM) and it will be found in the WINDOWS directory. TCP ports 31785, 31787 and UDP ports 31789, 31790 (by default) are used to establish the connection between the "client" and "server".
Once installed, it is rerun every time the computer is started by means of an entry under the **"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"** branch in the Registry.

This is what the GUI looks like for Hack'a'Tack:

Ref: http://www.commodon.com/threat/threat-hack.htm

The first step in any trojan hack is for the trojan to be installed on a system. This is usually accomplished by the unwitting recipient of the trojan opening a mail message with a "trojanized" attachment, such as an executable. The crackers can use programs called binders to combine trojans with seemingly innocuous files. "Binders are programs that allow hackers to "bind" two or more executables together resulting in a single .EXE file. These are useful tools as they easily allow a hacker to insert Trojan executables into harmless .EXE animations, e-greetings and other EXEs that are commonly passed around as e-mail attachments. "

The challenge for the cracker is finding a target. He does not necessarily know, in fact he has no idea, who has the trojan installed. This leads to regularly seen probes directed at machines from crackers looking for compromised hosts. However, if the machine hasn't been compromised, then these probes are not a problem. The probes themselves cannot compromise the machine.

Once compromised the attacker can perform the following functions:
FTP Transmit
IP: IP-Scanner
General Information i.e. Current User, Country, Time, OS and CPU.
Send Messages: Open/Close the CDROM
Hide/Show the taskbar Disable/enable the monitor
Disable keys Swap and click mouse buttons
Set/freeze the cursor at a position you can adjust by coordinates.
Window Events allowing you to kill, focus, hide, show and rename a process.

You can also see what the remote computer has in its clipboard and send text to the actually focused window.
Boot Operations i.e. shut down, reboot, poweroff and logoff the remote computer here.
Get Passwords
Keyspy Filemanager
Make Screenshot.

Active targeting: Yes. This individual is looking for the Hack'a'Tack Trojan on hosts in my ISP's address space. I would find it hard to believe that I am the only one that was targeted.

Source Address: 24.19.135.26

```
@Home Network (NETBLK-ATHOME)
  450 Broadway Street
  Redwood City, CA 94063
  US
  Netname: ATHOME
  Netblock: 24.0.0.0 - 24.23.255.255
  Maintainer: HOME
  Coordinator:
    Operations, Network  (HOME-NOC-ARIN)  noc@NOC.HOME.NET
    1-800-872-3595
  Domain System inverse mapping provided by:
  NS1.HOME.NET   24.0.0.27
  NS2.HOME.NET   24.2.0.27

  Record last updated on 20-Mar-2000.
  Database last updated on 7-Apr-2000 17:45:42 EDT.
```

History: This is the first time I have seen this particular source address. However, these types of probes occur regularly on my system and I take them seriously. I've added this one to a file of hack attempts on my system.

Technique: I can't say a lot about the technique. I am sure that the individual involved is using Hack'a'Tack itself to scan for the trojan. This is one of the 'nice' features of Hack'a'Tack, It comes with it's own built in scanner. This is a safe bet since only one port was scanned and the source port matches the one used by hack'a'tack. He or she is using the default mode as evidenced by the use of the default UDP ports.

Intent: There is intent here. He has a very specific goal in mind. If a connection is established to any responding Windows 95/98 PC, the Cracker will own that node.

Severity/Criticality: From my point of view this is very severe. If one of my family members is careless and manages to infect this machine with a Trojan I could be in big trouble. It is possible that even with my extra protective measures that there would be a window of opportunity where I could be exploited. If I lose the detects I am working on because I have left a port open through oversight, or I turn the firewall off for gaming and then forget to turn on my firewall—never happens, I could lose a lot of work. In fact I am going to back up as soon as I finish this. 8)
I&W techniques would consider this a **1 or 2** severity if we use the proper formula. I choose **2.**

Severity/Lethality: This is just a scan, and one that did not succeed. However Hack'a'Tack is a serious threat and could prove lethal. Lethality is **3** because they will own me if they succeed. They will not gain any sort of root access to the network however they will have the ability to port scan from my box which could lead to other system compromises, with me being fingered as the culprit.

System Countermeasures: I have implemented a few. This system is Windows 98, which provides little to no security — don't laugh. File and print sharing are turned off. I use Anti-Virus Software, Back-Officer, and a personal

firewall to beef up my security. I have also implemented Finjan's Surfin Shield to make sure that my family members or I, do not inadvertently open one of the many .exe's that we receive from family and friends without some sort of protection against malware. I rate this as a **3**.

Network Countermeasures: I rate this as a **3.** My firewall is working and it is blocking incoming connection attempts that I don't initiate.

Severity = (2+3) – (3+3) = **-1**

Note: Now that I've told everyone about my OS and security measures, perhaps I should increase the severity to 10. 8)

Detect #3

Overview:

On March 29, starting at 00:14.45, the preceding detects were logged on one of my customer's firewall . There are actually hundreds more of these entries that occurred over the course of the day. Each packet is a UDP packet with a length of 64, a destination port of 33434 and is destined for one of three nodes on the site. My.net.100.100 is a gateway. My.net.76.5 and .91.5 are Windows NT Proxy servers acting as cache servers. The firewall dropped all of the packets and logged them appropriately.

This particular signature confused me at first because of the consistent use of destination port 33434. Port 33434 is the first port used in a UDP trace route - see the IANA port assignments located at: http://www.isi.edu/innotes/iana/assignments/port-numbers. Normally when a trace route is performed the destination port number and TTL (Time to Live) values are incremented on each attempt. The first packet sent will use destination port 33434 and have a TTL of 1. The first router that receives the packet decrements the TTL value to 0, which causes an ICMP time exceeded error. The second packet sent by traceroute will use a TTL value of 2 and a destination port of 33435. This activity will continue with the TTL value being incremented by trace route for each hop attempt and decremented at each intervening router. The destination port is as well, normally incremented on each attempt. The port number is normally a good indicator of how far away the tracing host is, assuming you don't have the TTL value. The larger the value, the more hops that have taken place. You won't normally see a traceroute packet with a port value of 33434 that gets to your firewall. Intervening routers will have caused trace route to increment this value on each successive attempt.

In this case I do not have a TTL value for the packet as this is a firewall log and greater detail was not available. I can only surmise that this value is being incremented appropriately by the load balancer on each attempt, as the source addresses are numerous hops away from the destination hosts. There is another reason for making this ***assumption*** and that revolves around the particular activity we are seeing here.

Trace #3 – Firewall Log

Number	Date and Time		Act	port	Source	Destination	Prot	Length
"56"	"29Mar2000"	" 0:14:45"	"drop"	"33434"	"206.191.171.11"	"my.net.76.5"	"udp"	" len 64"
"57"	"29Mar2000"	" 0:14:45"	"drop"	"33434"	"206.191.171.11"	"my.net.76.5"	"udp"	" len 64"
"58"	"29Mar2000"	" 0:14:45"	"drop"	"33434"	"206.191.171.11"	"my.net.76.5"	"udp"	" len 64"
"3608"	"29Mar2000"	" 9:29:29"	"drop"	"33434"	"209.67.123.169"	"my.net.76.5"	"udp"	" len 64"
"3609"	"29Mar2000"	" 9:29:29"	"drop"	"33434"	"209.67.123.169"	"my.net.76.5"	"udp"	" len 64"
"3610"	"29Mar2000"	" 9:29:29"	"drop"	"33434"	"209.67.123.169"	"my.net.76.5"	"udp"	" len 64"
"3620"	"29Mar2000"	" 9:31:13"	"drop"	"33434"	"209.67.123.169"	"my.net.91.5"	"udp"	" len 64"
"3621"	"29Mar2000"	" 9:31:13"	"drop"	"33434"	"209.67.123.169"	"my.net.91.5"	"udp"	" len 64"
"3622"	"29Mar2000"	" 9:31:13"	"drop"	"33434"	"209.67.123.169"	"my.net.91.5"	"udp"	" len 64"
"5244"	"29Mar2000"	"11:24:49"	"drop"	"33434"	"206.191.171.4"	"my.net.91.5"	"udp"	" len 64"
"5245"	"29Mar2000"	"11:24:49"	"drop"	"33434"	"206.191.171.4"	"my.net.91.5	"udp"	" len 64"
"5246"	"29Mar2000"	"11:24:49"	"drop"	"33434"	"206.191.171.4"	"my.net.91.5"	"udp"	" len 64"
"7215"	"29Mar2000"	"12:51:30"	"drop"	"33434"	"206.251.19.88"	"my.net.100.100"	"udp"	" len 64"
"7216"	"29Mar2000"	"12:51:31"	"drop"	"33434"	"206.251.19.88"	"my.net.100.100"	"udp"	" len 64"
"7219"	"29Mar2000"	"12:51:32"	"drop"	"33434"	"206.251.19.88"	"my.net.100.100"	"udp"	" len 64"
"7221"	"29Mar2000"	"12:51:33"	"drop"	"33434"	"206.251.19.88"	"my.net.100.100"	"udp"	" len 64"
"7225"	"29Mar2000"	"12:52:27"	"drop"	"33434"	"167.8.29.91"	"my.net.100.100"	"udp"	" len 64"
"7226"	"29Mar2000"	"12:52:28"	"drop"	"33434"	"167.8.29.91"	"my.net.100.100"	"udp"	" len 64"
"7227"	"29Mar2000"	"12:52:29"	"drop"	"33434"	"167.8.29.91"	"my.net.100.100"	"udp"	" len 64"
"7228"	"29Mar2000"	"12:52:30"	"drop"	"33434"	"167.8.29.91"	"my.net.100.100"	"udp"	" len 64"
"7229"	"29Mar2000"	"12:52:31"	"drop"	"33434"	"167.8.29.91"	"my.net.100.100"	"udp"	" len 64"
"7231"	"29Mar2000"	"12:54:43"	"drop"	"33434"	"209.67.29.8"	"my.net.100.100"	"udp"	" len 64"
"7232"	"29Mar2000"	"12:54:44"	"drop"	"33434"	"209.67.29.8"	"my.net.100.100"	"udp"	" len 64"
"7233"	"29Mar2000"	"12:54:46"	"drop"	"33434"	"209.67.29.8"	"my.net.100.100"	"udp"	" len 64"
"7234"	"29Mar2000"	"12:54:47"	"drop"	"33434"	"209.67.29.8"	"my.net.100.100"	"udp"	" len 64"
"7247"	"29Mar2000"	"12:57:22"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp"	" len 64"
"7248"	"29Mar2000"	"12:57:23"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp"	" len 64"
"7249"	"29Mar2000"	"12:57:24"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp"	" len 64"
"7250"	"29Mar2000"	"12:57:25"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp"	" len 64"
"7251"	"29Mar2000"	"12:57:25"	"drop"	"33434"	"206.251.19.89"	"my.net.100.100"	"udp"	" len 64"
"7252"	"29Mar2000"	"12:58:31"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp"	" len 64"
"7253"	"29Mar2000"	"12:58:32"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp"	" len 64"
"7254"	"29Mar2000"	"12:58:33"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp"	" len 64"
"7255"	"29Mar2000"	"12:58:34"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp"	" len 64"
"7257"	"29Mar2000"	"12:59:46"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp"	" len 64"
"7258"	"29Mar2000"	"12:59:47"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp"	" len 64"
"7259"	"29Mar2000"	"12:59:49"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp"	" len 64"
"7260"	"29Mar2000"	"12:59:50"	"drop"	"33434"	"209.67.29.9"	"my.net.100.100"	"udp"	" len 64"

Analysis:

The groups responsible for these detects are:

206.191.171.11 Exodus Communications - Exodus.net
209.67.123.169 Rival Communications Network - www.rivalcom.net
206.191.171.4 Rival Communications Network
206.251.19.88 Global Crossing - telecommunications company - www.globalcrossing.com
167.8.29.91 Gannett Company - News and information Company - www.gannett.com
209.67.29.8 Exodus Communications - Exodus.net - USAToday - www.usatoday.com
209.67.29.9 Exodus Communications - Exodus.net- USAToday - www.usatoday.com

This activity is caused by WEB browsing though the stimulus is not shown in the log. Local users are browsing a number of different NEWS services that have Web Servers load balanced across multiple sites. The traffic we are seeing is the result of load balancers that are using a hop count algorithm to determine the closest application server to the requesting client.

There are a number of products on the market today, from companies like Cisco, Resonate, Nortel, F5LABS and GTE, that perform distributed load balancing. F5Labs' 3DNS, GTE Hopscotch, Resonate Global Dispatch, and Cisco's Distributed Director are all capable of performing load balancing across multiple Points of Presence or POPs. There are as well numerous algorithms that may be used by these products to determine the best path for a customer; DNS caching name server, HTTP session director, E-Business rules, SNMP metric probing, round robin, and of course hop count.

The hop count mechanism is a method of mapping the best route from the client to a POP. The latency, as measured by the trace route mechanism used by the load balancer, establishes an optimal path. The trace is initiated when a gethostbyname query is issued on behalf of the client by the local DNS server or cacheing appliance. The trace is directed at the source of the query and not at the client.

Resonate has a good description of the process flow on their web site which I have included here:

When a client connects to an internet site, it must resolve the site name (e.g., www.resonate.com) to an IP address.

The client sends a request to the local DNS.
If the requested site information is not cached in the local DNS, the local DNS server sends a request to the requested site's authoritative DNS server. The DNS server examines the name requested and determines the actual subdomain.
Global Dispatch determines if the scheduling information has been defined through an Advanced Traffic Mapping rule, or stored in cache.
If the information is not already defined or available in cache, Global Dispatch evaluates site load availability, computes latency, and determines the most appropriate POP for the client.
The IP address of the appropriate POP is returned to the client.
Client traffic is directed to the POP.

There are a number of products I know that uses a hop count algorithm, however this fingerprint is the F5LABS 3DNS controller. I have installed load balancers before at E-Commerce sites and I have run into F5Labs before. More info can be found on this product at: http://www.f5labs.com/solutions/whitepapers/3dnscontroller.html

The following is the correspondence I had with F5Labs about this issue:

My query:
I have a number of intrusion detects on customer's sites that appear to be traceroute mapping of the customer's site. The footprint for these is a UDP destination port of 33434 (the first port of traceroute using udp). Several attempts are made with the destination port not incrementing which is the usual behavior of traceroute. In none of these detects do I have the TTL value so I have to assume that it is the actual metric that is incremented by the load balancer and decremented by intervening nodes. All of this activity appears to be the result of WEB queries by users with a response from a distributed load balancer that uses a hop count algorithn. Can you tell me if this is in deed the behavior of your 3dns product when the hop count algorithm is in use?

F5Labs response:
This is correct. The only destination you should see in these queries are the dns server that queried the 3dns box. After recieveing a request the 3dns will respond with a default address for the client to go to, and then start to probe the dns server that the request came from so future requests from that dns server can be answered with the best site, according to the criteria defined in the configuration. The 3dns will continue probing the dns box long after the client is done at the site the 3dns is serving.

They state that only the DNS server should be receiving these packets. In this case the Proxy caching servers and the gateway, an Alpha Unix DNS server, are challenging the DNS servers on these sites and this is generating the hop count pattern we are seeing to these servers.

The reason for this is spelled out in RFC 1919.

“In most classical-proxy configurations, client systems pass the desired server name (or address) to the proxy system WITHOUT INTERPRETING IT. Because of this, the client system DOES NOT REQUIRE to be able to resolve the name of the server system in order to access it through a classical proxy. It only needs to be able to resolve the name of the proxy (if referencing the proxy system by name). Because of this, it can be said that a classical proxy system can offer DNS isolation. If two IP internetworks use completely separate DNS trees (each with their own DNS root servers), client software in one IP internetwork may still reference a server name in the other IP internetwork by passing its name to the classical proxy. The classical proxy itself will not be able alone to resolve DNS names in both environments (if running standard DNS resolution software), since it will need to point to one or the other of the two DNS "universes". “

Also, “Classical proxy connections have no impact on normal server software; the proxy looks like a normal client in most respects except for its IP address and its "group" nature. All connections from the network on the other side of the proxy appear to come from the proxy.”

This basically means that the Windows NT proxy server--cache server—becomes the client and does the gethostbyname query for the client. The result is that the load balancer directs it's hop count traffic to the proxy.

Note that F5Labs states that the 3DNS will continue to probe the DNS box long after the client is done at the site. So you may see this traffic long after everyone has gone home for the evening.

I decided to contact one of the sites that was generating this traffic and chose Gannett Co. because I didn't know who they were. This is a response I got back from Gannett Co. All of the sites listed above are co-located web servers.

> "Ron,
> What you are seeing in your logs are load-balancing packets because your customers are hitting http://www.usatoday.com. When a user or proxy server requests a lookup for http://www.usatoday.com, they are redirected to the first usatoday.com name server. After that first request, our load-balancing systems perform checks to determine which of our topographically closest co-location facilities to direct future traffic from that address to. The systems use pings, dns queries, and traceroutes to determine best round trip times, reliability checks, etc.
>
> You should be able to corroborate the times of the packets with any proxy logs you may have in order to verify the occurances. Although the logs you have provided do not indicate dns rule violations, if you log outbound DNS queries, you'll probably see a closer match in timings.
>
> You are right, they are F5 boxes.
>
> - Sam

This is targeted traffic but perhaps not in the normal sense because it is the result of user browsing activity. This therefore, is not hostile traffic, as the Firewall manager that sent me these logs and I initially thought, but nuisance traffic. In this particular case the firewall is dropping the packets, so "no harm, no foul.

Active Targeting: Yes

History: This is ongoing activity.

Technique: Use of a simple trace route algorithm to find the shortest path to the DNS server or proxy server that performed a gethostbyname query.

Intent: Find the shortest path between the initiating web browser's site and the web sites that the load balancer is responsible for.

Severity/Criticality: This is targeted at Core infrastructure systems. DNS servers and proxy servers. **5**

Severity/Lethality: 1 This is a mapping technique not an attack.

System Countermeasures: 3 Unknown but not hardened systems.

Network Countermeasures: 4 Restrictive firewall that is doing it's job.

Severity = (5+1) – (3+4) = **-1 Very low.**

Note: There are 2 things of note.

1) If we know about this type of load balancing and it's unique signature, you can bet that the "bad guys" know about it too. There are tools available that allow the mapper to modify the initial port used, number of attempts, whether or not the destination port increments and the source port. In particular, a technique known as firewalking can be used to map your point(s) of ingress and provide intimate knowledge of the firewall rules that you have in place. Once initial mapping using trace route is performed, the firewall can then be mapped using different source ports with queries directed to hosts behind the firewall. To detect this type of technique you need to look at the source port, the destination port, and the target function and port they are attempting to probe. A common source port for a probe using UDP would be port 53.

2) If you want to insure that this activity is indeed load balancing then take a look at the caching appliance or DNS server's outbound DNS logs. This can help in correlating the web site accessed with the load balancing response that you receive. Unsolicited traffic where there is no correlation of outbound DNS logs and the trace activity could be actual mapping attempts.

Detect #4

Overview:
The frame below was detected using black ice on an NT box. Because it was a SYN packet the daily packet log for SNORT was checked for anything involving the source IP address. The frame below is the only detailed packet available and the SNORT alert log provides evidence of the scope of the probe.

The highlighted entries in the SNORT detect are all responses from queried nodes.

```
Frame 15 (74 on wire, 74 captured)
    Arrival Time: Apr  4, 2000 20:49:31.0479
    Time delta from previous packet: 33249.951082 seconds
    Frame Number: 15
    Packet Length: 74 bytes
    Capture Length: 74 bytes
Ethernet II
    Destination: 00:c0:4f:2c:25:29 (00:c0:4f:2c:25:29)
    Source: 00:d0:bc:f2:79:6c (00:d0:bc:f2:79:6c)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Currently Unused: 0
    Total Length: 60
    Identification: 0x5434
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 48
    Protocol: TCP (0x06)
    Header checksum: 0x823e (correct)
    Source: 131.183.39.83 (131.183.39.83)
    Destination: MY.NET.70.234 (MY.NET.70.234)
Transmission Control Protocol, Src Port: 3611 (3611), Dst Port: 53 (53),
Seq: 2211586275, Ack: 0
    Source port: 3611 (3611)
    Destination port: 53 (53)
    Sequence number: 2211586275
    Header length: 40 bytes
    Flags: 0x0002 (SYN)
        ..0. .... = Urgent: Not set
        ...0 .... = Acknowledgment: Not set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..1. = Syn: Set
        .... ...0 = Fin: Not set
    Window size: 32120
    Checksum: 0xa981
    Options: (20 bytes)
        Maximum segment size: 1460 bytes
        SACK permitted
        Time stamp: tsval 78640398, tsecr 0
        NOP
        Window scale: 0 bytes
```

Trace # 4 – Snort Alert Log

There are 163 pages of alert logging for this host scan. This is just a small sample.

```
04/04-20:42:57.484472 131.183.39.83:1641 -> MY.NET.1.0:53
04/04-20:42:57.485577 131.183.39.83:1647 -> MY.NET.1.6:53
04/04-20:42:57.485655 MY.NET.1.3:53 -> 131.183.39.83:1644
04/04-20:42:57.485733 MY.NET.1.9:53 -> 131.183.39.83:1650
04/04-20:42:57.485800 MY.NET.1.1:53 -> 131.183.39.83:1642
04/04-20:42:57.486358 MY.NET.1.7:53 -> 131.183.39.83:1648
04/04-20:42:57.486438 MY.NET.1.2:53 -> 131.183.39.83:1643
04/04-20:42:57.486821 131.183.39.83:1660 -> MY.NET.1.19:53
04/04-20:42:57.486898 131.183.39.83:1664 -> MY.NET.1.23:53
04/04-20:42:57.488209 131.183.39.83:1672 -> MY.NET.1.31:53
04/04-20:42:57.525817 MY.NET.1.3:53 -> 131.183.39.83:1644
04/04-20:42:57.527121 MY.NET.1.9:53 -> 131.183.39.83:1650
04/04-20:42:57.527187 MY.NET.1.7:53 -> 131.183.39.83:1648
04/04-20:42:57.527570 131.183.39.83:1694 -> MY.NET.1.53:53
04/04-20:42:57.527649 131.183.39.83:1693 -> MY.NET.1.52:53
04/04-20:42:57.529024 131.183.39.83:1695 -> MY.NET.1.54:53
04/04-20:42:58.488617 131.183.39.83:1698 -> MY.NET.1.57:53
…
04/04-20:43:16.513253 131.183.39.83:2650 -> MY.NET.4.243:53
04/04-20:43:16.513330 131.183.39.83:2654 -> MY.NET.4.247:53
04/04-20:43:16.513410 131.183.39.83:2658 -> MY.NET.4.251:53
04/04-20:43:16.513488 131.183.39.83:2661 -> MY.NET.4.254:53
…
04/04-20:43:17.591603 131.183.39.83:2663 -> MY.NET.5.1:53
04/04-20:43:17.592907 131.183.39.83:2677 -> MY.NET.5.15:53
04/04-20:43:17.592985 131.183.39.83:2670 -> MY.NET.5.8:53
04/04-20:43:17.593063 131.183.39.83:2674 -> MY.NET.5.12:53
04/04-20:43:17.593139 MY.NET.5.13:53 -> 131.183.39.83:2675
04/04-20:43:17.593208 MY.NET.5.5:53 -> 131.183.39.83:2667
04/04-20:43:17.594517 131.183.39.83:2682 -> MY.NET.5.20:53
04/04-20:43:17.602027 MY.NET.5.19:53 -> 131.183.39.83:2681
04/04-20:43:17.654984 131.183.39.83:2722 -> MY.NET.5.60:53
04/04-20:43:17.655062 131.183.39.83:2723 -> MY.NET.5.61:53
04/04-20:43:17.672750 131.183.39.83:2732 -> MY.NET.5.70:53
04/04-20:43:17.711851 131.183.39.83:2733 -> MY.NET.5.71:53
04/04-20:43:18.493087 131.183.39.83:2743 -> MY.NET.5.81:53
04/04-20:43:18.493160 131.183.39.83:2750 -> MY.NET.5.88:53
04/04-20:43:18.493472 131.183.39.83:2752 -> MY.NET.5.90:53
04/04-20:43:18.493649 131.183.39.83:2755 -> MY.NET.5.93:53
04/04-20:43:18.493711 131.183.39.83:2759 -> MY.NET.5.97:53
04/04-20:43:18.493787 131.183.39.83:2760 -> MY.NET.5.98:53
04/04-20:43:18.493866 131.183.39.83:2764 -> MY.NET.5.102:53
04/04-20:43:18.493932 MY.NET.5.100:53 -> 131.183.39.83:2762
04/04-20:43:18.494010 131.183.39.83:2770 -> MY.NET.5.108:53
04/04-20:43:18.494090 131.183.39.83:2769 -> MY.NET.5.107:53
04/04-20:43:18.494227 MY.NET.5.111:53 -> 131.183.39.83:2773
04/04-20:43:18.494497 MY.NET.5.105:53 -> 131.183.39.83:2767
04/04-20:43:18.494564 MY.NET.5.99:53 -> 131.183.39.83:2761
…
```

```
04/04-21:02:37.766689 131.183.39.83:2568 -> MY.NET.253.102:53
04/04-21:02:37.766768 131.183.39.83:2570 -> MY.NET.253.104:53
04/04-21:02:37.766848 131.183.39.83:2573 -> MY.NET.253.107:53
04/04-21:02:37.766929 131.183.39.83:2571 -> MY.NET.253.105:53
04/04-21:02:37.767005 131.183.39.83:2572 -> MY.NET.253.106:53
04/04-21:02:37.767081 131.183.39.83:2575 -> MY.NET.253.109:53
04/04-21:02:37.767160 131.183.39.83:2574 -> MY.NET.253.108:53
04/04-21:02:37.767242 131.183.39.83:2581 -> MY.NET.253.115:53
04/04-21:02:37.767317 131.183.39.83:2583 -> MY.NET.253.117:53
04/04-21:02:37.767395 131.183.39.83:2585 -> MY.NET.253.119:53
04/04-21:02:37.767475 131.183.39.83:2582 -> MY.NET.253.116:53
04/04-21:02:37.767686 131.183.39.83:2577 -> MY.NET.253.111:53
04/04-21:02:37.767765 131.183.39.83:2579 -> MY.NET.253.113:53
04/04-21:02:37.767845 131.183.39.83:2576 -> MY.NET.253.110:53
04/04-21:02:37.767923 131.183.39.83:2578 -> MY.NET.253.112:53
04/04-21:02:37.768001 131.183.39.83:2580 -> MY.NET.253.114:53
04/04-21:02:37.768080 131.183.39.83:2584 -> MY.NET.253.118:53
04/04-21:02:37.768161 131.183.39.83:2589 -> MY.NET.253.123:53
04/04-21:02:37.768226 MY.NET.253.108:53 -> 131.183.39.83:2574
04/04-21:02:37.768303 131.183.39.83:2587 -> MY.NET.253.121:53
04/04-21:02:37.768381 131.183.39.83:2586 -> MY.NET.253.120:53
04/04-21:02:37.768463 131.183.39.83:2591 -> MY.NET.253.125:53
04/04-21:02:37.768527 MY.NET.253.109:53 -> 131.183.39.83:2575
04/04-21:02:37.768593 MY.NET.253.107:53 -> 131.183.39.83:2573
04/04-21:02:37.768674 131.183.39.83:2593 -> MY.NET.253.127:53
04/04-21:02:37.768755 131.183.39.83:2590 -> MY.NET.253.124:53
04/04-21:02:37.768832 131.183.39.83:2588 -> MY.NET.253.122:53
04/04-21:02:37.768907 131.183.39.83:2592 -> MY.NET.253.126:53
04/04-21:02:37.768987 131.183.39.83:2595 -> MY.NET.253.129:53
…
04/04-21:02:42.804099 131.183.39.83:2851 -> MY.NET.254.130:53
04/04-21:02:42.804155 131.183.39.83:2853 -> MY.NET.254.132:53
04/04-21:02:42.804233 131.183.39.83:2850 -> MY.NET.254.129:53
04/04-21:02:42.804310 131.183.39.83:2846 -> MY.NET.254.125:53
04/04-21:02:42.804389 131.183.39.83:2844 -> MY.NET.254.123:53
04/04-21:02:42.804467 131.183.39.83:2848 -> MY.NET.254.127:53
04/04-21:02:43.801043 131.183.39.83:2890 -> MY.NET.254.169:53
04/04-21:02:44.795187 131.183.39.83:2924 -> MY.NET.254.203:53
04/04-21:02:44.796316 131.183.39.83:2926 -> MY.NET.254.205:53
```

Activity Summary:
This detect is the result of a Host scan for DNS services or a scan for zone transfer. This activity was initiated by a node using IP address 131.183.39.83. A DNS query on this address using Sam Spade results in the following entry:

```
04/06/00 04:07:22 IP block 131.183.39.83
Trying 131.183.39.83 at ARIN
Trying 131.183.39 at ARIN
University of Toledo (NET-UTOLEDO-NET)
   2801 West Bancroft
   Toledo, OH  43606

   Netname: UTOLEDO-NET
   Netnumber: 131.183.0.0

   Coordinator:
      Nelson, Brian  (BN38-ARIN)  brian@UOFT02.UTOLEDO.EDU
      (419) 537-2841 537-4309

   Domain System inverse mapping provided by:

   UTNETW.UTOLEDO.EDU  131.183.1.1
   NS1.OAR.NET   192.88.193.144

   Record last updated on 16-Jun-1994.
   Database last updated on 6-Apr-2000 06:20:37 EDT.
```

Analysis:

Originally I figured that this was an nmap scan. This is an extremely fast, brute force scanning technique. So fast that it appears to be overloading SNORT. I had even figured out the possible command that the perpetrator could have used: **Nmap -sT –p53 -T5 –oN mynet.log 'my.net.*.*' or Nmap –Ss –p53 -T5 –oN mynet.log 'my.net.*.*'**

-sT TCP connect() scan: This is the most basic form of TCP scanning. The connect() system call provided by your operating system is used to open a connection to every interesting port on the machine. If the port is listening, connect() will succeed, other- wise the port isn't reachable. One strong advantage to this technique is that you don't need any special privileges. Any user on most UNIX boxes is free to use this call. This sort of scan is easily detectable as target host logs will show a bunch of connection and error messages for the services which accept() the connection just to have it immediately shutdown.

-sS TCP SYN scan: This technique is often referred to as "half-open" scanning, because you don't open a full TCP connection. You send a SYN packet, as if you are going to open a real connection and you wait for a response. A SYN|ACK indicates the port is listening. A RST is indicative of a non-listener. If a SYN|ACK is received, a RST is immediately sent to tear down the connection (actually our OS kernel does this for us). The primary advantage to this scanning technique is that fewer sites will log it. Unfortunately you need root privileges to build these custom SYN packets.
The other nmap options are described at http://www.insecure.org/nmap/nmap_manpage.html .

"A SYN scan is yet another attempt to bypass system level port scan detection. On most systems, a network connection is only recorded if the TCP three way handshake is completed. SYN scans send a single SYN packet to the destination port and then waits to see the response. If the response is a RESET packet, then the port is dead and no further action is taken. If the response is a SYN-ACK packet, then a RESET packet is sent back to

the target that disengages the three way handshake. Sometimes this RESET packet is generated by the SYN scan program and other times it is generated directly by a response from the scanning operating system's IP stack."
Ref: How to handle and Identify network probes. Ron Gula

The above information on how a SYN scan works, rules out the possibility of a simple SYN or half open scan to these systems. The fact that we are getting multiple responses from some hosts means that this is a different scanning technique. If this was a SYN scan using the –sS option, why the multiple responses from some of these hosts, why the completion of the handshake? Nmap should send a reset if this is a SYN or a half open scan. In a half open scan it sends a reset as soon as it sees the SYN ACK. If it's a SYN scan then the reset occurs upon completion of the handshake but there should be no more communication between the source and the target.

Because of this information I revised my opinion of what is going on and believe that this is a scan for zone transfer, based on this being TCP connection attempts, and the further dialogue beyond the completion of the three way handshake that is indicated in the detect.

http://www.cotse.com/name.htm and of course rootshell have a number of DNS scanning utilities and exploits that may have been used in this instance.

This is active targeting. The perpetrator "appears to be" host scanning using TCP SYN packets, of this site's entire address space, searching for open DNS ports (port 53). However, the analysis will show that there is much more going on than a mere SYN scan. The Snort Alert log does not specifically state that these are all TCP SYN packets, I therefore asked the IDS analyst for this site to provide more packet detail. The packets in the alert log from 131.183.39.83 are all TCP SYN packets. This is a typical example:

```
04/04-20:42:57.484472 131.183.39.83:1641 -> MY.NET.1.0:53
TCP TTL:49 TOS:0x0 ID:3656  DF
**S***** Seq: 0x6E88EDE7   Ack: 0x0   Win: 0x7D78
TCP Options => MSS: 1460 SackOK TS: 78605898 0 EOL EOL EOL EOL
```

The activity here is fast and furious and the entire class B address space is being scanned. Target addresses from my.net.1.0 to my.net.254.205 are scanned using a script or a tool. Addresses beyond my.net.254.205 may have been scanned but the log does not show the entries.

There is no evidence of decoying as we have only one source address.

The source ports increment in a logical fashion. Many of these packets arrive out of order and the source ports coincide with what they should have been if the packets had traversed the network by the same path. There is a direct correlation between the target address and the source port used. They increment in lockstep. This means that the target IP addresses that do not show up in the scan are actually being scanned but the packets are lost.

So why is there packet loss evident? There appears to be packet loss on the SNORT detector, possibly due to the speed of the scan or hardware limitations. In e-mail correspondence with the owner of this site, I discovered that the detector is configured on a switch using 10M spanning ports. So the packet loss may also be due to the spanning ports which feed the detector becoming overloaded. A number of the responses to the initiator appear to be unsolicited which is most definitely not the case, the detector just couldn't handle the load. If the packet loss is due to overload of the spanning port, then we could verify this by looking at the counters on the spanning ports for load, and check for dropped or errored packets.

The yellow highlighted area of the alert log shows responses coming from nodes that the detector did not scan. My.net.1.1 and .1.2 do not show an initiating SYN scan and yet these nodes responded. This can only mean that the sensor dropped the active open packets. This is a more detailed packet showing the actual response that my.net.1.1 sent. It is a reset ack. This is TCP's way of telling the perpetrator that DNS is not active on this system.

```
04/04-20:42:57.485800 MY.NET.1.1:53 -> 131.183.39.83:1642
TCP TTL:255 TOS:0x0 ID:13488
****R*A* Seq: 0x0   Ack: 0x6E8ABA3E   Win: 0x0
00 00 00 00 00 00
```

Note: Many of the responses to this probe are Reset ACKs however not all.

This is an example of one of the other response types seen:

```
04/04-20:44:05.527832 MY.NET.15.13 -> 131.183.39.83
ICMP TTL:253 TOS:0x0 ID:4771
DESTINATION UNREACHABLE: PROTOCOL UNREACHABLE
00 00 00 00 45 00 00 3C 1C 66 40 00 30 06 F1 E9  ....E..<.f@.0...
83 B7 27 53 82 55 0F 0D 04 E2 00 35 72 94        ..'S.U.....5r.
```

Note: The initial packet to my.net.15.13 is not shown in the logs. This would normally indicate that a UDP packet was directed at my.net.15.13 based on the response. In this case however, it is more likely that for some reason the host is unreachable and that this is a router response. I have only found one other indication of ICMP unreachable messages and that was to my.net.15.12. This response was found by luck more than anything else. due to the size of the logs it was very easy to go buggy eyed staring at the screen.

Most important of all, the yellow highlighted section shows additional responses from my.net.1.3, .7 and .9. These nodes do not just respond to the SYN packet, but some additional dialogue is indicated. The destination ports of the second response between my.net.1.3, 1.7 and 1.9, and the scanner are constant between the two attempts. I was not sure what to make of this. It seemed that Snort missed the completion of the three way handshake, an ACK from 131.183.39.83, and data was transferred (so minimally the bad guy got the version of bind and possibly did a zone transfer). The additional packet detail tells some of the story.

My.net.1.3 responds by sending a SYN ACK, step 2 of the three way handshake:

```
04/04-20:42:57.485655 MY.NET.1.3:53 -> 131.183.39.83:1644
TCP TTL:254 TOS:0x0 ID:64373  DF
**S***A* Seq: 0x9895D95B   Ack: 0x6E7B65D5   Win: 0x2798
TCP Options => NOP NOP TS: 61400562 78605898 NOP WS: 0 EOL EOL EOL EOL
```

We never see the ACK come back from the perpetrator but we must assume that it occurred. The final packet that we see between my.net.1.3 and 131.183.39.83 is a FIN ACK packet. A graceful termination of the dialogue. The perpetrator has got what he came for and terminates the connection with a FIN. The target ACKs, which we don't see, and then sends the packet detailed below.

```
04/04-20:42:57.525817 MY.NET.1.3:53 -> 131.183.39.83:1644
TCP TTL:254 TOS:0x0 ID:64376  DF
***F**A* Seq: 0x9895D95C   Ack: 0x6E7B65D6   Win: 0x2798
TCP Options => NOP NOP TS: 61400566 78605902
```

We never see the ACK for this final FIN packet. This is another indicator that ACKs are not being logged.

I did some further checking in the alert log and the completion of the three way handshake occurs on a number of occasions throughout the detect. In none of the cases is completion of the hand shake, the final ACK from the scanner, detected by snort. Nor is any 'pure' ACK traffic seen in any of the logs I looked at. Does this mean the final ACK is being filtered out by SNORT or did these packets just get dropped? I think in this case, though it is possible that some of these packets could have been dropped, that the ACK packets are being filtered.

The green highlighted area shows several more responses from targeted hosts and is indicative of the entire address space scanned. The single packet responses are resets.

The blue highlighted area gives us a great indication of how fast this scanning technique is. The "out of order" packets are normally indicative of the many different paths that the packets are taking over the Internet. The TTL value should be different for those packets that arrive "out of order". In the packet detail the TTL value for every packet from the perpetrator is 49. It never varies on any packet examined whether the packet came in order or appears to have traveled a different path over the Internet. This is anomalous behavior and the value may be crafted.

The source of these packets is most likely a compromised host at the University of Toledo or we have someone that is spoofing the address and gathering the traffic using a sniffer. The anomalous TTL value may be indicative of this. In any case the cracker doesn't mind if this system is found as the source of the offending packets. If the perpetrator is gathering packets with a sniffer, it is possible that the source of these packets is internal to the Organization and had some passing knowledge of the University of Toledo. This is where it becomes very important to know where your detectors are placed.

Remember the packet detail we got from BlackIce? There is a MAC address for the Source of these packets: 00:d0:bc:f2:79:6c. It would be interesting to see if this frame physical layer address corresponds to an internal node at this site or to some router or other node.

There is a very real threat here due to the large number of nodes that are actually responding to the scan. The small sample that I have provided does not really show the staggeringly large number of responses that this probe got. I stopped counting at 125 and had only gotten to my.net.53.219. This has a number of serious implications:

1) The customer does not use a firewall. Instead they rely on a packet filtering router. It is allowing all port 53 requests into the internal network. This needs to be rectified and would go a long way to mitigating the risk.
2) Judging by the number of responses, many of these boxes are probably Linux or Unix boxes running the 'named' service. There is no need for these services on these boxes and 'named' should be removed. See recommendations.
3) We do not know what packets were returned to the cracker or the commands sent by the hacker to the target. Many of the responses were resets. However, we know that the three-way handshake was completed in some instances and that further data was exchanged.
4) There are a number of vulnerabilities that can be exploited on each of the responding nodes, and it is possible that some of them have now been compromised. Namedsploit is a buffer overflow technique that could be used to compromise some of these responding hosts—assuming less than adequate patch levels. It is then a simple matter for the attacker to create the conditions for an x-term to be started or run execve()s /bin/sh.
5) Much of the activity during this scan was dropped by the IDS; effectively a DoS on the IDS. There is a possibility that some more serious hacking occurred during the partial blackouts caused by the scan. The multiple responses from some hosts may be indicative of this.
6) It is possible that **not** every one of the systems mentioned has been patched since the CERT advisory multiple vulnerabilities in BIND (CA-99-14 Nov 10, 1999) came out. This advisory identifies a number of named exploits.
7) For a bad guy to make use of the named exploit they really need to know a little more than just the version of named you are using. Otherwise they are forced to use shotgun tactics and hope that they get a version of named that is susceptible to hacking. This means they need to know the version of named, the OS and how

named was run. I would expect more activity on the systems that answered, possibly scripted attacks that verify the version of bind and launch an appropriate attack.

Active Targeting: Yes.

History: None identified.

Technique: Very fast scanning technique. Entire address space hit. Scanning performed on a standalone workstation or server with little or no other activity evident. I base this conclusion on the source ports used in the scan which increment once per scan and not by leaps and bounds. I believe that this is a scan for zone transfer, based on TCP connection attempts to DNS port 53 and the further dialogue, beyond the completion of the three way handshake, that is indicated in the detect. http://www.cotse.com/name.htm and of course rootshell have a number of DNS scanning utilities and exploits that may have been used in this instance.

Severity/Criticality: 5 This is active targeting of all manner of systems in the network. It is quite possible that important systems are responding.

Severity/Lethality: 5 Although this particular signature is likely that of a scan for Zone transfer, and does not definitively show that any systems have been compromised it does show that there is a huge potential for root access to be gained on multiple systems in this network. As I stated earlier, I gave up counting the number of responding hosts at 125. Scanning such as this is a precursor to an attack. There are way too many systems responding on port 53.
There are some serious vulnerabilities with the named service on various flavors of Unix and Linux and not knowing the patch level of these systems makes me look at this with a very critical eye. We should also keep in mind that several of these systems completed the three way handshake and sent data to the attacker.

System Countermeasures: 2 I don't know what system countermeasures are in place but with the number of systems responding, I have to assume that the systems have not been hardened and are probably not all patched to the latest versions.

Network Countermeasures: 2 The only effective countermeasure on this site is the packet filtering router and it is mis-configured or perhaps I should say loosely configured.

Severity = (5 + 5) – (2 + 2) = **6 Severe. Possibility of compromise definitely exists.**
Recommendations:

1) Contact the University of Toledo and inform them of this activity. They have most likely been compromised if they are indeed the source of these packets. Find out what type of system the scanning source is and a TTL value if possible. Ask if they have corroborating evidence of this probe on the 4rth of April.

2) Give priority to the hosts that appear to have completed the three-way handshake. Check these systems out thoroughly. Tripwire would have been a great tool to have installed in this case. Look at the traffic that occurred in these instances using detailed packet analysis; Hex dumps.

3) Perform a number of trace routes to the perpetrator and find out what the hop count is. Use the system type you discovered above to find the default TTL value for the node, if you didn't get one in step one. Subtract your mean hop count value and see if it comes out to 49. For example, AIX uses a default value of 60. If your hop count is in the neighborhood of 11 then this will provide some correlation. If there is no match then the TTL value was crafted. This might indicate a scenario where the address was spoofed and someone is gathering responses with a sniffer. The TTL value is then crafted to appear to be coming from a number of hops away.

4) To improve the network security shut down TCP port 53 inbound except for designated DNS servers. Since name lookups use UDP (except for very large DNS names), and zone transfers use TCP port 53, this will effectively shut down unauthorized zone transfers. Set up filters so that only Internal DNS servers may initiate and talk to the external ISP DNS servers. In other words, make sure a split brain DNS system is set up. They

really need a firewall to do this effectively.

5) Actively probe your own network using a tool like NMAP to locate these services and shut them down. Do this on a regular basis. Messala v1.5 is a good tool for finding vulnerable services such as named and RPC vulnerabilities.

6) From a DNS host configuration perspective zone transfers should be restricted to authorized DNS servers. The xfernets directive in the named.boot file can be used to enforce this restriction; at least on the newer versions of bind.

To configure zone security on a Windows NT server, use the following procedure:
Click Start, click Programs, click Administrative Tools (Common), and then click DNS Manager.
In DNS Manager, from the Server list, right-click the primary zone icon.
Click Properties.
Click the Notify tab.
In the Notify List, add the IP addresses of the secondaries that are allowed to access the primary.
Click the "Only Allow Access From Secondaries Included on the Notify List" check box.
Ref: http://support.microsoft.com/support/kb/articles/Q193/8/37.ASP?LN=EN-US&SD=gn&FR=0

Remove the named service on all of the responding Linux workstations and servers.
On a Linux box:
/etc/rc.d/init.d/named stop
rpm –e caching-nameserver
rpm –e bind
edit /etc/resolv.conf and add the IP address of your primary and secondary name servers. Make sure there is not an entry that points to your local IP number or 127.0.0.1
ref: Securing Linux Step-By-Step version 1.0

Detect #5

Overview:
The following detect is actually a subset of a daily snort alert from a .edu site. The log is 22 pages long. I pulled these entries because they all targeted the same node and port number and I had no idea what the port number was for. There is no information as to the function of the targeted node. The sensors on this site are connected to switches with port spanning enabled. There is no indication of any response from the targeted host.

Trace #5 – Snort Alert Log

```
1     [**] Null scan! [**]
      03/23-13:10:58.708950 24.112.172.98:2023 -> MY.NET.10.119:6699
2     [**] Null scan! [**]
      03/23-15:59:12.535209 24.92.244.23:1160 -> MY.NET.10.119:6699
3     [**] Null scan! [**]
      03/23-16:05:18.757998 24.92.244.23:1152 -> MY.NET.10.119:6699
4     [**] Null scan! [**]
      03/23-16:07:51.786229 24.66.174.129:2413 -> MY.NET.10.119:6699
5     [**] Null scan! [**]
      03/23-16:37:29.632437 24.92.244.23:1179 -> MY.NET.10.119:6699
6     [**] Null scan! [**]
      03/23-16:37:50.623460 24.92.244.23:1189 -> MY.NET.10.119:6699
7     [**] Probable NMAP fingerprint attempt [**]
      03/23-16:38:35.448817 24.92.244.23:1178 -> MY.NET.10.119:6699
8     [**] Null scan! [**]
      03/23-17:38:28.793281 24.200.61.225:1276 -> MY.NET.10.119:6699
```

Analysis:

It appears that we have several attempts to 'fingerprint' my.net.10.119. A null scan is an example of a fingerprinting technique. The theory is that certain operating systems will respond, others will not. The form that the response takes provides the cracker with evidence of the OS type in question. This allows the attacker to target vulnerable services specific to the OS in question.

"Stealth FIN, Xmas Tree, or Null scan modes: There are times when even SYN scanning isn't clandestine enough. Some firewalls and packet filters watch for SYNs to restricted ports, and programs like Synlogger and Courtney are available to detect these scans. These advanced scans, on the other hand, may be able to pass through unmolested. The idea is that closed ports are required to reply to your probe packet with an RST, while open ports must ignore the packets in question (see RFC 973 pp 64)."
Ref: http://www.insecure.org/nmap/nmap_manpage.html

The null scan indicates a TCP frame has been seen with a sequence number of zero and all control bits set to zero. This frame should never be seen in normal TCP operation. The perpetrator(s) are scanning my.net.10.119 by proffering unconventional stimulus in these crafted packets. Sometimes this is done in preparation for a future attack, and is often used to see if your system might have a service which is susceptible to attack. In this case the attacking systems are very specific about the host and service that they are interested in. They have already at least partially mapped the network.

In three of the four cases, a single null scan is performed. Packet 1, 4 and 8 are one shot null scans from three separate sources. It is possible that these are all from a single individual using spoofed addresses in all but one case or that the perpetrator has compromised all of these systems.

So what about the port being targeted? 6699 is the port for a program called **NAPSTER,** which is used for exchanging MP3 files and chatting. Here is what Napster has to say about their product:

> "Napster is a completely new way of thinking about music online.
> Imagine...an application that takes the hassle out of searching for MP3s. No more broken links, no more slow downloads, and no more busy, disorganized FTP sites. With Napster, you can locate and download your favorite music in MP3 format from one convenient, easy-to-use interface."
> **What else does it do?**
> Quite a bit, actually. Some highlights include:
>
> > **CHAT** – Allows users to chat with each other in forums based on music genre.
> > **AUDIO PLAYER** - Plays MP3 files from right inside Napster, in case you don't have an external player or would prefer not to use one.
> > **HOTLIST** - Lets you keep track of your favorite MP3 libraries for later browsing.

So why are these four characters trying to access port 6699 in such a stealthy fashion? Simple connect attempts to this server would not be overly suspicious and the logical response, by the owner of the IDS, would be to check to see if the service is actually offered on my.net.10.119 and shut it down if it is running. This assumes the Security Policy does not allow napster. Depending on the position of the sensor, it is possible that port 6699 is blocked on the firewall or packet filtering router and these stealthy scans are attempts at circumventing it and the IDS.

The attached quote may shed some light on this activity:

"Another controversy around the software (napster), however, has centered on the way it dramatically escalates network use. Taking up end-user bandwidth like few Internet applications before it, the service has provided something of a litmus test for broadband networks to handle popular new media applications of the future.
The Cox move extends for the first time an **ongoing battle on campuses** over Napster use to the consumer ISP realm. University network administrators have blocked student access to the service, blaming Napster users for soaking up 40 to as much as 60 percent of campus bandwidth. "

In looking at this little tidbit of information I would have to consider that perhaps this site is not or was not blocking 6699 and that some users have set up Napster servers. If so, it would be trivial for hackers that make use of these services to discover the port number that napster uses and the ip address of the server that they connect to. A nmap null scan and or fingerprinting could be used to find out how susceptible this system might be to hacking.

It is also important to note that this product comes with a chat channel and these are notorious for handing out IP addresses.

There is one thing about this I don't properly understand. Napster runs on Windows 95, 98 and Windows NT. All of these products use the same basic TCP/IP stack. This basically means that fingerprinting these systems using null scans will not tell you whether or not the target is running Windows 95, 98, or NT. Anyone that knows anything about Napster should be able to figure out that the target is running Windows of some flavor as these are the only OS's that run Napster. If they are using Nmap, and they read the manual, it states:

"The Null scan turns off all flags. Unfortunately Microsoft (like usual) decided to completely ignore the standard and do things their own way. Thus **this scan type will not work against systems running Windows95/NT**. On the positive side, this is a good way to distinguish between the two platforms(RR: meaning Unix flavors and Windows). If the scan finds open ports, you know the machine is not a Windows box. If a -sF,-sX,or -sN scan shows all ports closed, yet a SYN (-sS) scan shows ports being opened, you are probably looking at a Windows box. "

So a SYN scan would also be needed to correctly identify this as a windows box. There is no indication that this occurred.

Without more information on the target of these probes it is impossible to take this analysis any further.

Whois reverse mapping using Sam Spade reveals the following about the four actors in this detect:

```
04/09/00 11:26:23 IP block 24.112.172.98
Trying 24.112.172.98 at ARIN
Trying 24.112.172 at ARIN
Rogers@Home Ontario (NETBLK-ROGERS-1-BLOCK) ROGERS-1-BLOCK
         24.112.0.0 - 24.112.255.255
Rogers@Home Ktchnr (NETBLK-ON-ROG-KTCH-3) ON-ROG-KTCH-3
       24.112.172.0 - 24.112.173.255
```

```
04/09/00 11:27:18 IP block 24.92.244.23
Trying 24.92.244.23 at ARIN
Trying 24.92.244 at ARIN
Time Warner Cable (NETBLK-RR-1) RR-1       24.92.0.0 - 24.95.255.255
TimeWarnerCable-Road-Runner-TWCNY-24-92-241-0-to-244-0 (NETBLK-RRSYRACUSE1)
RRSYRACUSE1
         24.92.241.0 - 24.92.244.255
```

```
04/09/00 11:28:08 IP block 24.66.174.129
Trying 24.66.174.129 at ARIN
Trying 24.66.174 at ARIN
Shaw Fiberlink ltd. (NETBLK-FIBERLINK-CABLE)
   630 3rd Avenue SW, Suite 900
   Calgary, AB T2P 4L4
   CA
   Netname: FIBERLINK-CABLE
   Netblock: 24.64.0.0 - 24.70.95.255
   Maintainer: FBCA

   Coordinator:
      Shaw at Home, TOC  (SH2-ORG-ARIN)  internet.abuse@SHAW.CA
      (403) 750-7420
Fax- (403) 750-4501
```

```
04/09/00 11:28:57 IP block 24.200.61.225
Trying 24.200.61.225 at ARIN
Trying 24.200.61 at ARIN
Videotron Ltee (NETBLK-VL-2BL) VL-2BL     24.200.0.0 - 24.201.255.255

Videotron Ltee (NETBLK-VL-M-MH-18C83600) VL-M-MH-18C83600
         24.200.54.0 - 24.200.61.255
```

Active Targeting: Yes.

History: I have no history on these source addresses other than they all chose the same day to attempt null scans of port 6699. Other logs for this site show quite a bit of activity revolving around port 6699 from other source addresses to varying target systems. These other detects were not null scanning or fingerprinting techniques but connection attempts.

Technique: Null scan is attempting to evade detection by firewalls and IDS systems. At least one of these used nmap or something very similar. The scanning was done quite slowly; almost four and a half hours. If they were truly going after the Napster port and trying to fingerprint this system, we should have seen a SYN scan as well because Nmap cannot distinguish between Windows flavors by using bogus flags such as the null scan.

Intent: This is hard to say. The probe is so focused that I have to believe that they know that this host has a service running on 6699 or that it did in the past. With Educational institutes closing down access to napster, these individuals may be trying to find ways of circumventing their access controls or find ways of getting even for shutting down the service. It is also possible that the user of the target node has drawn unwanted attention in the chat channels they have visited.

Severity/Criticality: 3 I don't know the criticality of this system. It could be, and most likely is, a desktop workstation running Windows of some flavor but I will err on the side of caution and call it an NT server. The IDS expert on site should track this system down.

Severity/Lethality: 2 These are stealthy fingerprinting scans to a system that may or may not have anything running on 6699. This is pure recon. There is no evidence to suggest that this traffic actually got through to the system in question however as was pointed out a Windows system won't respond to these probes so the lack of a response is inconclusive. I know of no hacks for Napster, though just about anything is susceptible to buffer overflows nowadays.

System Countermeasures: Unknown. We cannot define categorically what system this is, or what OS, and whether or not there is a service running on port 6699. The systems patch level is also unknown. Napster runs on Windows 95, 98, and NT. There is a port coming out for Macs. **2**

Network Countermeasures: Unknown. There is a packet filtering router at this site. They use a capable IDS but I do not know where this sensor is located or I should say, I don't know where it is receiving it's packet stream from. It could be detects from the DMZ, screened subnet or Intranet The detector is being fed via spanned ports which could lead anywhere in the network. **3**

Severity = (3+2) – (2+3) = **0 Low severity**

Recommendations: Track down this system and check to see if it is a Windows box running Napster or if some other service is running on port 6699. I might do a quick Trojan check on the box as well using "thecleaner" or something similar. Netstat –a will quickly tell you if a service is running and what port it is listening on. You could also do a search for the napster executable. Find out from the user/administrator if napster ever ran on this system. If it is still an active service, and running this service goes against policy, shut it down. If using napster is not against policy inform the indivual responsible that they are being targeted and find out why, if possible. Find out if the chat function is being used and what chat channels they have been to. Verify that the packet filtering router or firewall drops incoming and outgoing port 6699. However be aware, "Napster already works with firewalls that block all incoming connections. Users can hit the configuration button that tells napster this. However, users will no longer be able to transfer files with other users behind firewalls."

As well, note that users can configure the Data Port to be some other value than 6699. It could be any value in the range 1024-65535. Another way around firewall blocking is to use SOCKS proxies.

Detect #6

Overview:

This detect was found using SNORT. There are two sets of 15 attempts at communicating with node my.net.253.24. The first is an attempt to connect to 34555. Roughly half an hour later another attempt is made from a different source address and destination port 35555. These are both known ports associated with the Win32 Trin00 Trojan. Trin00 for Windows uses UDP on port 34555 and on port 35555. There is no indication on whether there was any response to this communication. In fact there is no indication that TCP or UDP was actually used in either case due to the lack of flag information or any other indicator suggesting the actual protocol used.

These are two random samplings of several of these signatures that occur on this site on a fairly regular basis.

Trace #6 – Snort Alert Log

```
[**] GIAC 000218 VA-CIRT port 34555 [**]
03/24-22:14:49.945123 38.170.72.2:25 -> MY.NET.253.24:34555
[**] GIAC 000218 VA-CIRT port 34555 [**]
03/24-22:14:49.995437 38.170.72.2:25 -> MY.NET.253.24:34555
[**] GIAC 000218 VA-CIRT port 34555 [**]
03/24-22:14:50.103846 38.170.72.2:25 -> MY.NET.253.24:34555
[**] GIAC 000218 VA-CIRT port 34555 [**]
03/24-22:14:50.212387 38.170.72.2:25 -> MY.NET.253.24:34555
… Removed several attempts for clarity
[**] GIAC 000218 VA-CIRT port 34555 [**]
03/24-22:14:50.617964 38.170.72.2:25 -> MY.NET.253.24:34555
[**] GIAC 000218 VA-CIRT port 34555 [**]
03/24-22:14:50.640456 38.170.72.2:25 -> MY.NET.253.24:34555
[**] GIAC 000218 VA-CIRT port 34555 [**]
03/24-22:14:52.022045 38.170.72.2:25 -> MY.NET.253.24:34555



[**] GIAC 000218 VA-CIRT port 35555 [**]
03/24-22:56:01.529432 24.92.226.134:25 -> MY.NET.253.24:35555
[**] GIAC 000218 VA-CIRT port 35555 [**]
03/24-22:56:01.627347 24.92.226.134:25 -> MY.NET.253.24:35555
[**] GIAC 000218 VA-CIRT port 35555 [**]
03/24-22:56:01.628578 24.92.226.134:25 -> MY.NET.253.24:35555
… removed several attempts for clarity
 [**] GIAC 000218 VA-CIRT port 35555 [**]
03/24-22:56:03.568637 24.92.226.134:25 -> MY.NET.253.24:35555
[**] GIAC 000218 VA-CIRT port 35555 [**]
03/24-22:56:03.735423 24.92.226.134:25 -> MY.NET.253.24:35555
[**] GIAC 000218 VA-CIRT port 35555 [**]
```

Technical Information:

About SMTP:
I have put this piece of information in this analysis because it is important to understand how SMTP communicates between hosts. A proper understanding of this protocol can assist in defining whether this is a false alarm or not.

"SMTP Connection Establishment: The SMTP transmission channel is a TCP connection established between the sender process port U and the receiver process port L. This single full duplex connection is used as the transmission channel. This protocol is assigned the service port 25 (31 octal), that is L=25. "
ref: RFC 821

About Trin00:

"The Trin00 DDoS zombie tool has been ported to the Windows 32-bit platform. DDoS attack programs, including Trin00, were historically Unix programs designed to launch DDoS attacks from high-performance Unix servers (e.g., Unix servers were used in the DDoS attacks made upon Yahoo, Amazon.com and E*Trade a few months ago). This Trojan could be more dangerous than the recent spate of DDoS attacks because of the vast number of PCs as compared to Unix and Linux machines.
This Trojan tool gives hackers the ability to launch a coordinated DDoS attack on a company from Windows-based PCs and servers. The Trin00 "zombie" program can be delivered by e-mail like other Trojans and remote access tools.

Once delivered and executed this Trojan, like other remote access tools, becomes resident in memory where it waits for the master client program to communicate with it and gain control of the system. It opens port number "34555" and listens for specific commands from the client program.
Any Trin00 client that knows the IP address of the computer where this server Trojan is installed can gain access to the computer.
Trin00 copies itself into the system directory as "service.exe" and loads from the registry at this key:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run "
Ref: http://www.finjan.com/attack_release_detail.cfm?attack_release_id=31

And a little bit more information:

The program listens on UDP port 34555 for trinoo style "ping" packets and responds with trinoo style "PONG" packets to UDP port 35555. This activity seemed to precede a UDP flooding incident. Blocking port 34555 seemed to block further incidents. As reported elsewhere, unlike unix trinoo daemons, this program does not send a "hello" packet back to the compiled in master. Instead, commands to the daemon include a password.
You may be able to reduce risk by blocking incoming packets destined for UDP port 34555. Installing router ACLs that log packets destined for that port will show attempts to use the compromised machines.
Ref: http://www.jmu.edu/info-security/engineering/issues/wintrino.htm

Analysis:

There is no stimulus for the packets seen in the logs. This would imply that the sender process is port 25 and the receiving process is 34555, or 35555, which does not follow the SMTP standard. It is likely that the stimulus for these packets was not logged by SNORT and that this is normal traffic. This is most likely the response to an initial connect for mail transfer from the target, my.net.253.24. This then would be a false positive for the Windows TRINOO trojan. There are several bits of information that would help with this analysis:

1) Is my.net.253.24 a mail server? Is it an SMTP mail client? Is it a very busy system? What type of mail system is this?

Establishing the function of the host helps in determining if the traffic is appropriate. Establishing how busy it is, if it is a mail server or client, what product, sendmail, Exchange, Lotusnotes, might help to explain the high destination port numbers seen.

2) Is outbound mail logging enabled for SNORT? What is the filter used for this detect? What other filters are in place?

This is important because we appear to have unsolicited traffic. Knowing the filters used can help in determining what traffic we are seeing and may help us to tune it to remove false positives.

3) Are these TCP packets or UDP?

The Windows Trin00 Trojan listens on port 34555 using UDP. It transmits on port 35555, again using UDP. If these are TCP packets then this is a false detect.

4) What is the content of the packets we are seeing?

If this is a valid mail server response, then these packets should contain response codes and possibly the actual mail transfer. In other words, if it is mail, then we can find out easy enough by looking at the payload. If this is proper traffic, we won't see a Helo or Ehlo in the first connection attempt logged because this is a response from a mail server but we should see an appropriate response to one of the mail commands coming back from the source address.

In depth analysis is not possible because of the scant amount of information available in these alert logs. This appears to be a false positive probe for the Trin00 daemon on my.net.253.24. I considered the use of source port 25 as a possible attempt to bypass firewall scrutiny because this administrator allows SMTP traffic into their net through their packet filtering router. Trin00 allows any source port to be used by the master client program to communicate with the zombie program. This scenario would fit with the first sampling of connect attempts to port 34555. However the second group of packets are directed at port 35555 which is not the listening port used for the Windows Trin00 Trojan.

I believe that outbound mail logging by SNORT is disabled, possibly for confidentiality of the users, or to minimize the amount of information the IDS analyst has to wade through. If we were seeing responses to these packets from the server with appropriate payload, then this would be a false detect. Even so we can determine this by looking at the payload of these packets to see the exact type of traffic. RFC 821 explains this in detail.

History: I have no history on these two individuals however, there are a number of examples of this type of access from varying source addresses using source port 25.

Technique: Fairly rapid activity with some notable delays between some of the attempts.

Intent: SMTP mail.

Severity/Criticality: Role of the node is unknown. 3

Severity/Lethality: 3 This is most likely mail.

System Countermeasures: Unknown. 3

Network Countermeasures: There are no responses. 4
Severity = (3+3) – (3+4) = **-1 This is a false positive**.

Recommendations:

1. Check to see if this is actually UDP traffic.

2. Examine the payload of these packets for valid email commands and in the first packet grouping directed at 34555 check for the Trin00 password being transmitted in the packet detail. If this is not the case, and we see mail commands then we are done. Otherwise…

3. The Trojan listens on udp port 34555, and will respond to pings on udp port 35555. The password is "[]..Ks" (without the quotes). Therefore the following will detect it:
4. Set up a netcat listener: nc -u -n -l -p 35555 -v -w 100
5. Send a trinoo ping: echo 'png []..Ks l44' | nc -u -n -v -w 3 192.168.1.5 34555
6. The listener will display PONG if a trinoo daemon is listening.
7. This will kill it: echo 'd1e []..Ks l44' | nc -u -n -v -w 3 192.168.1.5 34555 After it is killed, the udp port may still be bound until a reboot, at least on Windows 95/98. Subsequent trinoo pings will return an ICMP destination unreachable/port unreachable if it is down.

Ref: http://packetstorm.securify.com/distributed/razor.wintrinoo.txt

Detect #7

Overview:
This is another daily Snort alert log from the same organization as the proceeding. The position of the sensor is unknown. The role of the system targeted is unknown.

Trace #7 – Snort Alert Log

```
1     [**] GIAC 08-feb-2000 [**]
      03/23-18:01:51.571941 195.11.50.204:4274 -> MY.NET.60.14:1347

2     [**] GIAC 08-feb-2000 [**]
      03/23-18:06:06.575952 195.11.50.204:27015 -> MY.NET.60.14:27005
3     [**] GIAC 08-feb-2000 [**]
      03/23-18:06:06.922227 195.11.50.204:27975 -> MY.NET.60.14:27960

4     [**] GIAC 08-feb-2000 [**]
      03/23-18:09:53.166443 195.11.50.204:53 -> MY.NET.60.14:723
5     [**] GIAC 08-feb-2000 [**]
      03/23-18:09:53.434453 195.11.50.204:3894 -> MY.NET.60.14:25
6     [**] GIAC 08-feb-2000 [**]
      03/23-18:09:54.294051 195.11.50.204:53 -> MY.NET.60.14:53
7     [**] GIAC 08-feb-2000 [**]
      03/23-18:09:55.219224 195.11.50.204:27025 -> MY.NET.60.14:27005

8     [**] GIAC 08-feb-2000 [**]
      03/23-18:09:59.968734 195.11.50.204:7744 -> MY.NET.60.14:1045
```

Analysis:

This is a port scan of my.net.60.14 by 195.11.50.204. The scanner hits the following ports:

1347 – Ephemeral port, unknown but possible trojan or malicious code

27005 – Unknown port, possible trojan, possible gaming port

27960 – Unknown Port, possible Trojan, Quake games and some of their derivatives fall in this range 27910 – 27961. The source port does not coincide with this (27975) as it falls outside the range of these gaming ports.

723 – Unassigned well known port, Unknown but possible Trojan.

25 – SMTP

53 – DNS

27005 – Unknown port, again.

1045 – Unassigned, Ephemeral port

Active Targeting: Yes.

History: I have no history on this source address. At some previous time this individual may have mapped out this network. He knows who he is going after because he doesn't scan any other systems or ports; at least not on this day.

Technique: This is a rather slow scan. The duration of this event is 8 minutes and 8 seconds. A single probe comes in at 18:01, followed 4 minutes and 15 seconds later by two quick port scans. Three minutes and 45 seconds later, beginning at 18:09:53, four probes are launched, and finally, four seconds later the final probe comes in.

For some reason port 27005 is hit twice. Maybe the perpetrator didn't like the answer the first time.

This is quite likely a scripted attack as I am not sure that NMAP or any of the other scanners can delay their packets in this random a fashion. NMAP does have the functionality to change timeout values, scan delays, and can script the ports in this fashion but it doesn't look like this is the case. The intervals between the attempts are too random. These are numerous options for nmap that allow for scripting of activity. The author has provided a manual at http://www.insecure.org/nmap/nmap_manpage.html

The source ports are randomized throughout this scan. It is instructive to note that the source node miraculously turns itself into a DNS server when probing port 53. Port 53 to port 53 communication is seen during DNS server to DNS server queries etc.

Port 53 is used again as the source port when a scan of port 723 is attempted. It is doubtful that anything is on this port however a trojan is possible.

Intent: The perpetrator is attempting to map services and or Trojans on my.net.60.14.

Severity/Criticality: I do not know the functionality of this node. It could be anything. I will assume a common server and not a critical system, **3** for the purposes of this detect.

Severity/Lethality: This is a port scan. The actual type of scan attempted cannot be defined from the detect. Lethality of **4** mostly because of the attempt on port 53 and port 25.

System Countermeasures: Unknown. I will assume that this system is not completely patched and wrappers and other security measures are not in place. **3**

Network Countermeasures: Unknown. There is a packet filtering router at this site and they use a capable IDS but I do not know where this particular sensor is receiving it's packet stream from; in the DMZ, screened subnet or Intranet. I have seen some examples from this site where protocols that should not be allowed into the network are getting through. With that said, there is no indication of any response **3**
Severity = (3+4) – (3+3) = **1 Low severity.**

Recommendations: If this sensor is located in the dmz, then no actions are indicated. The sensor does not pick up any responses from the host and the packet filtering router is likely dropping these packets. I can't make any concrete recommendations with the overall lack of knowledge I have on the site setup. Best thing to do is scan the host and determine for your self what ports are open on the host.

Detect # 8

Overview:
This is a trace pulled off of the SANS GIAC page.

Trace #8

```
Apr 7 10:51:28 144.92.98.76:3671 -> x.y.z.101:53 UDP
Apr 7 10:51:28 144.92.98.76:3708 -> x.y.z.116:53 UDP
Apr 7 10:51:28 144.92.98.76:3714 -> x.y.z.118:53 UDP
Apr 7 10:51:29 144.92.98.76:3736 -> x.y.z.128:53 UDP
Apr 7 10:51:29 144.92.98.76:3752 -> x.y.z.135:53 UDP
Apr 7 10:51:29 144.92.98.76:3768 -> x.y.z.142:53 UDP

Apr 7 10:51:55 144.92.98.76:3829 -> x.y.z.161:53 UDP
Apr 7 10:51:55 144.92.98.76:3835 -> x.y.z.164:53 UDP
Apr 7 10:51:55 144.92.98.76:3885 -> x.y.z.186:53 UDP
Apr 7 10:51:55 144.92.98.76:3904 -> x.y.z.195:53 UDP
Apr 7 10:51:55 144.92.98.76:3920 -> x.y.z.200:53 UDP
Apr 7 10:51:55 144.92.98.76:3922 -> x.y.z.201:53 UDP
Apr 7 10:51:55 144.92.98.76:3966 -> x.y.z.217:53 UDP

Apr 7 10:52:50 144.92.98.76:4132 -> x.y.z.52:53 UDP
Apr 7 10:52:50 144.92.98.76:4144 -> x.y.z.58:53 UDP
Apr 7 10:52:50 144.92.98.76:4166 -> x.y.z.67:53 UDP
Apr 7 10:52:50 144.92.98.76:4192 -> x.y.z.79:53 UDP
Apr 7 10:52:50 144.92.98.76:4204 -> x.y.z.83:53 UDP
Apr 7 10:52:50 144.92.98.76:4238 -> x.y.z.97:53 UDP

Apr 7 10:52:54 144.92.98.76:4226 -> x.y.z.91:53 UDP
```

Analysis:

This is a host scan for DNS services. If the port is not open, the target node will respond with an ICMP port unreachable message. If the port is open, the response will depend on the nature of the payload on the incoming packet. Most likely these will be reverse DNS queries.

The system that is running the scanner is quite busy, as evidenced by the ephemeral source ports. I tend to believe that the system is busy because the ports are advancing by leaps and bounds, and not incrementing by a single count for each node scanned. This is with the exception of the last two entries where the source port drops from 4238 to 4226.

UDP packets are notorious for getting lost in the shuffle and dropped packets are a possibility here. I looked at this from two angles. Are we losing packets and this is why not all hosts are being scanned? And if we assumed that this was not a busy system performing the scan, then would the source ports numbers we are seeing, coincide with the missed hosts in the scan. For example:

```
Apr 7 10:51:28 144.92.98.76:3671 -> x.y.z.101:53 UDP
Apr 7 10:51:28 144.92.98.76:3708 -> x.y.z.116:53 UDP
```

If packet loss was the culprit, there would be an increment of 15 source ports between these two scans – corresponding with scan attempts on x.y.z.102 through to 116 – give or take a few to allow for other activity on the source node. The source port jumps 37 in this instance which is well over double what we would expect. This is true in all of the cases. This is a busy host performing the scans and the large gaps between nodes scanned is unlikely to be due to simple packet loss.

This is a scanning tool or script that is quite rapid. There are a few exceptions in the speed of the scan and I have placed spaces where these gaps occur. It is interesting to note that there are relatively large jumps in the source port where these timing gaps occur. Further evidence that the system is quite busy and not particularly fast when stressed.

This attacker appears to have already mapped this subnet based on the nodes chosen for the Host scan. Assuming that mapping has already occurred, and that this is now a search for vulnerabilities, the next phase would be the attack.

Active Targeting: Yes. Targeting specific hosts and a specific service.

History: I have no history on this source address, however the nature of the scan suggests that there is a history there. Mapping is already done.

Technique: A rapid scan using a script or tool. Scanning was performed on a busy system.

Intent: The perpetrator is most likely attempting to perform reverse DNS queries on a list of known hosts that he or she had previously mapped. At minimum they can tell if a node is actually up and running. If it is and the node responds they gain BIND version information as well as host names that may give clues as to the host OS and it's function in the network. This can then help them to fine tune their attacks.

Severity/Criticality: I do not know the functionality of these nodes. I will assume that these are not critical nodes for this analysis, however this was a fairly discriminate attack. **4**

Severity/Lethality: The actual type of scan attempted cannot be 100% defined from the detect. I give it a lethality of **5** mostly because of the attempt on port 53 and the fact that it appears they have already accomplished mapping of the network.

System Countermeasures: Unknown. I will assume that these systems are not completely patched and wrappers and other security measures are not in place. **3**

Network Countermeasures: Unknown. There are no responses evident so the firewall is most likely dropping these packets. **4**

Severity = (3+5) – (3+4) = **1**

Recommendations:

Assuming that there was no response to these probes, and that these packets were seen in the DMZ, which seems to be the case, no action is required. I would copy the source address down in my little black book and watch for this individual in the future.

However…

The scary part about this is that they appear to know the network well. If the targeted hosts are actually existing nodes, and this is not arbitrary targeting, then they are getting to know this site quite well. If each of the hosts scanned has a named service running then this gets even more scary and I would want packet detail including payload to get an idea of what, if anything, they were attempting to send to these nodes. I'd also be tempted to

look back through my logs for any indication of successful mapping from that source address to find out what technique was used and if it could be blocked in the future.

With that said, there is no evidence of any response from the targeted hosts, so the firewall or filtering device is doing it's job.

Detect # 9

Overview:
This is a Snort detect that I found on the SANS GIAC web site. The detect is triggered because it has passed a threshold number of packets directed to a single target node.

Trace #9 – Snort detect
Speed Era Networks, Santa Clara CA, USA
Apr 6 13:07:20 dns2 snort[5950]: spp_portscan:
PORTSCAN DETECTED from 204.176.88.5
Apr 6 13:07:26 dns2 snort[5950]: spp_portscan:
portscan status from 204.176.88.5: **5 connections across 1 hosts**: TCP(0), UDP(5)
Apr 6 13:07:32 dns2 snort[5950]: spp_portscan:

End of portscan from 204.176.88.5 -------

```
Apr 6 13:07:18 204.176.88.5:54605 -> a.b.c.66:33651 UDP
Apr 6 13:07:19 204.176.88.5:54605 -> a.b.c.66:33652 UDP
Apr 6 13:07:19 204.176.88.5:54605 -> a.b.c.66:33653 UDP
Apr 6 13:07:20 204.176.88.5:54605 -> a.b.c.66:33654 UDP
Apr 6 13:07:21 204.176.88.5:54605 -> a.b.c.66:33655 UDP
```

```
04/16/00 10:33:27 IP block 204.176.0.0
Trying 204.176.0.0 at ARIN
Trying 204.176.0 at ARIN
UUNET Technologies, Inc. (NETBLK-UUNETCBLK176-179) UUNETCBLK176-179
        204.176.0.0 - 204.179.255.0
MaineStreet Communications, Inc. (NET-MAINE-NET) MAINE-NET    204.176.0.0
```

Analysis:

This scan takes 4 seconds to probe 5 ports in ascending order; 33651 to 33655. This is not a real paranoid stealthy attempt but a slow 'polite' form of knocking on the doors. The source address and source port remain constant. If the Time to Live values were available we could tell a bit more about what is being attempted here. If the TTL value was 1 then this would almost certainly be a traceroute of some kind. It would be very rare indeed to have a TTL value of 1 on a valid connection.

The time that this occurs is important to look at and can provide a clue to what the activity is. This is occurring in the middle of the day, right after people get back from lunch—a good time for surfing. If this was malevolent activity it would be less likely to be performed at this time. Wait until night after everyone has gone home, inthe wee hours, to try this kind of stuff. This of course is not conclusive but is just one of the clues.

The source ports remain constant throughout this trace.

I looked at the possibility of a traceroute attempt in a little more detail to see if this is really what is going on. The incrementing destination port number is instructional, however 33651 – 33655 are not in the range of the normal traceroute ports (33434 - 33600). Load balancers can do some strange things with the destination port numbers but I am not sure if this is the case here, yet. There are a myriad of trace route tools available for download, or that

you can run off of a web site, that can use arbitrary or user assigned ports so a traceroute is not out of the question. Traceroute itself has a number of options of interest. In particular the –p switch:

"Traceroute options that can be entered along with the hostname:
[-nv] [-m max_ttl] [-p port] [-q queries] [-t tos] [-w wait] host [data size]
-n print addresses numerically rather than symbolically
-v verbose output
-p port set base UDP port used in probes (default 33434)
-q queries set number of probes per ttl (default 3)
-t tos set type-of-service in probe packets
-w wait set time in seconds to wait for a response (default 3) host hostname or IP address to trace the route to
datasize data size in bytes of each probe packet (default 38)"

So, if this is a traceroute and the trace started with a destination port of 33434, with 5 attempts per intermediate site (I use 5 attempts based on the five attempts made on the target node in the detect):

33650 – 33434 = 215

215 / 5 = 43 hops assuming that 5 attempts are made at each intermediate router.

Just for fun, I attempted a traceroute to the source address from various locations. From my location in Western Canada, I came up with only 12 hops using neotrace. The initiating site is down in California. I'm not sure where this detect comes from, but I believe it is one of the .edu sites in the States, (based on some comments from the GIAC handlers), and would not require 40 plus hops. This is obviously not conclusive but provides a clue, 40 hops is a heck of a lot.

I went to http://as5388.net/cgi-bin/lg.pl and attempted a trace route from Leeds, England and only got 19 hops.

There are no Trojans listed for these destination ports and the IANA port assignments do not correlate either.

It would help to know the function of the target node. That might help in determining if there is any possibility of load balancing being involved. If the target is a DNS server or caching appliance then I would categorically state this is a load balancer, **especially** if the packets were coming in with a TTL of 1.

Active Targeting: Yes.

History: There is no history available for this source address. I was unable to resolve the ip address any further than that shown by Sam Spade using any tool. Dig queries did not provide any further enlightenment.

Intent: Load balancer or activity generated by one of the many tracing tools that can be found all over the Internet. www.webattack.com has some interesting trace route, pingers, and scanning utilities.

Technique: Slow scanning from an appliance, a tool or script. Again this is quite possibly load balancer activity. A load balancer is much more likely to have sporadic timing like this, than any of the tools that are available on the Internet. The going thing with the various tracer programs is that they all want to be faster with multi-threading capability and bells and whistles. A load balancer on the other hand may have to contend with hundreds of different connection attempts at a time and is more likely to exhibit this kind of behavior, especially during prime time. This would be particularly true of co-located facilities, where multiple web servers from multiple Companies are co-located on sites across the Globe.

Severity/Criticality: 3 Unknown node type. I assume that this is a server of some description. If this is load balancing then this should be a caching appliance or DNS server.

Severity/Lethality: 2 This is mapping of a host at worst and hitting on some unlikely ports as well.

System Countermeasures: 3 Unknown. I will assume that it is not hardened and wrappers and other defensive techniques are not in use.

Network Countermeasures: 4 There is no evidence of any ICMP responses so the firewall or packet filtering device for this site is working.

Severity = (3 + 2) – (3 + 4) = **-2 very low severity.**

Detect # 10

Overview:

I'm sorry, but yes, this is yet another detect from the GIAC site. This one was interesting to me because of the targeted proxy ports, including the squid proxy. I don't know if this is the entire detect or just a subset of one.

Trace #10

```
16:55:35.440651 1Cust219.tnt1.bryan.oh.da.uu.net.1865 > @.home.com.8080: S 12492586:12492586(0) win
8192 (DF)
16:55:35.465692 1Cust219.tnt1.bryan.oh.da.uu.net.1866 > @.home.com.www: S 12492589:12492589(0) win
8192 (DF)
16:55:35.484070 1Cust219.tnt1.bryan.oh.da.uu.net.1868 > @.home.com.3128: S 12492595:12492595(0) win
8192 (DF)
16:55:35.484222 1Cust219.tnt1.bryan.oh.da.uu.net.1870 > @.home.com.8050: S 12492601:12492601(0) win
8192 (DF)
16:55:35.484367 1Cust219.tnt1.bryan.oh.da.uu.net.1869 > @.home.com.8002: S 12492598:12492598(0) win
8192 (DF)

16:55:36.664311 1Cust219.tnt1.bryan.oh.da.uu.net.1865 > @.home.com.8080: S 12492586:12492586(0) win
8192 (DF)
16:55:36.666792 1Cust219.tnt1.bryan.oh.da.uu.net.1868 > @.home.com.3128: S 12492595:12492595(0) win
8192 (DF)
16:55:36.706460 1Cust219.tnt1.bryan.oh.da.uu.net.1869 > @.home.com.8002: S 12492598:12492598(0) win
8192 (DF)
16:55:36.758762 1Cust219.tnt1.bryan.oh.da.uu.net.1870 > @.home.com.8050: S 12492601:12492601(0) win
8192 (DF)

16:55:37.625224 1Cust219.tnt1.bryan.oh.da.uu.net.1865 > @.home.com.8080: S 12492586:12492586(0) win
8192 (DF)
16:55:37.939332 1Cust219.tnt1.bryan.oh.da.uu.net.1868 > @.home.com.3128: S 12492595:12492595(0) win
8192 (DF)
16:55:37.949391 1Cust219.tnt1.bryan.oh.da.uu.net.1869 > @.home.com.8002: S 12492598:12492598(0) win
8192 (DF)
16:55:37.985345 1Cust219.tnt1.bryan.oh.da.uu.net.1870 > @.home.com.8050: S 12492601:12492601(0) win
8192 (DF)

16:55:38.396403 1Cust219.tnt1.bryan.oh.da.uu.net.1866 > @.home.com.www: S 12492589:12492589(0) win
8192 (DF)
16:55:38.786750 1Cust219.tnt1.bryan.oh.da.uu.net.1865 > @.home.com.8080: S 12492586:12492586(0) win
8192 (DF)
```

Analysis:

This is a scan for proxies using active open attempts. The scanner searches for proxies on port 80, 8080, as well as the Squid Proxy on port 3128. In addition to these ports the scanner also attempts connections to more obscure proxy ports, 8002 and 8050.

I doubt very much that the individual is looking for a Database application for mapping and terrain analysis as listed in the IANA port assignments.

8002/tcp Teradata ORDBMS teradataordbms
8002/udp Teradata ORDBMS teradataordbms

In an attempt to find the 8002 and 8050 port usage, I went to an anonymiser site.
http://www.magusnet.com/proxy.html lists the following ports as those used for http access. Each of these is a proxy port:

HTTP: **80**,81,82,83,84,85,**3128**,**8000-8009**,**8080**-8090,8888,10080,34567.

I searched for port 8002 proxies using http://allfreeweb.hypermart.net/proxy/index.html . This resulted in two hits. Both of these are free http proxies that you can use to anonymise your connections.
admin.sbo.wjcc.k12.va.us 8002 Location: **USA** Domain:
us url.cms.k12.nc.us 8002 Location: **USA** Domain: **us**

A search for 8050 using the same search engine did not reveal any known sites providing this proxy service. However, I did find one at, http://proxylist.hypermart.net/list1.htm, 202.35.224.200:8050

So what does this tell us? Nothing conclusive, however all of the ports attempted in this scan can and are used as proxies elsewhere.

The pattern is very interesting. We start with 5 scans for all five ports of interest. In the second and third scan port 80 is ignored and the scan concentrates on port 8080, 3128, 8002, and 8050, in that order. The final syn scan revisits port 80 and 8080 while ignoring the other ports.

Each of the four scan segments is very quick with a gap of 0.5 to 1 second between attempts.

The sequence number cycles. In all cases we see the same sequence number, source port, destination port number combination. For example, all attempts at the http proxy port 8080 use a source port of 1865 and a sequence number of 12492586. The source port being the same for each connect attempt to a single target port is anomalous as is the sequence number. These are crafted packets.

So why is this individual after proxies?

"If a protocol or service can be exploited by a network scanner such that the service can make arbitrary network connections, then the protocol can be used for port scanning. Some proxy servers and most FTP daemons can be used to conduct port scans. The classic example is the FTP Port Scanner in which a surrogate FTP server is used to make many network connections to a target system. The FTP protocol allows for the client to specify which IP address and port the server should send data to. Information returned by the FTP server can be used to identify open or closed ports on the target system."
Ref: How to handle and Identify network probes, Ron Gula

Active Targeting: Yes.

History: There is no history available for this source address. If you look at the second segment you could say there is a history, the same guy did this to us a second ago. 8)

Intent: Our perpetrator is looking for proxy services. Usually the reason for doing this is a search for sites that can be used to hide a cracker's tracks. Once a site like this is compromised the cracker can anonymize mail connections etc… Though these connections can be tracked down, multiple proxies can be used to make it very time consuming and difficult to track the culprit.

Technique: Uses a tool or script that creates crafted packets. The ordering of the scan is unique along with the number of attempts.

Severity/Criticality: 3 Assuming a web server.

Severity/Lethality: 3 The lethality of this attack really depends on what the function of this node is, what it's OS is, the flavor of it's applications and where this node sits within the network. If it's a Windows NT Server running IIS 4.0 and resides on the Customer LAN then this should be taken very seriously. If this node resides in a screened subnet and has no trust relationships with the Internal network, then the lethality drops significantly.

System Countermeasures: 3 Unknown countermeasures in place on the system.

Network Countermeasures: 3 Unknown, however there is no evidence of a response so the firewall or packet filtering router is most likely doing it's job.

Severity = (3 + 3) – (3 + 3) = **0 very low severity.**

Last Updated: July 27th, 2017

Upcoming Training

SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201709,	Sep 11, 2017 - Oct 18, 2017	vLive
Baltimore Fall 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Scottsdale SEC503	Scottsdale, AZ	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced