GCIA PRACTICAL

Intrusion detection in depth

Royans K Tharakan
rkt (at) poboxes.com

Version 3.0 (revised Friday, February 15, 2002)

CONTENTS

INTRODUCTION

Active Targeting
I could not find a good definition for “Active Targeting”. The way I am defining this is that a host would be called an “active target” if there is evidence of multiple reconnaissance probes and attacks indicating that “someone” is attacking a particular host and its not just an automated “worm” or “script” doing random attacks. OS fingerprinting is extremely strong indication of an “Active targeting”.

ASSIGNMENT 1 - State of intrusion detection

Development of Stealth Scans.

INTRODUCTION

During the course of collecting network detects for this assignment, I saw one of the most bizarre network scans on my home network. The scan apparently originated from (or at least seemed to have originated from) multiple hosts (as many as 5 confirmed source). This bizarre attack lead me to dig up more information on how stealth attacks improved over the years. [Refrence: Assignment 2: Network Detect 5]

Intrusion detection always succeeds in catching up with the latest techniques in intrusion. The very first intrusion detection systems were based on syslog servers, which used syslog dumps to locate break-ins. Over the years Firewalls and other sophisticated intrusion detection servers have taken over this role of detecting break-ins. However, what hasn’t changed much is the fact that each of these detection mechanism still rely on numbers to cross a certain threshold for the alerts to be sent out.

DISECTION OF A SCAN

I’ve read a lot of papers on this topic, but first one which I’ve read on this topic was from Fyodor@insecure.org http://www.insecure.org/nmap/fin. [Ref 25]

TCP Scan: 3 Way Handshake

One of the first steps of information gathering about a target host before attack is to find out what ports a host is listening on. In the good old days there used to be scripts which used to telnet to port 25 (sendmail) or 23(telnet) to find out what the banner says. Probability of such a scan to be detected would be low if only one host on a network is attacked and it would be even lower if both of these ports were actually used regularly by the server.

To create each open connection the attacker would have to complete a “3 way handshake”.

ATTACKER → SYN → TARGET

TARGET → SYN/ACK → ATTACKER

ATTACKER → ACK/ACK → TARGET

But if this were a script which was doing a network wide scan, opening up TCP/(3 way handshake) connections to all hosts on the network would be extremely noisy and annoying for the analyst.

Counter measure for such a scan is to count the number of open connections from all hosts. If any client is connecting to more than a few servers at the same time or within a given amount of time, this would be an intrusion alert.

SYN Scan: Incomplete Handshake

This mechanism is the improvement of the ‘3 way Handshake’. As you would notice in the previous example, an open TCP connection is only complete when the third step of the handshake is completed. The second step of a successful connection is always a SYN/ACK. But had the port been closed it would have sent back an “RST” instead.

ATTACKER → SYN → TARGET

TARGET → RST → ATTACKER

Counter measure: Most firewalls today detect a SYN probe, and sufficient number of SYN probes can isolate an intruder.

Random SYN Scan: Random PortScans

Incomplete Handshake is an effective way of working under the intrusion radar. However this will fail when the same sender sends too many SYN probes to the same target host/network at the same time. By introducing sufficient delays between scans and by Randomizing PortScans over a large Class B network, some radars can be avoided. Some of the most popular Internet worms use some form of Random PortScans mechanism in their scripts today. The number of hosts a worm could infect without raising too many alarms is proportional to the quality of Randomizing algorithm used. Counter measure: Same as SYN Scan.

ICMP Usage in scanning

ICMP has a long history for scanning. Traditionally only ICMP echo/reply were used to collect this information. A simple script in a loop running ping could quickly gather a list of active hosts in a network. Interestingly UDP packets to closed ports could also generate ICMP port unreachable messages. This can be used to find live hosts and ports. Ofir Arkin & Fyodor Yarochkin did great write-up

"ICMP Usage in Scanning" which talks about the other ways of using ICMP for scanning.

FIN Scan: Sneaking under the Firewall

Lots of routers and Firewalls today are careful about letting SYNs through the network. FINs however could evade firewalls since they are only looking for SYNs. FINs sent to closed port usually come back with a RST. Those sent to open ports are ignored. According to Fyodor (nmap), Microsoft TCP stack sends RST in either case.
Countermeasures: Use statefull firewalls

Multiple Flags: Confusing the Firewall

Some firewalls recognize a stray FIN packets when it sees one. They keep track of sessions. But then there are enough firewalls which don't properly understand flag combinations. For example SYN+FIN together could trigger an "OK" from the firewall and let it through the firewall.
Countermeasures: Use popular Firewalls which would have most of these issues taken care of. Install snort or other similar IDS to detect packet anomalies.

Using Decoys while scanning

One of the problems as I explained before is the keep the attacker under the radar so that the intrusion could go unnoticed. The other side of the problem is that if the attack cannot go unnoticed, the attacker needs to make sure he cannot be isolated. To do a scan under such circumstances either the attacker would have to start the scan while other people are scanning at the same time, or he could at least make the scan look like its coming from multiple hosts. Decoy scan can be more effective if it originates from real active IP addresses, and even more if the packets from different IPs have significantly different TCP stack characteristics.
Countermeasures: Use offline filtering/analysis to do closer analysis.

OS Fingerprinting

Before OS fingerprinting became popular the standard way of retrieving OS info was using banners sent by sendmail or login from the server.

```
[rkt@torque] ~$ telnet hostname2 25
Trying 10.32.2.21...
Connected to hostname2.
Escape character is '^]'.
220 hostname.somecompany.com ESMTP Sendmail 8.10.2+Sun/8.10.2; Wed, 13 Feb 2002 14:02:12
-0800 (PST)
```

For a long time System Administrators considered themselves secure enough by just removing or replacing standard OS banners. Then some other folks figured out that Apache and a few other web servers do the same thing.

OS fingerprinting makes use of the dissimilarities between different TCP Stack implementations. Though most of the OS do follow the Internet Standards, a lot has been left out by the standards group for the individual vendors to implement on their own. To get a fingerprint of a particular host the fingerprinting tool sends a series of crafted packets as a stimulus and waits for a response.

DISTRIBUTED SCANNING

The standard mechanism of detecting break-ins, as I mentioned before was to match numbers. If a particular IP sent more than X number of packets which looked suspicious then alarms would go off, and if this is a IDS module in a router or firewall, then probably that source address would be blocked for some time. [Ref 24] [Ref 23]

To keep the number of probes per IP low but still get all the information it is necessary to do a coordinated probe using different sources. Coordinated scans reduce the risk of getting detected.

- ***Faster Scan***
  Breaking up the targets into multiple clusters, each of which is scanned by a different node could dramatically speed up a recon.
- ***Can generate physical network map to the target***
  TTLs embedded in packets especially those of ICMP reply can help generate physical map of the network which is being probed. A traceroute would be amazingly informative. Such information is considered very sensitive and can be exploited by DDoS tools to initiate a DoS.
- ***Can help predict Sequence numbers***
  Predicting Sequence numbers and using it to hijack connections is one of the oldest tricks in the book. Starting with the 'Mitnick attack' sequence number predictability has been a very crutial part of OS security. Connecting multiple times from different hosts can help predict Sequence numbers the same way Kevin Mitnick Predicted in his famous hack. Since the connections didn't come from the same host, without a trained pair of eyes this will go under the radar for sure.
- ***Can help identify the OS***
  It usually requires more than 1 packet to confidently identify a particular OS. And sending more than one crafted packets can at time be sufficient to send an alarm. Sending packets from multiple sources can infact let this happen under the radar.

ARICHITECTURE FOR DISTRIBUTED SCANNING

Most DDoS tools like stacheldraht and trinoo use tree like architecture for DoS attacks. For distributed scanning however, each of the individual nodes need certain characteristics that might be absent for stacheldraht/trinoo nodes.

- Ability to communicate with master node using conventional/covert channels
- Ability to send stimulus to any server
- Ability to receive response from any server
- Ability to craft packets using RAW IP. This requires root privileges
- Time needs to be synchronized for best results

There are typically two kinds of remote agents which an attacker could use.

Multi-sender Multi-receiever with control node

This is perhaps the most common architecture which could be used for distributed scanning. The control node makes all plans and informs each of the child nodes to gather a set of information on a set of IPs. Each of the receivers later process the results and send it to the master control node.

Single-sender Multiple Listener with control node

Its possible to send probes from a single host with different source addresses, each of which belong to hosts on which the attackers run their own listener to wait for the packets. This is a better design where most activity happens at the master node. The child nodes only duty is to listen to the reponses and forward it to the master node.

Number of nodes for a typical Distributed scanning network

It is important to understand that Distributed scan done using scripts would still generate too much noise. The real power of Distributed scan would be when its tarteged at specific hosts/network about which some recon data is already available.

Unlike a DDoS network where higher numbers are favorable, for scanning one doesn't need that many. Depending on the kind of recon required it would be helpful to have nodes distributed all over the world.

For a port scan of a node of 10 servers where one is just looking for 111 (statd) and 22 (ssh), it would be ideal to have a group of 20 source addresses so that each of the probes could go under the radar.

To find a networks border routers, multiple traceroutes from as many as 10 to 15 hosts would be helpful. This ofcourse depends on how big the target network is.

To do OS fingerprinting, typically it requires anything between 4 to 10 different packets depending on what kind of OS is on the other end, and what tool you are using to do the analysis. Nmap uses a set of 8 different probes to get this information. Having a different node to do each of this test would be ideal.

How to detect a Distributed Reconnaissance

Most intrusion alarms won't go off in this kind of a recon activity, because the same IP doesn't send multiple probes. To recognize a distributed recon, its important to have a complete picture of network activity. The Intrusion detection model has to move away from source IP based counters to global counters. If multiple IDS are installed, it needs to talk to each other for activity happening at different parts of network.

By the time someone reads this document, I'm sure there are better ways of information gathering will come up. I saw a note about "packet bouncing" which is one of the most advanced form of information retrieval yet. In this attack, the attacker send multiple packets with wrong source address to the victim and sends a one a few probes to the victim. The change in IP Ids could theoretically indicate how active this server is. [Ref 26] [Second Order Effects]

Hopefully better ways of detecting these techniques will come out sometime soon.

ASSIGNMENT 2 – NETWORK DETECTS

NETWORK DETECT 1: ANOMALOUS PACKET

```
[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 4.3400 [**]
12/28-08:06:06.702394 XXX.XXX.X.XX:4513 -> XXX.XX.XXX.XXX:9274
TCP TTL:115 TOS:0x0 ID:14182 IpLen:20 DgmLen:48 DF
******S* Seq: 0x201AC3D4  Ack: 0x0  Win: 0x2238  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

[**] [104:1:1] spp_anomsensor: Anomaly threshold exceeded: 4.2908 [**]
12/28-08:06:09.511201 XXX.XXX.X.XX:4513 -> XXX.XX.XXX.XXX:9274
TCP TTL:115 TOS:0x0 ID:14500 IpLen:20 DgmLen:48 DF
******S* Seq: 0x201AC3D4  Ack: 0x0  Win: 0x2238  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
```

SOURCE

http://www.securityfocus.com/archive/75/247527

TIME

Dec 28th 2001

TOOL

Snort spade module. [Ref 5].
This analogous packet warning was probably generated because traffic on this port probably never happened before. SPADE which works as a module for snort, generates an anomaly score for each packet which passes it. Packets which come in frequently are allowed through without a warning. However this particular pattern triggered the alarm since it never was probably not seen before.

PROBABLE SOURCE

Unknown. The IP is hashed out.

PROBABILITY OF BEING SPOOFED

This is most probably a real packet which was not spoofed. I base my analysis on the following factors

1. IP ID is changing with packet
2. The source port changes with every destination host
3. There is a 3 second time lag, which could be the standard packet retransmission time.
4. Passive finger printing shows that this packet does fit a finger print profile of Windows95/98/NT as documented in [Ref 4]

5. This packet is probably a stimulus to obtain a response from applications running on this port. If the source was spoofed it would have never got status of this port back making the probe useless (unless there is another listener at the spoofed address).

DESCRIPTION OF THE PROBE

BlackGate WWW Proxy discovery probe

The probe is part of a reconnaissance effort from attacker to gather information on target network.. Port 9274 is relatively unused [Ref 6] [Ref 7] and hence doesn't look like any popular application or Trojan resides on these ports. This probe could have been a OS fingerprinting effort, however, since we see two probes with exactly same info separated by 3 seconds without any indications of packet crafting, I'm forced to believe this probe is specifically looking for a particular unknown Trojan or Application across the network.

On further research a research I found that Matt Scarborough [Ref 8] talks about "BlackGate" Trojan using 9274 as the WWW proxy port.

ATTACH MECHANISM

The subsequent attack depends on this port being open. If the port is open the standard response from an OS would be to send a "Syn/Ack" packet back to the source host. From the reference documents on the net [Ref 8] it looks like this port was hosting a WWW proxy. Having an access to a proxy could be invaluable for the attacker. There could be two outcomes if this probe was successful

1. If this port was open and WWW proxy was enabled, the attacker could have used this host for further HTTP based attacks [for example Unicode attack on IIS) on other hosts.
2. If blackgate has a backdoor, attacker might have been trying to find a list of hosts infected by blackgate so that he could use these hosts for more advanced attacks against other networks.

CORRELATION

Though this probe is extremely rare, there have been reports of this port being used by a Trojan for some time.

Dec 12 2001: http://www.incidents.org/diary.php?id=115

This is another incident which was reported which looked similar to this one

Jun 2001: http://keir.net/attacklist.html

This report talks about a port scan done on blackgate infected systems which show up 9274 as a listening port.

Feb 25 2001: This is an old report which matches the report by keir.net.

http://cert.uni-stuttgart.de/archive/incidents/2001/02/msg00355.html

EVIDENCE OF ACTIVE TARGETING

The probe according to the reference email was going to multiple hosts on his network. We do not have sufficient information to say that his network was an active target. It could be a random network scan.

SEVERITY:

Severity =
(Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality: 1 (It was detected on a sensor.)
Lethality: 1 (No signs of compromise.)
System Countermeasures: 3 (not aware of)
Network Countermeasures: 5 (Was detected by spade module)
Severity = -6 = (1 + 1) – (3 + 5)

DEFENSIVE RECOMENDATAION

We do not have sufficient information on whether the probe was blocked by the firewall. We do know however that two packets with same sequence number were sent to the network, which makes be believe that the SYN/ACK reply was not sent. If the network in question does have a firewall it would be good to make sure that there is a rule to block this port. However since this particular network was attacked, and we don't see too many of these packets in the wild, I'd recommend that a network scan be done to confirm the absence of this particular Trojan in their entire network. Its entirely possible that this is a part of a reconnaissance after a previous automated attack to gather information on infected hosts.

MULTIPLE CHOICE TEST QUESTION

What does the IP ID change in the two packets indicate about the src tool/hosts used ?

Select the wrong answer.

1) This could be spoofed using a packet crafting tool
2) This cannot be spoofed using a packet crafting tool
3) The receiving host generates random IP ID, this question is irrelevant
4) Probably the Source host OS generated the IP headers and it indicates lot of network activity is going on at the source host.

Answer is 2, because this can be spoofed.

NETWORK DETECT 2: Port 53 Scan

```
08:56:28.847738 63.100.168.57.1059 > mynetworkhost.53: S 1583156677:1583156677(0) win
32120 <mss 1460,sackOK,timestamp 26363088 0,nop,wscale 0> (DF)
0x0000   4500 003c faa1 4000 3206 d3d1 3f64 a839        E..<..@.2...?d.9
0x0010   xxxx xxxx 0423 0035 5e5d 0dc5 0000 0000        ..y..#.5^]......
0x0020   a002 7d78 9969 0000 0204 05b4 0402 080a        ..}x.i..........
0x0030   0192 44d0 0000 0000 0103 0300                  ..D.........
08:56:28.848256 mynetworkhost.53 > 63.100.168.57.1059: S 3051693243:3051693243(0) ack
1583156678 win 5792 <mss 1460,sackOK,timestamp 235788005 26363088,nop,wscale 0> (DF)
0x0000   4500 003c 0000 4000 4006 c073 xxxx xxxx        E..<..@.@..s..y.
0x0010   3f64 a839 0035 0423 b5e5 24bb 5e5d 0dc6        ?d.9.5.#..$.^]..
0x0020   a012 16a0 409d 0000 0204 05b4 0402 080a        ....@...........
0x0030   0e0d d6e5 0192 44d0 0103 0300                  ......D.....
08:56:28.951314 63.100.168.57.1059 > mynetworkhost.53: . ack 1 win 32120
<nop,nop,timestamp 26363100 235788005> (DF)
0x0000   4500 0034 fc43 4000 3206 d237 3f64 a839        E..4.C@.2..7?d.9
0x0010   xxxx xxxx 0423 0035 5e5d 0dc6 b5e5 24bc        ..y..#.5^]....$.
0x0020   8010 7d78 087e 0000 0101 080a 0192 44dc        ..}x.~........D.
0x0030   0e0d d6e5                                      ....
08:56:28.972782 63.100.168.57.2916 > mynetworkhost.53:  22218+ TXT CHAOS)? VERSION.BIND.
(30)
0x0000   4500 003a fc44 0000 3211 1226 3f64 a839        E..:.D..2..&?d.9
0x0010   xxxx xxxx 0b64 0035 0026 a111 56ca 0100        ..y..d.5.&..V...
0x0020   0001 0000 0000 0000 0756 4552 5349 4f4e        .........VERSION
0x0030   0442 494e 4400 0010 0003                       .BIND.....
08:56:28.973611 mynetworkhost.53 > 63.100.168.57.2916:  22218* 1/0/0 CHAOS) TXT 8.2.3-REL
(64) (DF)
0x0000   4500 005c 0000 4000 4011 c048 xxxx xxxx        E..\..@.@..H..y.
0x0010   3f64 a839 0035 0b64 0048 c322 56ca 8580        ?d.9.5.d.H."V...
0x0020   0001 0001 0000 0000 0756 4552 5349 4f4e        .........VERSION
0x0030   0442 494e 4400 0010 0003 0756 4552 5349        .BIND......VERSI
0x0040   4f4e 0442 494e 4400 0010 0003 0000 0000        ON.BIND.........
0x0050   000a 0938 2e32 2e33 2d52 454c                  ...8.2.3-REL
08:56:29.124057 63.100.168.57.1059 > mynetworkhost.53: F 1:1(0) ack 1 win 32120
<nop,nop,timestamp 26363117 235788005> (DF)
0x0000   4500 0034 fc50 4000 3206 d22a 3f64 a839        E..4.P@.2..*?d.9
0x0010   xxxx xxxx 0423 0035 5e5d 0dc6 b5e5 24bc        ..y..#.5^]....$.
0x0020   8011 7d78 086c 0000 0101 080a 0192 44ed        ..}x.l........D.
0x0030   0e0d d6e5                                      ....
08:56:29.124664 mynetworkhost.53 > 63.100.168.57.1059: F 1:1(0) ack 2 win 5792
<nop,nop,timestamp 235788032 26363117> (DF)
0x0000   4500 0034 f253 4000 4006 ce27 xxxx xxxx        E..4.S@.@..'..y.
0x0010   3f64 a839 0035 0423 b5e5 24bc 5e5d 0dc7        ?d.9.5.#..$.^]..
0x0020   8011 16a0 6f28 0000 0101 080a 0e0d d700        ....o(..........
0x0030   0192 44ed                                      ..D.
08:56:29.221545 63.100.168.57.1059 > mynetworkhost.53: . ack 2 win 32120
<nop,nop,timestamp 26363127 235788032> (DF)
0x0000   4500 0034 fc56 4000 3206 d224 3f64 a839        E..4.V@.2..$?d.9
0x0010   xxxx xxxx 0423 0035 5e5d 0dc7 b5e5 24bd        ..y..#.5^]....$.
0x0020   8010 7d78 0846 0000 0101 080a 0192 44f7        ..}x.F........D.
0x0030   0e0d d700                                      ....
```

SOURCE

Home network. Captured using TCPDUMP running on linux with kernel 2.4 RH 7.1

TIME

Dec 29th 2001

TOOL:

Tcpdump 3.6
This packet was captured as a part of the regular tcpdump dump logs which is regularly checked for anomalies. DNS port 53

PROBABLE SOURCE

```
UUNET Technologies, Inc. (NETBLK-
UUNET63) UUNET63   63.64.0.0 - 63.127.255.255
Intra West (NETBLK-UU-63-100-168-48) UU-63-100-168-48
 63.100.168.48 - 63.100.168.63
```

PROBABILITY OF BEING SPOOFED

Extremely low. I've based my answer on the following observations.

1. IP Id of source address was changing. It is surprising to note 2 IP packets with ID of 0 originating from the home network, and this seems to be normal when I checked tcpdump.
2. TCP handshake was completed, which leads me to believe that packets are being routed to the originating server. Probability of source routing and sniffing on foreign host exist, but very small. Objective of this probe it to find out which DNS server I'm running.
3. TTL is 50 (probably 64 when it started) a traceroute to the IP address confirms that its about 16 hops away.

```
11  0.so-1-0-0.XL2.CHI2.ALTER.NET (152.63.67.122)  129.676 ms
250.345 ms  252.621 ms
12  0.so-7-0-0.XR2.CHI2.ALTER.NET (152.63.67.134)  267.192 ms
278.174 ms  263.834 ms
13  192.ATM7-0.XR2.CHI6.ALTER.NET (152.63.65.42)  214.041 ms
279.062 ms  330.077 ms
14  190.ATM4-0.GW3.CLE1.ALTER.NET (152.63.67.77)  80.612 ms  82.368
ms  93.659 ms
15  intrawest-u39706-gw.customer.alter.net (157.130.120.62)
276.227 ms  203.190 ms  247.188 ms
16  63.100.168.57 (63.100.168.57)  98.804 ms  101.637 ms  107.113
ms
```

DESCRIPTION OF THE PROBE

DNS version probe
Bind is a very complex and powerful Name Server. Over the years a lot of features have been added to support the growing Internet. Unfortunately it also had its own share to bugs in multiple versions of bind. It is very helpful for the attacker to know which version of bind one is running so that they can use the right exploits against the server without raising too many alarms. The best way to do this is by requesting for version information from bind using the chaos class.

Its very simple to find out if your version of bind gives out version information.

```
$nslookup
> set class=chaos
> set type=txt
> version.bind
Server:  xxxx-xxx.xxx.xx.xxxx.net
Address:  24.xxx.xxx.xxx

VERSION.BIND    text = "8.2.3-REL"
```

List of Bind versions which have bugs as of Dec 31 2001
Vulnerable: '+', Not Vulnerable: '-', Feature does not exist: '   '
Source: http://www.isc.org/products/BIND/bind-security.html

version	zxfr	sigdiv0	srv	nxt	sig	naptr	maxdname	solinger	fdmax	complain	infoleak	tsig
4.8											+	
4.8.1							-				+	
4.8.2.1							-				+	
4.8.3							-				+	
4.9.3							-			+	+	
4.9.4							-			+	+	
4.9.4 p1							-			+	+	
4.9.5			-		+	+	+			+	+	
4.9.5 p1			-		+	+	+			+	+	
4.9.6			-		+	+	+			+	+	
4.9.7			-		-	+	+			+	+	
4.9.8			-		-	+	+			-	-	
8.1			-		+	+	+	+	+	-	+	
8.1.1			-		+	+	+	+	+	-	+	
8.1.2			-		-	+	+	+	+	-	+	
8.2	-	+	+	+	+	+	+	+	+	-	+	+
8.2 p1	-	+	+	+	+	+	+	+	+	-	+	+
8.2.1	-	+	+	+	+	+	+	+	+	-	+	+
8.2.2	+	+	+	-	-	+	+	-	-	-	+	+
8.2.2 p1	+	+	+	-	-	+	+	-	-	-	+	+
8.2.2 p2	+	+	+	-	-	-	-	-	-	-	+	+
8.2.2 p3	+	+	+	-	-	-	-	-	-	-	+	+
8.2.2 p4	+	+	+	-	-	-	-	-	-	-	+	+
8.2.2 p5	+	+	+	-	-	-	-	-	-	-	+	+
8.2.2 p6	+	-	+	-	-	-	-	-	-	-	+	+
8.2.2 p7	-	-	-	-	-	-	-	-	-	-	+	+
8.2.3	-	-	-	-	-	-	-	-	-	-	-	-
8.2.4	-	-	-	-	-	-	-	-	-	-	-	-
8.2.5	-	-	-	-	-	-	-	-	-	-	-	-
9.0.0		-	-	-	-	-	-	-	-	-	-	-
9.1.0		-	-	-	-	-	-	-	-	-	-	-
9.1.1		-	-	-	-	-	-	-	-	-	-	-
9.1.2		-	-	-	-	-	-	-	-	-	-	-
9.1.3		-	-	-	-	-	-	-	-	-	-	-

EVIDENCE OF ACTIVE TARGETING

It is possible that this activity was created by the "Lion" worm which randomly scans the Internet for un-patched DNS bind servers. I would rule out active targeting against this server since I don't see any other activity from the originating server. This is most probably a part of a larger scripted scan.

ATTACK MECHANISM

The attacker and the honeypot both completed a standard TCP handshake and the attacker gathered sufficient information on whether the target is venerable or not. Fortunately the version of DNS I was running was not prone to any known bugs at the time of attack. Hence I didn't get the second connection from the attacker which would have tried to use one of the bugs. A few months back "lion" was a popular DNS exploit worm which was making its rounds.

CORRELATION

This is not new and has been sufficiently documented on multiple websites. [Ref 9] [Ref 10] [Ref 11]

SERVERITY

Severity =
(Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality: 5 (Was a DNS server)
Lethality: 1 (No signs of compromise.)
System Countermeasures: 4 (Patched, Linux server running 2.4 kernel)
Network Countermeasures: 2 (No filtering on port 53, but snort installed)
Severity = 0 = (5 + 1) – (4 + 2)

DEFENSIVE RECOMENDATION

There are many was of reducing risks of a DNS attack. Here are a few quick pointers

1. Setup port 53 filtering [Ref 11] to refuse TCP connection from non-secondary servers. Unless the DNS query reply is bigger than 512 bytes all communication between a client and server could be done using UDP. From my experience as DNS administrator most DNS servers do not require TCP connectivity and can be safely shutdown. [Ref 12] Section 2.3.4

2. Setup snort/ids/firewall alerting to alert any requests for Chaos class to indicate a reconnaissance effort.
3. Edit named.conf to refuse/log or give incorrect information to probes to through attackers off guard.

```
//Add this to named.conf [ View is supported only in version 9 of bind
]
view "external-chaos" chaos {
recursion no;
zone "bind" {
        type master;
        file "forward/db.bind";
        allow-query {all;};
        allow-transfer {none;};
};
zone "." {
        type hint;
        file "forward/db.bind.root";
};
};
```

```
//this is the forward/db.bind.root file which is used by named.conf
$TTL    1D
$ORIGIN bind.
@       1D      CHAOS   SOA     ns.somecompany.com.
hostmaster.somecompany.com. (
                2001013101      ; serial
                3H              ; refresh
                1H              ; retry
                1W              ; expiry
                1D )            ; minimum
                CHAOS NS        ns.somecompany.com.

version.bind.   CHAOS  TXT "BIND 8.0.0-modifiedw"
authors.bind.   CHAOS  TXT "This is where false authors list goes"
```

MULTIPLE CHOICE TEST QUESTION

Question: TCP port 53 is required for regular DNS communication between a DNS server and DNS client ?

i) True
ii) False

Answer : False
TCP is used only if the DNS reply data doesn't fit a single UDP packet. In most cases this is close to 512 bytes of data, which is far more than a regular DNS server and client exchange.

NETWORK DETECT 3: Port 22 (ssh) Scan

```
01/01/2002 08:12:27.727681 217.96.220.67.22 > mynetworkhost.22: SF 2117545868:2117545868(0) win 1028
01/01/2002 08:12:27.733086 mynetworkhost.22 > 217.96.220.67.22: S 3027711864:3027711864(0) ack 2117545869 win
5840 <mss 1460> (DF)
01/01/2002 08:12:27.962934 217.96.220.67.22 > mynetworkhost.22: R 2117545869:2117545869(0) win 0
01/01/2002 08:12:28.353254 217.96.220.67.2126 > mynetworkhost.22: S 1747216558:1747216558(0) win 32320 <mss
1616,sackOK,timestamp 33382341 0,nop,wscale 0> (DF)
01/01/2002 08:12:28.353613 mynetworkhost.22 > 217.96.220.67.2126: S 3041406059:3041406059(0) ack 1747216559
win 5792 <mss 1460,sackOK,timestamp 258561470 33382341,nop,wscale 0> (DF)
01/01/2002 08:12:28.576523 217.96.220.67.2126 > mynetworkhost.22: . ack 1 win 32320 <nop,nop,timestamp 33382364
258561470> (DF)
01/01/2002 08:12:29.002455 mynetworkhost.22 > 217.96.220.67.2126: P 1:24(23) ack 1 win 5792 <nop,nop,timestamp
258561535 33382364> (DF)
01/01/2002 08:12:29.228597 217.96.220.67.2126 > mynetworkhost.22: . ack 24 win 32297 <nop,nop,timestamp
33382429 258561535> (DF)
01/01/2002 08:12:29.291382 217.96.220.67.2126 > mynetworkhost.22: F 1:1(0) ack 24 win 32320 <nop,nop,timestamp
33382435 258561535> (DF)
01/01/2002 08:12:29.293590 mynetworkhost.22 > 217.96.220.67.2126: F 24:24(0) ack 2 win 5792 <nop,nop,timestamp
258561564 33382435> (DF)
01/01/2002 08:12:29.506028 217.96.220.67.2126 > mynetworkhost.22: . ack 25 win 32319 <nop,nop,timestamp
33382457 258561564> (DF)
```

SOURCE

Home network, using TCPDUMP on linux running 2.4 kernel with RH 7.1

TIME

Jan 1st 2002

TOOL

Tcpdump

PROBABLE SOURCE

```
inetnum:      217.96.220.0 - 217.96.220.127
netname:      SIEDLCE-SDI
descr:        TP S.A. SDI
descr:        Siedlce Blonie
country:      PL
admin-c:      JZ1363-RIPE
tech-c:       WR2851-RIPE
status:       ASSIGNED PA
mnt-by:       AS5617-MNT
changed:      tkielb@cst.tpsa.pl 20001218
source:       RIPE

route:        217.96.0.0/14
descr:        TPNET (PL)
descr:        Provider Local Registry
origin:       AS5617
notify:       konradpl@zt.piotrkow.tpsa.pl
```

```
mnt-by:        TPNET
changed:       konradpl@zt.piotrkow.tpsa.pl 20001122
source:        RIPE

person:        Jaroslaw Zaciura
address:       Zaklad Telekomunikacji Siedlce
address:       ul. Blonie 5
address:       Siedlce
address:       POLAND
phone:         +48 25 6446060
nic-hdl:       JZ1363-RIPE
mnt-by:        AS5617-MNT
changed:       tkielb@cst.tpsa.pl 20001012
source:        RIPE

person:        Wojciech Rawa
address:       Telekomunikacja Polska S.A.
address:       Zaklad Telekomunikacji w Siedlcach
address:       ul. Blonie 5
address:       08-110 Siedlce
address:       POLAND
phone:         +48 25 6649631
fax-no:        +48 25 6339660
nic-hdl:       WR2851-RIPE
mnt-by:        AS5617-MNT
changed:       tkielb@cst.tpsa.pl 20000612
source:        RIPE
```

PROBABILITY OF BEING SPOOFED

Extreemly low. I base my answer on the following observations.

1. Complete TCP transactions were completed which indicate successful routing of packets from attacked hosts to the attacker. Probability of source routing and remote listener do exist, but its rare.
2. There are two distinct probes with totally different signatures in this packet capture. The first probe has strong signature for a crafted packet, however because of the "Reset" send from the attacker for the response of the first packet and the subsequent second probe sent from the attacker, I'm forced to believe that the source address was not spoofed.

DESCRIPTION OF THE PROBE

There are two distinct sets of probes here both happening one after the other indicating some relation between the first and the second probe.

1. The first probe is a standard "stimulus/response" probe, looking for open port 22 . Notice the following characteristics
   a. "SF" flag in the first probe
   b. The identical source/destination ports
   c. Window size of 1028

   These have a strong signature of a crafted packet. A quick research on the google showed that this particular packet is not just used to detect port 22 but has also been used to detect port 53. [Ref 18] It leads me to believe that this particular tool was not written by this attacker or for this particular attack.

2. The second probe was launched after the confirmation of existence of listener on port 53.

EVIDENCE OF ACTIVE TARGETING

These probes are too close together indicating that this is probably an automated script which is randomly selecting hosts to attack. “SF” flag could be called a OS fingerprinting attempt, however from the way its used here it seems like the probe is actually designed to infiltrate low-end firewalls and was not making an attempt to do a fingerprint. Absence of additional probes or OS fingerprinting probes rules out this host as an “Active Target”.

ATTACK MECHANISM

The objective of the second probe was to gather further information on version of SSH running. In the recent past a number of SSH related bugs have come to known[Ref 19] [Ref 20]. If the second reconnaissance probe reported a venerable host, its possible that a third connection would have been established to exploit the host.

CORRELATIONS

[Ref 19] [Ref 20] This particular attack is on the rise in last one month after the discovery of the SSH compensator attack which affects a large number of hosts.

SEVERITY

Severity =
(Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality: 5 (A DNS server)
Lethality: 1 (No signs of compromise.)
System Countermeasures: 4 (Patched, Linux server running 2.4 kernel)
Network Countermeasures: 2 (No filtering on port 53, but snort installed)
Severity = 0 = (5 + 1) – (4 + 2)

DEFENSIVE RECOMENDATION

This particular host was not venerable to the SSH compensator attack.

MULTIPLE CHOICE TEST QUESTION

```
01/01/2002 08:12:27.727681 217.96.220.67.22 > mynetworkhost.22: SF 2117545868:2117545868(0) win 1028
01/01/2002 08:12:27.733086 mynetworkhost.22 > 217.96.220.67.22: S 3027711864:3027711864(0) ack 2117545869 win
5840 <mss 1460> (DF)
```

Question:
What does source port 22 and destination port 22 indicate in this set of communication.
Select the wrong answer.

1) port 22 as client and server indicates that both the processes required super user privileges to start up
2) This is most probably a port scan
3) Its impossible to have both source and destination port same in the same connection.

Answer.
3 is the wrong answer. This pattern is typical of a port scan.

NETWORK DETECT 4: dtspcd scan

```
                                                                                            TCPDUMP VIEW

12/31/2001 10:25:04.130000 209.207.216.179.6112 > mynetworkhost.6112: S 175889461:175889461(0) win 19301
0x0000   4500 0028 c1d2 0000 6e06 64bc d1cf d8b3        E..(....n.d.....
0x0010   xxxx xxxx 17e0 17e0 0a7b dc35 6c05 9d2d        @.;......{.5l..-
0x0020   5002 4b65 1e98 0000 0000 0000 0000             P.Ke..........
12/31/2001 10:25:04.130000 mynetworkhost.6112 > 209.207.216.179.6112: R 0:0(0) ack 175889462 win 0 (DF)
0x0000   4500 0028 0000 4000 ff06 558e xxxx xxxx        E..(..@...U.@.;.
0x0010   d1cf d8b3 17e0 17e0 0000 0000 0a7b dc36        .............{.6
0x0020   5014 0000 731d 0000                            P...s...
                                                                                              SNORT VIEW
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/31-02:25:04.130000 209.207.216.179:6112 -> mynetworkhost:6112
TCP TTL:110 TOS:0x0 ID:49618 IpLen:20 DgmLen:40
******S* Seq: 0xA7BDC35  Ack: 0x6C059D2D  Win: 0x4B65  TcpLen: 20

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

12/31-02:25:04.130000 mynetworkhost:6112 -> 209.207.216.179:6112
TCP TTL:255 TOS:0x0 ID:0 IpLen:20 DgmLen:40 DF
***A*R** Seq: 0x0  Ack: 0xA7BDC36  Win: 0x0  TcpLen: 20

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

SOURCE

Honeypot on Employer's external network

TIME

Dec 30th 2001

TOOL

Tcpdump and snort.

PROBABLE SOURCE

```
        Verio, Inc. (NET-VRIO-209-207-128)
           8005 South Chester Street
           Englewood, CO 80112
           US

           Netname: VRIO-209-207-128
           Netblock: 209.207.128.0 - 209.207.255.255
           Maintainer: VRIO

           Coordinator:
              Verio, Inc.  (VIA4-ORG-ARIN)  vipar@verio.net
              303.645.1900
```

PROBABILITY OF BEING SPOOFED

It is possible that this is spoofed, but I'm led to believe that this was not spoofed based on following reasons

1. Only one packet was noticed, and this is a SYN packet and not a ACK packet
2. The target port is 6112 which could mean that the attacker is looking for venerable dtspcd hosts. This reconnaissance would be incomplete without a return probe from the attacked host.
3. TTL is 110 on this particular packet. A traceroute to source puts this host about 15 hops away which puts the default TTL of the source close to 128 which is a realistic TTL for many lots of Windows boxes.

```
.
.
10  209.133.31.106  70.078 ms  69.801 ms  70.866 ms
11  129.250.2.174  69.849 ms  69.869 ms  69.825 ms
12  129.250.17.58  70.266 ms  70.146 ms  70.219 ms
13  216.167.88.116  70.163 ms  70.354 ms  70.436 ms
14  * * *
15  * * *
```

DESCRIPTION OF THE ATTACK

dtspcd remote root exploit

The probe above is a standard "stimulus/response" packet looking for open ports. This attack exploits a remote buffer bug in dtspcd, which is part of CDE Xwindows application running on different kinds of Unix servers. This bug was first detected approximately 2 years back, but there have been resurgence in scans. Quite a few advisories have been posted in last 30 days indicating that this kind of scan is on the rise.

ATTACK MECHANISM

Looking at the packet (source port and destination port) it has a strong signature for a crafted packet. The attacked at this stage of probe is looking for open 6112 tcp port on the target network using a standard port scanner. Once this tool generates a list of possible takeover targets, it would be fed to the remote exploit which would then make an attempt to exploit the remote hosts.

CORRELATIONS

[Ref 13] [Ref 14] [Ref 15] [Ref 16] Talks about this exploit and port 6112/tcp scan.

SEVERITY

Severity =
(Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality: 1 (Was a honeypot)
Lethality: 1 (No signs of compromise.)
System Countermeasures: 4 (Patched, Linux server running 2.4 kernel)
Network Countermeasures: 5 (Firewalls with snort IDS)
Severity = -7 = (1 + 1) – (4 + 5)

DEFENSIVE RECOMENDATION

X Windows is not usually used on most critical servers. Disable Xwindows so that such services can not be compromised.
Or at least disable dtspcd from inetd.conf. If you want to use this service, please obtain the latest patch from your vendor.

MULTIPLE CHOICE TEST QUESTION

Quesiton: What does inetd.conf do on a unix server ?
Answer:

1) connects to the internet
2) starts up network routing
3) starts up networking services
4) inetd.conf has nothing to do with networking

Answer is 3. It starts up networking services like telnet/ftp etc.

NETWORK DETECT 5: Coordinated scans

```
18:36:40.237060 24.126.117.66.1234 > my_network_host.27374: S
1522565747:1522565747(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
18:36:40.237620 my_network_host.27374 > 24.126.117.66.1234: S
4254502594:4254502594(0) ack 1522565748 win 5840 <mss 1460,nop,nop,sackOK> (DF)
18:36:40.266672 24.126.117.66.1234 > my_network_host.27374: . ack 1 win 17520 (DF)
18:36:40.284923 24.126.117.66.1234 > my_network_host.27374: F 1:1(0) ack 1 win 17520
(DF)
18:36:40.292110 my_network_host.27374 > 24.126.117.66.1234: . ack 2 win 5840 (DF)
18:36:40.345636 my_network_host.27374 > 24.126.117.66.1234: F 1:1(0) ack 2 win 5840
(DF)
18:36:40.380784 24.126.117.66.1234 > my_network_host.27374: . ack 2 win 17520 (DF)

18:36:40.425688 24.49.241.234.4065 > my_network_host.27374: S 660983127:660983127(0)
win 16384 <mss 1460,nop,nop,sackOK> (DF)
18:36:40.428170 my_network_host.27374 > 24.49.241.234.4065: R 0:0(0) ack 660983128
win 0 (DF)
18:36:41.093082 24.49.241.234.4065 > my_network_host.27374: S 660983127:660983127(0)
win 16384 <mss 1460,nop,nop,sackOK> (DF)
18:36:41.093413 my_network_host.27374 > 24.49.241.234.4065: R 0:0(0) ack 1 win 0 (DF)
18:36:41.645351 24.49.241.234.4065 > my_network_host.27374: S 660983127:660983127(0)
win 16384 <mss 1460,nop,nop,sackOK> (DF)
18:36:41.645673 my_network_host.27374 > 24.49.241.234.4065: R 0:0(0) ack 1 win 0 (DF)

18:36:40.451162 24.52.38.191.1336 > my_network_host.27374: S 5111247:5111247(0) win
8192 <mss 1460,nop,nop,sackOK> (DF)
18:36:40.451634 my_network_host.27374 > 24.52.38.191.1336: R 0:0(0) ack 5111248 win 0
(DF)
18:36:41.029328 24.52.38.191.1336 > my_network_host.27374: S 5111247:5111247(0) win
8192 <mss 1460,nop,nop,sackOK> (DF)
18:36:41.029680 my_network_host.27374 > 24.52.38.191.1336: R 0:0(0) ack 1 win 0 (DF)
18:36:41.633647 24.52.38.191.1336 > my_network_host.27374: S 5111247:5111247(0) win
8192 <mss 1460,nop,nop,sackOK> (DF)
18:36:41.633964 my_network_host.27374 > 24.52.38.191.1336: R 0:0(0) ack 1 win 0 (DF)
18:36:42.237411 24.52.38.191.1336 > my_network_host.27374: S 5111247:5111247(0) win
8192 <mss 1460,nop,nop,sackOK> (DF)
18:36:42.237727 my_network_host.27374 > 24.52.38.191.1336: R 0:0(0) ack 1 win 0 (DF)

18:36:40.634403 199.120.100.12.1984 > my_network_host.27374: S 12312144:12312144(0)
win 8192 <mss 1460> (DF)
18:36:40.634969 my_network_host.27374 > 199.120.100.12.1984: S
4258200991:4258200991(0) ack 12312145 win 5840 <mss 1460> (DF)
18:36:40.912317 199.120.100.12.1984 > my_network_host.27374: . ack 1 win 8760 (DF)
a
18:36:40.693931 66.50.92.111.2213 > my_network_host.27374: S 8829385:8829385(0) win
8192 <mss 536,nop,nop,sackOK> (DF)
18:36:40.694579 my_network_host.27374 > 66.50.92.111.2213: S 4264008171:4264008171(0)
ack 8829386 win 5840 <mss 1460,nop,nop,sackOK> (DF)
18:36:41.029116 66.50.92.111.2213 > my_network_host.27374: . ack 1 win 8576 (DF)
18:36:41.029420 my_network_host.27374 > 66.50.92.111.2213: R 4264008172:4264008172(0)
win 0 (DF)

18:36:40.710193 24.80.72.230.1409 > my_network_host.27374: S 808035627:808035627(0)
win 5840 <mss 1460,nop,nop,sackOK> (DF)
18:36:40.710752 my_network_host.27374 > 24.80.72.230.1409: S 4262141193:4262141193(0)
ack 808035628 win 5840 <mss 1460,nop,nop,sackOK> (DF)
18:36:41.232308 24.80.72.230.1409 > my_network_host.27374: . ack 1 win 5840 (DF)
18:36:41.232555 my_network_host.27374 > 24.80.72.230.1409: R 4262141194:4262141194(0)
win 0 (DF)

18:36:40.712432 199.199.245.35.2032 > my_network_host.27374: S 36873853:36873853(0)
win 8192 <mss 536,nop,nop,sackOK> (DF)
18:36:40.712818 my_network_host.27374 > 199.199.245.35.2032: S
```

```
4254428773:4254428773(0) ack 36873854 win 5840 <mss 1460,nop,nop,sackOK> (DF)
18:36:41.191115 199.199.245.35.2032 > my_network_host.27374: . ack 1 win 8576 (DF)
18:36:41.191400 my_network_host.27374 > 199.199.245.35.2032: R
4254428774:4254428774(0) win 0 (DF)

18:36:40.947348 66.57.188.145.2511 > my_network_host.27374: S 24821064:24821064(0)
win 8192 <mss 1460,nop,nop,sackOK> (DF)
18:36:40.947793 my_network_host.27374 > 66.57.188.145.2511: R 0:0(0) ack 24821065 win
0 (DF)
18:36:42.118518 66.57.188.145.2511 > my_network_host.27374: S 24821064:24821064(0)
win 8192 <mss 1460,nop,nop,sackOK> (DF)
18:36:42.118847 my_network_host.27374 > 66.57.188.145.2511: R 0:0(0) ack 1 win 0 (DF)
18:36:42.735185 66.57.188.145.2511 > my_network_host.27374: S 24821064:24821064(0)
win 8192 <mss 1460,nop,nop,sackOK> (DF)
18:36:42.735612 my_network_host.27374 > 66.57.188.145.2511: R 0:0(0) ack 1 win 0 (DF)
18:36:41.538836 66.57.188.145.2511 > my_network_host.27374: S 24821064:24821064(0)
win 8192 <mss 1460,nop,nop,sackOK> (DF)
18:36:41.539171 my_network_host.27374 > 66.57.188.145.2511: R 0:0(0) ack 1 win 0 (DF)

18:36:41.979173 64.251.138.10.4734 > my_network_host.27374: S
1623373672:1623373672(0) win 8760 <mss 536,nop,nop,sackOK> (DF)
18:36:41.979616 my_network_host.27374 > 64.251.138.10.4734: R 0:0(0) ack 1623373673
win 0 (DF)
18:36:43.288759 64.251.138.10.4734 > my_network_host.27374: S
1623373672:1623373672(0) win 8760 <mss 536,nop,nop,sackOK> (DF)
18:36:43.289111 my_network_host.27374 > 64.251.138.10.4734: R 0:0(0) ack 1 win 0 (DF)
18:36:43.918064 64.251.138.10.4734 > my_network_host.27374: S
1623373672:1623373672(0) win 8760 <mss 536,nop,nop,sackOK> (DF)
18:36:43.918379 my_network_host.27374 > 64.251.138.10.4734: R 0:0(0) ack 1 win 0 (DF)
```

SOURCE

Home network, TCPDUMP running on linux with kernel 2.4 and RH 7.1

TIME

Jan 1st 2001

TOOL

tcpdump 3.6

PROBABLE SOURCE

```
                         4 Hosts did not send Fin Packets.
66.57.188.145
64.251.138.10
24.49.241.234
24.52.38.191

                              5 Hosts did ACK to SYN/ACK
199.120.100.12
       Iowa Network Services, Inc. (NET-IOWA) NETIOWA-64
                                             199.120.64.0 - 199.120.127.255
       Pioneer Internet (NETBLK-NET-WIT-PIONET-020) NET-WIT-PIONET-020
                                             199.120.98.0 - 199.120.100.255

66.50.92.111
       Puerto Rico Telephone Company (NETBLK-PRTC-NET)
          PO Box 360998
          San Juan PR 00936-0998
          PR
```

```
            Netname: PRTC-NET
            Netblock: 66.50.0.0 - 66.50.255.255
            Maintainer: PRTC

            Coordinator:
               PRTC Nameservice  (PN50-ORG-ARIN)  nameserv@PRTC.NET
               787-288-9401
         Fax- 787-782-7940

24.80.72.230
         Shaw Fiberlink (aka Shaw@home) (NETBLK-FIBERLINK-CABLE-3BLK)
            Suite 800, 630 3rd Avenue SW
            Calgary, Alberta T2P 4L4
            CA

            Netname: FIBERLINK-CABLE-3BLK
            Netblock: 24.80.0.0 - 24.87.255.255
            Maintainer: FBCA

            Coordinator:
               Shaw High-Speed Internet  (ZS178-ARIN)  ipadmin@sjrb.ca
               (403)750-7428

199.199.245.35
         Minnesota Regional Network (NETBLK-MRNET-C-BLOCK4) MRNET-C-BLOCK4
                                                   199.199.0.0 - 199.199.255.255
         Bevcomm (NETBLK-BEVCOMM-BLK-1) BEVCOMM-BLK-1  199.199.240.0 - 199.199.247.255

24.126.117.66
         AT&T Broadband West (NET-ATTB-WEST-4)
            27 Industrial Ave.
            Chelmsford, MA 01824
            US

            Netname: ATTB-WEST-4
            Netblock: 24.126.0.0 - 24.127.191.255
            Maintainer: ATBW

            Coordinator:
               ATT Broadband  (ZM117-ARIN)  ipadmin@attbroadband.com
```

PROBABILITY OF BEING SPOOFED

Some of these could have been spoofed SYN packets. However I'd say that most of it is not spoofed based on the following reasons.

1. We do notice a few ACKs in respose to the SYN/ACK response from the attacked host
2. Packets originating from each host have distinct packet signatures which makes us believe that these were not created on the same host.
3. There are multiple objectives from this scan and since this is the total number of packets we received we rule out a DOS attack which is one of the common reasons of spoofing packets.

DESCRIPTION OF THE PROBE

Coordinated SubSevent discovery probes

This is perhaps one of the most interesting probes I've ever encountered. Not having network dumps for the entire class C or class B network is a pity, because there is no way to figure out using this info whether this evidence of active targeting on a specific host. Please notice that though I've bunched the probes from similar hosts together, they were actually happening all at the same time.
There could be multiple objective to this probe, some of this are listed below

1. Confirm the existence of a listening port on this specfic host
2. Get a list of Sequence numbers to predict sequence numbers for future attacks.
3. OS finger printing based on TCP communication
4. To gather information network activity by sending coordinated probes.
5. To gather information on system activity by flooding the system with multiple SYNs.
6. To gather routing information (to triangulate) a host on the network by sending probes from multiple sources (using TTL)

ATTACK MECHANISM

27374 is a popular Trojan port used by SubSeven. However, the attacker could have stopped the reconnaissance after the first probe. The information I collected is insufficient to conclusively prove the objective of this probe. Though some may argue that this is a hostile activity, I'd like to categorize this as a probable reconnaissance probe which could lead to a full scan hostile activity at a later date.

CORRELATIONS

I'd sent the probe description to multiple mailing lists but have failed to obtain any correlations yet.

EVIDENCE OF ACTIVE TARGETTING

Its highly probable that this was a result of active targeting, since I didn't notice such activity reported by anyone else in the past.

SEVERITY

Severity =
(Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality: 5 (A DNS server)
Lethality: 1 (No signs of compromise.)

System Countermeasures: 4 (Patched, Linux server running 2.4 kernel)
Network Countermeasures: 1 (snort didn't detect the distributed scan, was detected during a visual check)
Severity = 1 = (5 + 1) – (4 + 1)

DEFENSIVE RECOMENDATION

Since this probe looks like a result of active targeting, its recommended to closely monitor this host for further activity of such nature.

MULTIPLE CHOICE TEST QUESTION

What does the IP ID change in the two packets indicate about the src tool/hosts used ?
Select the wrong answer.

5) This could be spoofed using a packet crafting tool
6) This cannot be spoofed using a packet crafting tool
7) The receiving host generates random IP ID, this question is irrelevant
8) Probably the Source host OS generated the IP headers and it indicates lot of network activity is going on at the source host.

Answer is 2, because this can be spoofed.

ASSIGNMENT 3 - Analyze This

Other practicals looked at during this analysis
http://www.giac.org/practical/Gregory_Lajon_GCIA.doc
http://www.giac.org/practical/Thomas_Rodriguez_GCIA.doc
http://www.giac.org/practical/Philipp_Stadler_GCIA.doc

SUMMARY

The data files picked were between December 19 and December 26. I picked more than 5 days of data to make more correlations between the data from different dates. This period has historical significance in the Intrusion world.

Network profile

- The network is very active during working days which indicates this is a educational network or a big corporate network.
- The network has lot of kazaa, Gnutella users data.
- Lot of portscans originate from inside the network which is abnormal. This could be signs of compromised systems. Top 5 portscanners on this network is internal servers.
- Lot of hosts use UDP based applications which tend to trigger snort to think UDP portscanning is going on..
- Probably atleast 105 FTP servers are active on this network, which are visible from outside.

The following IP addresses were extensively investigated for suspicious activity and their contact information is included in the report below.

i)	65.2.208.87	- Tiny Fragment source
ii)	67.161.190.60	- Tiny Fragment source
iii)	24.0.28.234	- Heavy SSH scan
iV)	210.125.178.52	- Heavy port scanning from OOS logs
v)	62.211.247.3	- Heavy port scanning from portscan logs

Dates covered for this analisys, Dec,19-25 2001

TOP TALKERS

Top Alerts

```
        cat  alert* | grep -v spp_portscan > alert
        cat alert  | perl -e 'while(<STDIN>) {@a=split(/\[\*\*\]/,$_);print "$a[1]\n";}'
| sort -nr | uniq -c
```

Signature	# of alerts	% of total alerts	# of source	# of Dst
Tiny Fragments - Possible Hostile Activity	101269	9	8	5
ICMP Source Quench	91743	8	122	149
MISC traceroute	79572	7	153	26
MISC Large UDP Packet	79280	7	56	10
MISC source port 53 to <1024	71891	7	10591	19
Watchlist 000220 IL-ISDNNET-990517	66116	6	66	33
CS WEBSERVER - external web traffic	65324	6	9485	2
INFO MSN IM Chat data	36366	3	306	351
WEB-MISC prefix-get //	32485	3	1381	8
SYN-FIN scan!	21300	2	5	17024
ICMP Destination Unreachable (Host Unreachable)	13344	1	847	60
ICMP Echo Request BSDtype	11375	1	30	32
ICMP Destination Unreachable (Communication Administratively Prohibited)	8616	0	253	118
SCAN Proxy attempt	8059	0	115	4703
INFO Outbound GNUTella Connect accept	7257	0	1096	52
Queso fingerprint	6184	0	85	57
ICMP Echo Request Nmap or HPING2	5567	0	37	256
ICMP Fragment Reassembly Time Exceeded	5045	0	38	86
ICMP Echo Request L3retriever Ping	4296	0	7	13
SMB Name Wildcard	3993	0	262	1302
BACKDOOR NetMetro File List	3586	0	1	1
Watchlist 000222 NET-NCFC	2519	0	30	22
ICMP Destination Unreachable (Protocol Unreachable)	1964	0	27	75
INFO FTP anonymous FTP	1712	0	420	175
External RPC call	1519	0	7	923
connect to 515 from outside	1487	0	5	821
WEB-MISC Attempt to execute cmd	1381	0	107	43
WEB-MISC 403 Forbidden	1276	0	15	642
Null scan!	1275	0	350	60
INFO Inbound GNUTella Connect accept	1233	0	44	1020

Top Source IP Addresses

```
cat al | perl -e 'while(<STDIN>) {@a=split(/\[\*\*\]/,$_);@b=split(/
/,$a[2]);@c=split(/:/,$b[1]);print "$c[0]\n";}'  | sort -nr | uniq -c | sort -nr
```

```
100851 MY.NET.8.1
  90513 MY.NET.5.13
  69994 209.190.237.123
  63069 212.179.35.118
  12174 62.211.247.3
   7663 MY.NET.137.7
   6212 192.115.189.100
   5652 24.166.247.206
   5648 216.106.172.149
   5597 198.32.224.31
   5027 24.0.28.234
   4908 206.65.191.129
   4668 65.165.14.43
   4098 200.57.36.68
   4065 66.77.74.235
   3783 MY.NET.60.11
   3351 65.207.94.30
   3307 128.223.4.21
   3145 141.213.11.120
   3093 134.93.19.12
   2905 147.46.59.144
   2522 MY.NET.87.50
   2361 61.219.53.135
   2355 63.146.1.33
```

Top Source ports

```
cat al | perl -e 'while(<STDIN>) {@a=split(/\[\*\*\]/,$_);@b=split(/
/,$a[2]);@c=split(/:/,$b[1]);print "$c[1]\n";}'  | sort -nr | uniq -c | sort -nr
```

```
  72004 53
  61300 60339
  40000 0
  15240 1863
  12193 21
   9139 22
   8611 6346
   3885 137
   3590 20
   3403 54567
   3377 8448
   3063 80
   1984 1654
   1812 1214
   1754 7000
   1379 21172
   1109 2353
   1060 2083
   1056 5031
    907 25
    686 4170
    597 4506
    534 23
    500 3434
    486 29369
    458 1172
    429 1405
    407 69
    372 111
    339 3319
```

Top Destination IPs

```
cat al | perl -e 'while(<STDIN>) {@a=split(/\[\*\*\]/,$_);@b=split(/
/,$a[2]);@c=split(/:/,$b[3]);print "$c[0]\n";}' | sort -nr | uniq -c | sort -nr
```

```
100917 MY.NET.16.42
  88388 MY.NET.140.9
  69997 MY.NET.70.134
  67261 MY.NET.100.165
  64969 MY.NET.70.70
  33728 MY.NET.253.114
  24146 MY.NET.1.3
  18364 MY.NET.1.5
  17790 MY.NET.1.4
  10044 MY.NET.70.148
   8011 MY.NET.153.210
   7207 216.158.50.2
   6951 MY.NET.111.157
   6264 MY.NET.88.88
   5516 MY.NET.137.7
   4644 MY.NET.98.177
   3586 209.49.12.32
   3124 MY.NET.130.122
   3092 MY.NET.253.105
   2511 MY.NET.200.158
   2446 MY.NET.200.156
   2425 MY.NET.200.151
   2417 MY.NET.200.149
   2390 MY.NET.200.154
   2332 MY.NET.200.147
   2110 MY.NET.200.175
   2096 MY.NET.200.167
   2081 MY.NET.200.163
   2067 MY.NET.200.169
   2061 MY.NET.200.173
   2041 MY.NET.200.171
   2035 MY.NET.200.178
```

Top Destination ports

```
cat al | perl -e 'while(<STDIN>) {@a=split(/\[\*\*\]/,$_);@b=split(/
/,$a[2]);@c=split(/:/,$b[3]);print "$c[1]\n";}'  | sort -nr | uniq -c | sort -nr
```

```
105503 80      Web
  72100 53      DNS
  63915 1214    Kazaa running on these servers.
  37761 0       port 0 probes
  21151 1863    MSN messenger
  14799 21      Telnet
   9575 22      SSH
   5657 1643
   4883 1080    Wingate proxy port
   4834 33464
   4790 33463
   4763 33462
   4087 33465
   4083 33459
   4071 33461
   4063 33460
   4038 33466
   4030 137      NBT
   4007 33467
   3586 5032     Probable Trojan (Net Metropolitan) [Ref 22]
   3413 1434
   3300 8080     Web Proxy
   3252 38001
   3200 33470
   3183 33469
   3165 33468
```

Top Portscaning hosts – probably compromised systems

```
cat scans.logs | grep -v UDP | cut -d" " -f4 | cut -d":" -f1| sort -nr | uniq -c | sort -
nr
```

I filtered UDP while getting this list to remove the hosts which were doing lots of UDP transactions. Aparently this network has a lot of UDP based applications which generate wrong portscan alerts.

```
926049 MY.NET.162.233
 685164 MY.NET.163.15
  65024 MY.NET.111.157
  40847 MY.NET.70.225
  12599 MY.NET.253.24
  12251 62.211.247.3
   9895 24.205.153.114
   9876 211.248.231.10
   9802 210.61.63.66
   9508 65.165.14.43
   8570 MY.NET.60.11
   7952 210.77.145.30
   7680 210.58.102.86
```

Top OOS (Out of Spec) Source Ips

```
cat oos.logs | grep "^12" | cut -d" " -f2 | cut -d":" -f1 | sort -nr | uniq -c | sort -nr
| more
```

```
   7931 24.0.28.234
    167 210.125.178.52
     80 199.183.24.194
     40 64.172.24.155
     15 24.36.185.188
     12 141.157.92.22
     11 211.39.150.91
      9 65.165.238.50
      7 213.84.157.192
      7 202.168.254.178
      6 65.105.159.22
      5 193.120.224.170
      4 24.222.59.173
      4 204.228.228.145
      4 202.130.239.149
      4 12.230.253.9
      3 24.28.134.6
```

Top OOS (Out of Spec) Source Ports

```
cat oos.logs | grep "^12" | cut -d" " -f2 | cut -d":" -f2 | sort -nr | uniq -c | sort -nr
| more
```

```
   7931 22
     19 18245
      8 5635
      8 1770
      3 2889
      3 2213
      3 1690
      3 0
      2 50067
      2 45672
      2 45210
      2 42172
      2 42159
      2 42157
```

Top OOS (Out of Spec) Destination Ips

```
cat oos.logs | grep "^12" | cut -d" " -f4 | cut -d":" -f1 | sort -nr | uniq -c | sort -nr
| more
```

```
    168 MY.NET.163.15
     89 MY.NET.253.43
     44 MY.NET.70.70
     17 MY.NET.253.114
     16 MY.NET.70.49
     16 MY.NET.253.125
     14 MY.NET.253.41
     12 MY.NET.1.6
     10 MY.NET.60.14
      9 MY.NET.100.165
      8 MY.NET.100.236
      6 MY.NET.99.39
      6 MY.NET.5.29
```

Top OOS (Out of Spec) Destination Ports

```
cat oos.logs | grep "^12" | cut -d" " -f4 | cut -d":" -f2 | sort -nr | uniq -c | sort -nr
| more
```

```
   7932 22
    116 25
     42 80
     34 1214
     19 21536
     12 563
     10 0
      7 113
      6 6346
      2 98
      2 97
      2 9
      2 86
      2 83
      2 79
      2 78
      2 68
```

DEEPER TRAFFIC ANALYSIS

Tiny Fragments - Possible Hostile Activity

Description

Any TCP packet with “More Fragments” bit enabled is said to be fragmented. Fragmentation happens when the data send by a host needs to be broken up into multiple fragments due to network hardware limitations. It can also happen due to bad packets which get corrupted on the way. The phrase “tiny Fragments” is reserved for those packets that do not justify the reason of its fragmentation. The size of the fragments below which this warning should be generated (by snort) is configurable. We do not have this information in the logs.

Problem

Fragmentation is used by a lot of tools to avoid detection by weak firewalls that do not re-assemble packets. Without a re-assembly mechanism its difficult for the firewalls/router to find out the state of the TCP packet. Once inside the network Fragmented packets act as regular stimulus that will almost always generate a successful response from a listening system.

Dump

Source Addresses

```
    223 67.161.190.60
    190 65.2.208.87
      2 24.8.58.167
      2 24.34.101.43
      1 MY.NET.8.112/19-09
      2 MY.NET.8.112/19-00
 100851 MY.NET.8.1 <-> MY.NET.16.42
```

```
      1 MY.NET.5.7512/20-00
```

Destination Addresses

```
    413 MY.NET.99.39
      2 MY.NET.253.125
 100851 MY.NET.16.42 <-> MY.NET.8.1
      2 MY.NET.100.236
```

Interestingly 99% of this traffic is between two internal servers. We do not have enough information to find out the details of this transaction. Since these are both behind firewalls (this is an assumption) I believe this is real traffic which is not meant to evade firewalls. However the 2 hosts from outside network which seem to be talking to MY.NET.99.39 doesn't look normal at all.

Source 67.161.190.60

```
12/26-14:03:31.895201  [**] spp_portscan: PORTSCAN DETECTED from 67.161.190.60 (STEALTH) [**]
12/26-13:50:46.478284  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-13:50:46.503825  [**] Null scan! [**] 67.161.190.60:10 -> MY.NET.99.39:5633
12/26-13:50:46.569662  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-13:50:46.582469  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-13:50:46.634190  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-13:50:46.645753  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-13:50:46.721760  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-13:50:46.870756  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-14:03:33.838989  [**] spp_portscan: portscan status from 67.161.190.60: 5 connections across 1 hosts: TCP(5), UDP(0) STEALTH [**]
12/26-13:50:52.960712  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-13:50:53.626741  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-14:03:35.954350  [**] spp_portscan: portscan status from 67.161.190.60: 2 connections across 1 hosts: TCP(2), UDP(0) STEALTH [**]
12/26-13:50:54.751361  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-13:50:54.764859  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-13:50:55.121809  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-13:50:56.648850  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-13:50:57.227524  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-14:03:38.032123  [**] spp_portscan: portscan status from 67.161.190.60: 2 connections across 1 hosts: TCP(2), UDP(0) STEALTH [**]
12/26-13:50:58.647827  [**] Null scan! [**] 67.161.190.60:31375 -> MY.NET.99.39:0
12/26-13:50:58.835950  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-13:50:58.848267  [**] Null scan! [**] 67.161.190.60:0 -> MY.NET.99.39:0
12/26-14:03:40.212583  [**] spp_portscan: portscan status from 67.161.190.60: 1 connections across 1 hosts: TCP(1), UDP(0) STEALTH [**]
```

Source 65.2.208.87

```
12/26-20:46:12.536754  [**] spp_portscan: PORTSCAN DETECTED from 65.2.208.87 (STEALTH) [**]
12/26-20:30:20.572335  [**] Null scan! [**] 65.2.208.87:0 -> MY.NET.99.39:0
12/26-20:30:20.857092  [**] Null scan! [**] 65.2.208.87:1785 -> MY.NET.99.39:1214
12/26-20:30:20.978203  [**] Null scan! [**] 65.2.208.87:0 -> MY.NET.99.39:0
12/26-20:30:22.238769  [**] Null scan! [**] 65.2.208.87:0 -> MY.NET.99.39:0
12/26-20:30:22.261296  [**] Null scan! [**] 65.2.208.87:0 -> MY.NET.99.39:0
12/26-20:30:22.274589  [**] Null scan! [**] 65.2.208.87:0 -> MY.NET.99.39:0
12/26-20:30:22.398934  [**] Null scan! [**] 65.2.208.87:0 -> MY.NET.99.39:0
12/26-20:30:22.764343  [**] Null scan! [**] 65.2.208.87:0 -> MY.NET.99.39:0
12/26-20:30:22.776960  [**] Null scan! [**] 65.2.208.87:0 -> MY.NET.99.39:0
12/26-20:30:22.956666  [**] Null scan! [**] 65.2.208.87:0 -> MY.NET.99.39:0
12/26-20:46:14.603134  [**] spp_portscan: portscan status from 65.2.208.87: 6 connections across 1 hosts: TCP(6),
UDP(0) STEALTH [**]
12/26-20:30:24.084146  [**] Null scan! [**] 65.2.208.87:0 -> MY.NET.99.39:0
12/26-20:46:16.691921  [**] spp_portscan: portscan status from 65.2.208.87: 4 connections across 1 hosts: TCP(4),
UDP(0) STEALTH [**]
12/26-20:30:28.465994  [**] Null scan! [**] 65.2.208.87:0 -> MY.NET.99.39:0
12/26-20:30:30.033092  [**] Null scan! [**] 65.2.208.87:0 -> MY.NET.99.39:0
12/26-20:46:18.885640  [**] spp_portscan: portscan status from 65.2.208.87: 1 connections across 1 hosts: TCP(1),
UDP(0) STEALTH [**]
12/26-20:46:21.192270  [**] spp_portscan: portscan status from 65.2.208.87: 1 connections across 1 hosts: TCP(1),
UDP(0) STEALTH [**]
12/26-20:46:23.285524  [**] spp_portscan: End of portscan from 65.2.208.87: TOTAL time(15s) hosts(1) TCP(12) UDP(0)
STEALTH [**]
12/26-20:46:23.449704  [**] spp_portscan: PORTSCAN DETECTED from 65.2.208.87 (STEALTH) [**]
12/26-20:30:39.333982  [**] Null scan! [**] 65.2.208.87:4 -> MY.NET.99.39:57702
12/26-20:46:25.436074  [**] spp_portscan: portscan status from 65.2.208.87: 1 connections across 1 hosts: TCP(1),
UDP(0) STEALTH [**]
12/26-20:46:27.575673  [**] spp_portscan: End of portscan from 65.2.208.87: TOTAL time(0s) hosts(1) TCP(1) UDP(0)
STEALTH [**]
```

Both of the logs above indicate that MY.NET.99.39 was probably an active target for both of these hosts which could be the reason why we noticed abnormal Fragmentation from these hosts.

Registration Information for **67.161.190.60**

```
@Home Network (NETBLK-NWRKNJ1-NJ-19)
  425 Broadway
  Redwood City, CA 94063
  US

  Netname: NWRKNJ1-NJ-19
  Netblock: 67.161.128.0 - 67.161.191.255

  Coordinator:
    Operations, Network  (HOME-NOC-ARIN)  noc-abuse@noc.home.net
    (650) 556-5599
```

Registration Information **for 65.2.208.87**

```
@Home Network (NETBLK-NWRKNJ1-NJ-8)
  425 Broadway
  Redwood City, CA 94063
  US

  Netname: NWRKNJ1-NJ-8
  Netblock: 65.2.208.0 - 65.2.223.255

  Coordinator:
    Operations, Network  (HOME-NOC-ARIN)  noc-abuse@noc.home.net
    (650) 556-5599
```

Defensive Recommendations

Its safe to block fragmentation at border routers/firewalls.

ICMP Source Quench

Description

An ICMP packet with Type=4 is a Source Quench Information packet. RFC 896 talks about the importance of Source Quench in complex networks. Source

Quench messages are sent by the receiver to let the sender know that its sending the data faster than receiver can process.

Problem

ICMP packets are regularly used by DoS (Denial of service) tools as a distructive tool to disrupt service on any server. Due to the nature of ICMP of being connectio nless this cannot be used as a Stimulus. But a large number of ICMP packets can make the kernel use up most of the CPU resource time effectively making it useless for anything else.

Dump

Source Addresses

```
 90513 MY.NET.5.13
   676 MY.NET.86.30
   111 130.149.1.3
    54 203.175.0.10
    43 198.161.103.53
    24 216.188.214.157
    14 62.180.14.9
    13 200.135.240.4
```

Destination Addresses

```
  2511 MY.NET.200.158
  2446 MY.NET.200.156
  2425 MY.NET.200.151
  2417 MY.NET.200.149
  2390 MY.NET.200.154
  2332 MY.NET.200.147
  2110 MY.NET.200.175
  2096 MY.NET.200.167
  2081 MY.NET.200.163
```

Destination of the Source Quench messages are evenly distributed which is normal. It is surprising to notice that MY.NET.5.13 generated most of the Source Quench messages which could only mean that lot of clients are sending data to this server. There is no reason of concern here.

Defensive Recommendations

Its safe to stop ICMP type=4 messages at border router/firewalls.

MISC traceroute

DESCRIPTION

Traceroute is a tool used to find out the path taken by a packet between any two IP addresses. A traceroute is detected by looking at the TTL value which should be either 1 or close to 1.

PROBLEM

Traceroute is used to gather information on the host and the routers in between. At times Low TTLs are used to evade firewall detection.

DUMP

Destination Addresses

```
  79237 MY.NET.140.9
    271 MY.NET.70.148
     11 MY.NET.1.8
      8 MY.NET.1.9
      6 MY.NET.97.174
      5 MY.NET.1.10
```

The source addresses were evenly distributed among approximate 100 addresses. A more careful analysis of logs indicates that these hosts were doing regular 10 minute interval traceroutes to the destination MY.NET.140.9.

DEFENSIVE RECOMENDATION

Allowing traceroutes to enter the network is considered risky. This can stopped by shutting down ICMP Time exceeded messages from leaving the network.

ACTIVITY ON 18245 -> 21536

DESCRIPTION

This is a very interesting dump. The OOS has quite a few packets which fit this profile, and whats surprising about this is that these are all supposed to be corrupted packets.
http://archives.neohapsis.com/archives/incidents/2001-01/0055.html
http://www.google.com/search?hl=en&q=port+18245

DUMP

```
oos.logs:12/22-01:48:26.418312 65.129.38.2:18245 -> MY.NET.253.114:21536
oos.logs:12/22-09:40:28.549128 65.129.33.89:18245 -> MY.NET.253.114:21536
oos.logs:12/22-11:58:34.413866 65.129.21.105:18245 -> MY.NET.253.114:21536
oos.logs:12/22-12:48:22.892210 65.129.52.45:18245 -> MY.NET.253.114:21536
oos.logs:12/22-23:27:57.456183 65.129.48.98:18245 -> MY.NET.253.114:21536
oos.logs:12/23-11:09:15.974118 65.129.32.4:18245 -> MY.NET.253.114:21536
oos.logs:12/23-11:32:50.399356 65.128.133.148:18245 -> MY.NET.60.14:21536
oos.logs:12/23-11:51:29.193186 65.129.41.99:18245 -> MY.NET.253.114:21536
oos.logs:12/24-13:38:41.570586 65.129.38.118:18245 -> MY.NET.11.4:21536
oos.logs:12/24-15:04:40.755071 65.129.29.16:18245 -> MY.NET.253.114:21536
oos.logs:12/24-15:54:14.581394 66.50.26.220:18245 -> MY.NET.253.114:21536
oos.logs:12/24-16:40:40.549022 65.129.31.168:18245 -> MY.NET.253.114:21536
oos.logs:12/24-18:18:03.315858 65.129.21.34:18245 -> MY.NET.253.114:21536
oos.logs:12/24-19:19:02.365924 66.50.49.113:18245 -> MY.NET.253.114:21536
oos.logs:12/24-22:29:31.510742 65.129.46.147:18245 -> MY.NET.253.125:21536
oos.logs:12/25-01:57:49.449545 65.129.57.114:18245 -> MY.NET.253.114:21536
oos.logs:12/25-11:29:18.123489 65.129.24.90:18245 -> MY.NET.11.4:21536
oos.logs:12/25-12:45:48.334746 65.129.16.140:18245 -> MY.NET.253.114:21536
oos.logs:12/25-19:03:02.069272 65.129.44.128:18245 -> MY.NET.253.114:21536
```

According to multiple source this kind of packet is generated by a faulty hardware device which generates IP packets without the TCP header and puts the TCP Data in the IP Data segment. When this happens to be a HTTP header , the GET request gets translated to source port of 18245 and destination port 21536.

DEFENSIVE RECOMMENDATION

None. This is not an intrusion.

SSH Brute Force Scanning

DESCRIPTION

The OOS dump and the scan dump, pointed out one particular IP which was extremely active. OOS dump usually only contains hosts which are trying to send illegimate traffic.

DUMP

```
12/25-21:50:46.405655 24.0.28.234:22 -> MY.NET.1.2:22
12/25-21:50:46.415952 24.0.28.234:22 -> MY.NET.1.3:22
12/25-21:50:46.521709 24.0.28.234:22 -> MY.NET.1.8:22
12/25-21:50:46.526144 24.0.28.234:22 -> MY.NET.1.9:22
12/25-21:50:46.568282 24.0.28.234:22 -> MY.NET.1.12:22
12/25-21:50:46.765077 24.0.28.234:22 -> MY.NET.1.20:22
12/25-21:50:46.769902 24.0.28.234:22 -> MY.NET.1.21:22
12/25-21:50:46.907141 24.0.28.234:22 -> MY.NET.1.27:22
12/25-21:50:46.936960 24.0.28.234:22 -> MY.NET.1.29:22
12/25-21:50:47.127930 24.0.28.234:22 -> MY.NET.1.38:22
12/25-21:50:47.240332 24.0.28.234:22 -> MY.NET.1.44:22
12/25-21:50:47.267402 24.0.28.234:22 -> MY.NET.1.45:22
12/25-21:50:47.404650 24.0.28.234:22 -> MY.NET.1.52:22

Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.66:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.67:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.69:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.72:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.73:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.74:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.79:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.86:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.87:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.93:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.95:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.96:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.100:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.104:22 SYNFIN ******SF
Dec 25 22:03:15 24.0.28.234:22 -> MY.NET.149.107:22 SYNFIN ******SF
```

This particular host initiated a scan on 25th December on almost an entire class B network. The scan was a SIN+FIN scan on port 22. Even without the volume, its easy to figure out that this was a scan just because the source port and destination ports are same. The fact that it scanned over 5000 different hosts in a matter of 15 seconds shows how desperate this particular attacker was.

PROBLEM

The attacker at this point is generating a list of hosts with SSH listening. By sending a SYN/FIN, its trying to go around week firewalls which all packets with FIN to go through even if there wasn't a previous connection on that socket. Once the attacker has a list of hosts which are listening on port 22, the next step would be to run a ssh exploit in script against all the possible target hosts generated from this scan.

Registration Information **for 24.0.28.234**

```
@Home Network (NETBLK-HOME-CORP-1)
  425 Broadway
  Redwood City, CA 94063
  US

  Netname: HOME-CORP-1
  Netblock: 24.0.16.0 - 24.0.31.255

  Coordinator:
    Operations, Network  (HOME-NOC-ARIN)  noc-abuse@noc.home.net
    (650) 556-5599
```

ACTIVITY FROM 210.125.178.52

DESCRIPTION

During analysis of OOS files, suspiciously high number of alerts originated from 210.125.178.52. The target MY.NET.163.15 could be called an active target since that was specifically probed during this scan.

OOS logs

```
12/22-02:48:31.500692 210.125.178.52:41989 -> MY.NET.163.15:0
12/22-02:48:34.509093 210.125.178.52:41989 -> MY.NET.163.15:0
12/22-02:48:38.504416 210.125.178.52:41990 -> MY.NET.163.15:1
12/22-02:48:48.476183 210.125.178.52:41991 -> MY.NET.163.15:2
12/22-02:48:52.471001 210.125.178.52:41992 -> MY.NET.163.15:3
12/22-02:48:55.467687 210.125.178.52:41992 -> MY.NET.163.15:3
.
.
.
12/22-03:08:21.926015 210.125.178.52:42173 -> MY.NET.163.15:171
12/22-03:08:28.954180 210.125.178.52:42174 -> MY.NET.163.15:172
12/22-03:08:35.953528 210.125.178.52:42175 -> MY.NET.163.15:173
12/22-03:08:52.933371 210.125.178.52:42177 -> MY.NET.163.15:175
12/22-03:09:13.960836 210.125.178.52:42180 -> MY.NET.163.15:178
12/22-03:09:20.967884 210.125.178.52:42181 -> MY.NET.163.15:179
12/22-03:09:31.975979 210.125.178.52:42183 -> MY.NET.163.15:181
12/22-03:09:41.976825 210.125.178.52:42184 -> MY.NET.163.15:182
12/22-03:09:46.019838 210.125.178.52:42185 -> MY.NET.163.15:183
```

Portscan logs confirmed the activity to be a special scan

```
Dec 22 02:48:34 210.125.178.52:41989 -> MY.NET.163.15:0 SYN 12****S* RESERVEDBITS
Dec 22 02:48:38 210.125.178.52:41990 -> MY.NET.163.15:1 SYN 12****S* RESERVEDBITS
Dec 22 02:48:48 210.125.178.52:41991 -> MY.NET.163.15:2 SYN 12****S* RESERVEDBITS
Dec 22 02:48:55 210.125.178.52:41992 -> MY.NET.163.15:3 SYN 12****S* RESERVEDBITS
Dec 22 02:48:59 210.125.178.52:41993 -> MY.NET.163.15:4 SYN 12****S* RESERVEDBITS
Dec 22 02:49:02 210.125.178.52:41993 -> MY.NET.163.15:4 SYN 12****S* RESERVEDBITS
Dec 22 02:49:06 210.125.178.52:41994 -> MY.NET.163.15:5 SYN 12****S* RESERVEDBITS
Dec 22 02:49:09 210.125.178.52:41994 -> MY.NET.163.15:5 SYN 12****S* RESERVEDBITS
Dec 22 02:49:37 210.125.178.52:41998 -> MY.NET.163.15:9 SYN 12****S* RESERVEDBITS
Dec 22 02:49:51 210.125.178.52:42000 -> MY.NET.163.15:11 SYN 12****S* RESERVEDBITS
Dec 22 02:49:55 210.125.178.52:42001 -> MY.NET.163.15:12 SYN 12****S* RESERVEDBITS
Dec 22 02:50:05 210.125.178.52:42002 -> MY.NET.163.15:13 SYN 12****S* RESERVEDBITS
Dec 22 02:50:30 210.125.178.52:42006 -> MY.NET.163.15:17 SYN 12****S* RESERVEDBITS
Dec 22 02:50:37 210.125.178.52:42007 -> MY.NET.163.15:18 SYN 12****S* RESERVEDBITS
Dec 22 02:50:47 210.125.178.52:42009 -> MY.NET.163.15:19 SYN 12****S* RESERVEDBITS
```

Snort alert log files also showed the activity. Interestingly not only does it say it detected it as a portscan, it also mentions that this particular set of flags also used for Queso fingerprinting.

```
12/22-03:02:29.243177  [**] spp_portscan: PORTSCAN DETECTED from 210.125.178.52 (STEALTH)
[**]
12/22-02:48:31.238039  [**] Queso fingerprint [**] 210.125.178.52:41989 ->
MY.NET.163.15:0
12/22-02:48:34.246123  [**] Queso fingerprint [**] 210.125.178.52:41989 ->
MY.NET.163.15:0
12/22-03:02:30.758813  [**] spp_portscan: portscan status from 210.125.178.52: 1
connections across 1 hosts: TCP(1), UDP(0) STEALTH [**]
12/22-02:48:38.241038  [**] Queso fingerprint [**] 210.125.178.52:41990 ->
MY.NET.163.15:1
12/22-03:02:32.274473  [**] spp_portscan: portscan status from 210.125.178.52: 1
connections across 1 hosts: TCP(1), UDP(0) STEALTH [**]
```

```
12/22-03:02:33.925355  [**] spp_portscan: End of portscan from 210.125.178.52: TOTAL
time(7s) hosts(1) TCP(2) UDP(0) STEALTH [**]
12/22-03:02:36.067832  [**] spp_portscan: PORTSCAN DETECTED from 210.125.178.52 (STEALTH)
[**]
12/22-02:48:48.211787  [**] Queso fingerprint [**] 210.125.178.52:41991 ->
MY.NET.163.15:2
1
```

PROBLEM

This particular scan lasted for about 15 minutes and only scanned one host. The scan was extremely noisy and if the host was alive and did receive the stimulus, it would have reported back sensitive port information to the attacker.

Registration Information for **210.125.178.52**

inetnum	210.124.0.0 - 210.127.255.255
netname	KRNIC-KR
descr	KRNIC
descr	Korea Network Information Center
country	KR
admin-c	HM127-AP, inverse
tech-c	HM127-AP, inverse
remarks	***
remarks	KRNIC is the National Internet Registry
remarks	in Korea under APNIC. If you would like to
remarks	find assignment information in detail
remarks	please refer to the KRNIC Whois DB
remarks	http://whois.nic.or.kr/english/index.html
remarks	***
mnt-by	APNIC-HM, inverse
mnt-lower	MNT-KRNIC-AP, inverse
changed	hostmaster@apnic.net 19981001
changed	hostmaster@apnic.net 20010606
source	APNIC

person	Host Master, inverse
address	Korea Network Information Center
address	Narajongkeum B/D 14F, 1328-3, Seocho-dong, Seocho-ku, Seoul, 137-070, Republic of Korea
country	KR

phone	+82-2-2186-4500
fax-no	+82-2-2186-4496
e-mail	hostmaster@nic.or.kr, inverse
nic-hdl	HM127-AP, inverse
mnt-by	MNT-KRNIC-AP, inverse
changed	hostmaster@nic.or.kr 20010514
source	APNIC

DEFENSIVE ACTION

If possible contact the network admin at korea and report the activity. Closely monitor MY.NET.163.15 and check it for possible attacks after the scan. If tripwire was running on the server, check for changes to critical files.

ACTIVITY FROM 62.211.247.3

DESCRIPTION

Port scanning logs shows this host to be doing heavy port scanning of almost the entire class B network owned by this organization. The entire probe took just over 20 minutes.

```
Dec 20 08:16:25 62.211.247.3:21 -> MY.NET.1.190:21 SYNFIN ******SF
Dec 20 08:16:25 62.211.247.3:21 -> MY.NET.1.199:21 SYNFIN ******SF
Dec 20 08:16:26 62.211.247.3:34917 -> MY.NET.1.199:21 SYN ******S*
Dec 20 08:16:25 62.211.247.3:21 -> MY.NET.1.201:21 SYNFIN ******SF
Dec 20 08:16:25 62.211.247.3:21 -> MY.NET.1.202:21 SYNFIN ******SF
Dec 20 08:16:25 62.211.247.3:21 -> MY.NET.1.203:21 SYNFIN ******SF
Dec 20 08:16:25 62.211.247.3:21 -> MY.NET.1.207:21 SYNFIN ******SF
Dec 20 08:16:25 62.211.247.3:21 -> MY.NET.1.211:21 SYNFIN ******SF
Dec 20 08:16:25 62.211.247.3:21 -> MY.NET.1.212:21 SYNFIN ******SF
Dec 20 08:16:25 62.211.247.3:21 -> MY.NET.1.213:21 SYNFIN ******SF
Dec 20 08:16:25 62.211.247.3:21 -> MY.NET.1.215:21 SYNFIN ******SF
Dec 20 08:16:26 62.211.247.3:21 -> MY.NET.1.218:21 SYNFIN ******SF
Dec 20 08:16:26 62.211.247.3:34916 -> MY.NET.1.197:21 SYN ******S*
Dec 20 08:16:26 62.211.247.3:21 -> MY.NET.1.220:21 SYNFIN ******SF
```

PROBLEM

The attacker is probing specifically for ftp servers on port 21. From the logs its clear that the attacker is probably running an integrated scanner + attacker. The clip of log above shows that the “SYN” Connection to host MY.NET.1.199 happened right after the SYN/FIN to the same host. The second packet the the host is most probably a different tool launched by the scanner which probably must have got back a reply to the SYNFIN probe. A which search for the SYN packets to port 21 shows that the attacker probably got replies back from 105 hosts, indicating 105 active FTP servers in the network.

Most port scanners launch the second tool only after the first scan is completed. This feature makes this scan standout of most other tools.

Registration Information for **62.211.247.3**

```
inetnum:     62.211.128.0 - 62.211.255.255
netname:     TINIT-ADSL-LITE
descr:       Telecom Italia
descr:       Accesso ADSL BBB
country:     IT
admin-c:     BS104-RIPE
tech-c:      BS104-RIPE
status:      ASSIGNED PA
remarks:     Please send abuse notification to abuse-bbb@telecomitalia.it
notify:      ripe-staff@telecomitalia.it
mnt-by:      TIN-MNT
changed:     net_ti@telecomitalia.it 20020213
source:      RIPE
route:       62.211.0.0/16
descr:       INTERBUSINESS
origin:      AS3269
remarks:     Please report spam/abuse notification to abuse@tin.it
notify:      network@cgi.interbusiness.it
mnt-by:      INTERB-MNT
changed:     network@cgi.interbusiness.it 20011029
source:      RIPE
person:      BBBEASYIP STAFF
address:     Via Val Cannuta, 250
address:     I-00100 Roma
address:     Italy
phone:       +39 06 36881
e-mail:      ripe-staff@telecomitalia.it
nic-hdl:     BS104-RIPE
notify:      ripe-staff@telecomitalia.it
changed:     net_ti@telecomitalia.it 20001019
source:      RIPE
```

DEFENSIVE RECOMMENDATION

Shutdown unused FTP servers. Patch up the existing ones. And make sure none of the Active FTP servers were compromised

ACTIVITY ON 8448 -> 38001

DESCRIPTION

This is one of the most suspicious traffic I noticed for which I had no explanation. Whats even more strange is that the source port 8448 is always outside the network and the destination port 38001 is always inside the network. A closer analysis of these port combinations came up with lots of alerts. Most of these were UDP alerts so it would be a safe guess that this protocol is UDP based. I also noticed a whole bunch of scans from hosts outside the network with non-standard flags in the packets. This could be an indication of OS fingerprinting, but could also be result of packet corruption.

DUMP - source and destination

```
 987 209.190.237.123:8448         987 MY.NET.70.134:38001
   415 216.106.173.146:8448       982 MY.NET.190.15:38001
   368 216.106.172.146:8448       444 MY.NET.88.155:38001
   187 216.106.172.147:8448       243 MY.NET.85.114:38001
   166 200.69.193.177:8448        150 MY.NET.104.220:38001
   114 66.77.13.108:8448          111 MY.NET.53.59:38001
   113 216.106.173.144:8448       111 MY.NET.53.53:38001
   102 216.109.69.164:8448         88 MY.NET.53.42:38001
    85 66.77.13.109:8448           87 MY.NET.116.47:38001
    78 216.106.173.147:8448        76 MY.NET.111.157:38001
    75 216.106.172.69:8448         64 MY.NET.146.42:38001
    68 216.109.69.158:8448         54 MY.NET.153.171:38001
    67 216.106.172.68:8448         54 MY.NET.152.164:38001
    61 216.109.69.161:8448         49 MY.NET.53.52:38001
    57 216.109.69.167:8448         44 MY.NET.177.59:38001
    54 66.77.13.110:8448           37 MY.NET.177.48:38001
    46 200.42.92.235:8448          36 MY.NET.152.215:38001
    45 66.77.13.106:8448           35 MY.NET.83.53:38001
    44 63.209.213.44:8448          31 MY.NET.86.23:38001
    44 195.92.252.254:8448         24 MY.NET.153.210:38001
    40 200.42.92.5:8448            21 MY.NET.98.161:38001
    37 209.10.56.39:8448           19 MY.NET.152.213:38001
    36 216.109.69.172:8448         18 MY.NET.190.14:38001
    33 61.132.222.12:8448          13 MY.NET.87.50:38001
    32 216.109.69.152:8448          9 MY.NET.104.201:38001
    31 216.109.69.166:8448          6 MY.NET.53.50:38001
    27 210.222.18.195:8448          5 MY.NET.97.174:38001
    24 198.5.135.93:8448            3 MY.NET.87.44:38001
    23 211.218.149.180:8448         2 MY.NET.98.113:38001
    21 66.77.13.112:8448            2 MY.NET.86.22:38001
    21 61.78.35.45:8448             2 MY.NET.82.133:38001
    19 66.137.106.177:8448          2 MY.NET.253.42:38001
    19 216.109.69.153:8448          1 MY.NET.97.209:38001
    19 216.10.244.152:8448          1 MY.NET.86.30:38001
    18 216.109.69.174:8448          1 MY.NET.75.145:38001
    18 216.106.172.144:8448         1 MY.NET.70.195:38001
    16 66.77.13.122:8448            1 MY.NET.70.134:9265
    16 66.77.13.120:8448            1 MY.NET.6.35:38001
    16 216.109.69.179:8448          1 MY.NET.6.34:38001
    13 66.77.13.132:8448            1 MY.NET.53.40:38001
    13 216.10.244.52:8448           1 MY.NET.190.15:16416
    13 216.10.244.153:8448          1 MY.NET.182.71:38001
    12 208.36.115.152:8448          1 MY.NET.151.95:38001
```

DUMP - Anomalous packets

```
scans.011219:Dec 19 07:06:08 195.121.225.241:8448 -> MY.NET.75.145:38001 NOACK *2***RS* RESERVEDBITS
scans.011219:Dec 19 08:15:49 65.43.118.85:8448 -> MY.NET.70.195:38001 NOACK 12***R*F RESERVEDBITS
scans.011219:Dec 19 13:11:49 66.137.106.177:8448 -> MY.NET.152.213:38001 NOACK 12U**RSF RESERVEDBITS
scans.011219:Dec 19 13:12:06 66.137.106.177:8448 -> MY.NET.152.213:38001 UNKNOWN 12UA***F RESERVEDBITS
scans.011219:Dec 19 13:12:09 66.137.106.177:8448 -> MY.NET.152.213:38001 NOACK 12**PRS* RESERVEDBITS
scans.011219:Dec 19 13:12:10 66.137.106.177:8448 -> MY.NET.152.213:38001 INVALIDACK 12UA*R** RESERVEDBITS
scans.011219:Dec 19 13:12:15 66.137.106.177:8448 -> MY.NET.152.213:38001 INVALIDACK 12UA*RS* RESERVEDBITS
scans.011219:Dec 19 13:12:15 66.137.106.177:8448 -> MY.NET.152.213:38001 UNKNOWN 12*AP**F RESERVEDBITS
scans.011219:Dec 19 13:12:18 66.137.106.177:8448 -> MY.NET.152.213:38001 NOACK 12U**R*F RESERVEDBITS
scans.011219:Dec 19 13:12:28 66.137.106.177:8448 -> MY.NET.152.213:38001 NULL ********
scans.011219:Dec 19 13:12:32 66.137.106.177:8448 -> MY.NET.152.213:38001 FIN *******F
scans.011219:Dec 19 13:12:34 66.137.106.177:8448 -> MY.NET.152.213:38001 SYNFIN ******SF
scans.011219:Dec 19 13:12:36 66.137.106.177:8448 -> MY.NET.152.213:38001 NOACK 12U*PR** RESERVEDBITS
scans.011219:Dec 19 13:12:52 66.137.106.177:8448 -> MY.NET.152.213:38001 NOACK ****PRS*
scans.011219:Dec 19 13:12:53 66.137.106.177:8448 -> MY.NET.152.213:38001 NOACK ****PRSF
scans.011219:Dec 19 13:12:59 66.137.106.177:8448 -> MY.NET.152.213:38001 INVALIDACK ***A**SF
scans.011219:Dec 19 13:13:21 66.137.106.177:8448 -> MY.NET.152.213:38001 NOACK *****RSF
scans.011219:Dec 19 13:13:29 66.137.106.177:8448 -> MY.NET.152.213:38001 INVALIDACK 12UA*R** RESERVEDBITS
scans.011219:Dec 19 13:13:43 66.137.106.177:8448 -> MY.NET.152.213:38001 NOACK **U*PRS*
scans.011219:Dec 19 13:13:45 66.137.106.177:8448 -> MY.NET.152.213:38001 NOACK **U*PRSF
scans.011219:Dec 19 13:13:58 66.137.106.177:8448 -> MY.NET.152.213:38001 NOACK **U*PR**
scans.011219:Dec 19 16:25:17 65.80.73.238:8448 -> MY.NET.111.157:38001 VECNA *2**P**F RESERVEDBITS
scans.011219:Dec 19 21:38:47 207.33.23.73:8448 -> MY.NET.6.34:38001 UNKNOWN *2*A***F RESERVEDBITS
scans.011220:Dec 20 21:32:42 207.33.23.73:8448 -> MY.NET.6.35:38001 NOACK 1*U*PR** RESERVEDBITS
scans.011220:Dec 20 22:07:08 207.33.23.73:8448 -> MY.NET.253.42:38001 INVALIDACK *2*APR*F RESERVEDBITS
scans.011220:Dec 20 22:07:12 207.33.23.73:8448 -> MY.NET.253.42:38001 INVALIDACK *2*A*RSF RESERVEDBITS
scans.011221:Dec 21 05:31:51 217.1.30.8:8448 -> MY.NET.111.157:38001 NOACK 12***R*F RESERVEDBITS
scans.011221:Dec 21 05:39:37 217.1.30.8:8448 -> MY.NET.111.157:38001 NOACK ****PR*F
scans.011221:Dec 21 05:40:43 217.1.30.8:8448 -> MY.NET.111.157:38001 INVALIDACK **UAP*SF
scans.011221:Dec 21 06:20:00 217.1.30.8:8448 -> MY.NET.111.157:38001 INVALIDACK 1*UA*R*F RESERVEDBITS
```

DEFENSIVE RECOMENDATION

The information provided in the logs is insufficient to come to a conclusion. It would be interesting to find out if this is a legitimate application. If it isn't then probably the first action item is to shutdown these two ports on the firewall and investigate the hosts which are active within the network.

KAZAA

DESCRIPTION

Kazaa is a file sharing protocol extensively used by internet users to share music files. Using Kazaa has been controversial for many reasons, one of which is the music copyright issue. However the real reason why a network administrator doesn't like it is because of wastage of precious internet bandwidth.

Kazaa usually runs on port 1214 and has been noticed to be generating a lot of alerts in the logs on this particular network. An quick analysis listed about 243 hosts as kazaa clients.

DUMP

```
    225 MY.NET.99.39
    218 MY.NET.88.162
    199 MY.NET.150.133
    191 MY.NET.157.105
    142 MY.NET.178.86
    132 MY.NET.70.134
    117 MY.NET.100.236
     72 MY.NET.75.145
     57 MY.NET.70.195
     39 MY.NET.10.115
     38 MY.NET.106.174
     17 MY.NET.150.220
     12 MY.NET.70.11
```

DEFENSIVE RECOMENDATION

For network performance reasons, Kazaa should be shutdown. This usually requires a policy to be implemented by higher management without which implementing such a drastic filter might create problems.

MISC source port 53 to <1024

DESCRIPTION

Its normal for a port scanner to use the same source and destination ports while scanning for hosts to attack. And though snort generated a lot of warning for this particular event, this could just be DNS to DNS lookups using port 53 as source and destination. In Bind the named.conf needs to have the following line to make this happen

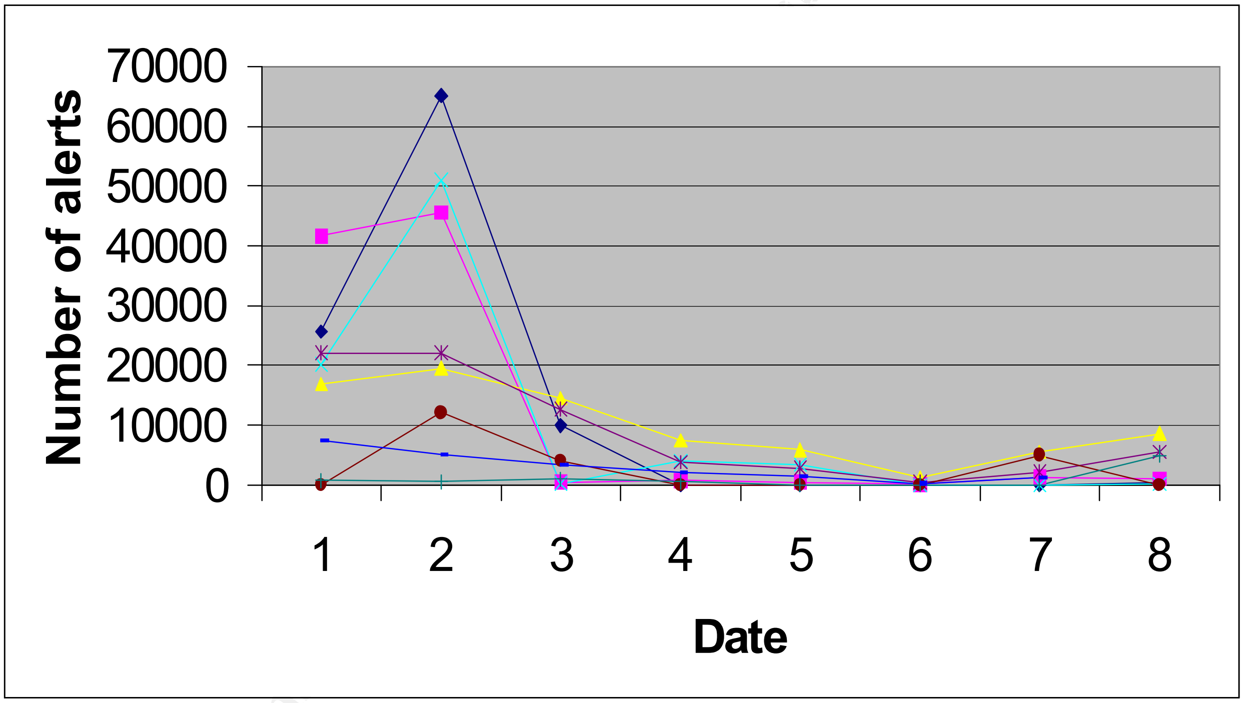
```
query-source port 53;
```

DEFENSIVE RECOMENDATION

None required without more information on specific attack.

Alerts pattern over the days

December	19	20	21	22	23	24	25	26	
Tiny Fragments - Possible Hostile Activity	25692	65156	10007	2	0	0	2	413	
ICMP Source Quench	41642	45613	478	835	509	164	1385	1117	
MISC traceroute	16918	19518	14587	7495	5970	1192	5540	8637	
MISC Large UDP Packet	20211	51030	295	4085	3423	0	16	220	
MISC source port 53 to <1024	22074	22080	12732	3876	2912	502	2174	5544	
SYN-FIN scan!	3	12173	4098	0	0	0	5026	0	
SCAN Proxy attempt	787	610	972	536	66	37	120	4961	
ICMP Destination Unreachable	7583	5226	3511	2046	1593	331	1339		
Total	134910	221406	46680	18875	14473	2226	15602	20892	475064
Traffic %	28.3983	46.60551	9.826	3.9731	3.0465	0.4686	3.284	4.3977	
Port Scans	374864	1609122	508872	131532	87479	75361	59609	155792	



Christmas eve is one of the historically most active day for intruders. One of the reasons why I selected this period for analysis is to see the impact of holiday season on internet users. And as history showed before, this time too the least traffic occurs on 24th.

Other interesting observations include that the traffic over weekends is significantly lesser than weekdays. This could have been because it was the week before Christmas, nevertheless the drop in traffic was substantial.

ANALYSIS PROCESS

1. The analysis process started by combining logs
2. Then generated sorted lists of ports and IP address from different kinds of logs
3. Analysis of these sorted lists, gave insight to problems in the network
4. Once we know problem IPs, I did grep on that IP in all the logs to see correlations between them.
5. Later logs were broken into per day format to see how scans increased/decreased as the date gets closer to Christmas.
6. The shell code to grep most of the logs are listed above.

REFRENCES

[Ref 1] OS Detection Methods/Concepts by J Lewis http://www.packetnexus.com/kb/greyarts/docs/981766898:16776.html
[Ref 2] Port numbers for Gnutella and Napster (6346/6699) - http://groups.yahoo.com/group/pluc/message/2306
[Ref 3] Port number for kazaa is TCP 1214, Gaming is UDP 28800/6112/27015 - http://www.securityfocus.com/archive/75/224368
[Ref 4] OS fingerprinting Database http://www.enteract.com/~lspitz/traces.txt
[Ref 5] Readme about SPADE http://www.silicondefense.com/software/spice/spicereadme.htm
[Ref 6] Ports used by Trojans http://www.simovits.com/sve/nyhetsarkiv/1999/nyheter9902.html
[Ref 7] Ports used by Applications http://advice.networkice.com/advice/Exploits/Ports/
[Ref 8] SANS: Blackgate Kit Analysis and Defence : http://www.incidents.org/react/unicode.php
[Ref 9] CERT Advisory CA-2001-02 Multiple Vulnerabilities in BIND http://www.cert.org/advisories/CA-2001-02.html
[Ref 10] TSIG bug in Bind http://www.kb.cert.org/vuls/id/196945
[Ref 11] Sans write up on Bind bugs [Lion worm] http://www.sans.org/y2k/lion.htm#Protection
[Ref 12] RFC 1035: http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1035.html [Section 2.3.4]
[Ref 13] CERT CDE Buffer overflow Advisory http://www.cert.org/advisories/CA-2001-31.html http://www.kb.cert.org/vuls/id/172583
[Ref 14] ISS CDE Advisory http://xforce.iss.net/static/7396.php
[Ref 15] How to disable CDE dtspcd http://cert.uni-stuttgart.de/ticker/article.php?mid=564
[Ref 16] CERT talks about resurgence of 2 year old exploit tcp/6112 exploit http://it.mycareer.com.au/breaking/2001/11/13/FFXKNAJJYTC.html
[Ref 17] Security Focus talks about dtspcd bug http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=3517
[Ref 18] Report about SYN-FIN Scan with same source dest port and TTL of approx 36 hops. http://www.sans.org/y2k/practical/Haruna_Isa.txt
[Ref 19] SSH crc32 compensation attack detector http://razor.bindview.com/publish/advisories/adv_ssh1crc.html , http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0144
[Ref 20] David Dittrich's review of X2 SSH exploit http://www.linuxsecurity.com/articles/intrusion_detection_article-4002.html
[Ref 21} RFC 896 http://deesse.univ-lemans.fr:8003/Connected/RFC/896/index.html
[Ref 22] Net Metropolitan Trojan http://www.blackcode.com/trojans/details.php?id=873
[Ref 23] Coordinated scan http://www.gcn.com/archives/gcn/1998/October19/32a.htm
[Ref 24] Shealth Coordinated Attack http://www.nswc.navy.mil/ISSEC/CID/Stealth_Coordinated_Attack.html
[Ref 25] http://www.insecure.org/nmap/nmap_doc.html#fin
[Ref 26] Detecting coordinated attacks/probes: http://secinf.net/info/misc/coord.html
[Ref 27] Abnormal packets generated by faulty devices http://archives.neohapsis.com/archives/incidents/2001-01/0055.html http://www.google.com/search?hl=en&q=port+18245
[Ref 28] Gregory Lajon GCIA **http://www.giac.org/practical/Gregory_Lajon_GCIA.doc**
[Ref 29] Thomas Rodriquez GCIA **http://www.giac.org/practical/Thomas_Rodriguez_GCIA.doc**
[Ref 30] Philipp Stadler GCIA **http://www.giac.org/practical/Philipp_Stadler_GCIA.doc**