John Jenkinson

Intrusion Detection In Depth
GCIA Practical Assignment
Version 3.0 (revised August 13, 2001)

Assignment 1 – Describe the State of Intrusion Detection

Various Issues with Network Intrusion Detection Systems (NIDS)

Introduction
The who, what, when, where, why, which and how of some aspects of Network Intrusion Detection.

Who
You. Everyone is affected to some degree by Network Intrusion Detection Systems. Those of us charged with NIDS responsibilities are affected for the obvious reasons. For others with no direct dealings with networks, computers, or intrusions the affect is there just the same. Computer networks have made their way into all our lives. Computer networks drive the economy. If your live is devoid of economic dependence on computer networks by a spartan existence using bartering, you probably rely on your country's military defense and/or governmental services like public schools, birth records, mail, etc. Our lives now depend on these computer networks doing what they do as they were designed (or something close) *without* interference or alteration from unauthorized sources. NIDS, as part of a security strategy deploying defense in depth, help to assure those computer network systems function as intended.

What
Network Intrusion Detection Systems means different things to different people. For this paper we will use Edward Amoroso's[1] definition:

Intrusion detection is the process of identifying and responding to malicious activity targeted at computing and network resources.
To paraphrase his expansion on that definition:
"process" first and foremost – one that involves technology, people and tools. This is important because processes involve time and interaction between entities, and many of the hard problems in intrusion detection will stem from this inherent interaction. This emphasis on process also illustrates that intrusion detection will never be implemented as a plug-and-play black box solution. People will always be involved in the process.
"identifying" The identification of an intrusion can be done before, during, or after the target malicious activity proceeds. If the identification is done before the activity proceeds, then the activity might be prevented and any potentially damaged assets might be salvaged. If the identification is done during the occurrence of the malicious activity, then decisions must be made about whether to allow target activity to proceed and whether to create alarms (which might alert the attacker). Finally, if the identification of an intrusion is done after the malicious

activity completes, then questions of what damage might have been done and how such actions could have occurred must be addressed.
"responding" Considerations for the response include whether the response should terminate service, whether the response is targeted at catching the attacker, and whether the response includes counterattacks, as in an information warfare context.
"malicious activity" The security-relevant actions by people who intend to do harm.

Network based IDS are usually signature based, i.e. they look for patterns in the packets transversing the network. Some NIDS also use or add the capability of detecting abnormal usage patterns, i.e. the normal behavior of the network is known and traffic patterns outside this normal behavior creates the alert. Such behavior patterns are usually better suited for host based intrusion detection systems. It is easier to discern that a user whose access is normally the workday is accessing the network at 3am by host logs than by network traffic.

When

As soon as possible. After the tragic events of 11 September 2001 we know we are not dealing with just script kiddies any more. The National Infrastructure Protection Center issued an advisiory on September 14 to expect an "upswing in incidents". The alert was reissued October 11.

Where

Limiting our scope to network based IDS the question becomes where to place NIDS probe points. We have gone from the days of a thick coax ethernet cable with vampire taps drilled to the cable core - to ethernet switches for most of our network infrastructure. This removed the snaking of the coax cable shared media and provided increased efficiencies by virtually connecting the two communicating ports on the switch. This provides some security as the traffic now flows port to port and a sniffer can not be placed on the coax cable to monitor all the ethernet traffic. Add Virtual LAN (VLANs), topologies that involve several switches, physical security of ethernet switches, zealous guarding of the spanning port by the networking groups and the fact that spanning ports can be overrun on busy switches - you are soon looking for better placement of the probe points. Cisco has a IDS module that plugs into the backplane of some of their ethernet switches which runs at wire speeds and overcomes some of the limitations of spanning port[9]. The issues with that solution are the expense and the problem if the network topology has more than one ethernet switch. Depending on your network topology there are probably other probe points to consider. We chose the link between the border router(s) and the outmost firewall. We later added a probe point just inside the innermost firewall and very recently added a managed Dragon sensor at the same inside point. This gives us defense in depth and multiple NIDS systems watching the same traffic. All NIDS have strengths and weaknesses, by placing multiple systems at the same point we hope to negate some of those weaknesses and add strength to the intrusion detection task. Having systems on both sides of the firewalls gives a sanity check to the firewall rule set. If a rule should stop traffic with a signature but the interior NIDS systems indicate such traffic has made it through then investigation can be made to determine if the firewall is at fault or the intruder is using mechanisms to bypass firewall rules. We also catch instances where a firewall ruleset build and install was in error.

Another factor to consider in probe placement is encryption. Systems that communicate with Secure Socket Layer (SSL) or systems using encrypted tunnels like Virtual Private Networks (VPN) will have portions of their packets encrypted. NIDS that do content matching for the signature will not find a match on a packet with the actual packet payload if that payload is encrypted. Likewise VPN encrypts the packet below the IP layer. So false positives will abound for scanned items like illegal TCP flags due to the VPN tunnel encryption. In such cases you can do some host based IDS on the VPN server and add a probe point beyond the VPN server to capture traffic from the VPN server/gateway to the application server. You may also consider decrypting the traffic, doing the intrusion detection function, re-encrypt the traffic before passing it on to the VPN server.

Several vendors make taps or network splitters designed for NIDS eg NetOptics, Inc. and Shomiti Systems which was recently purchased by Finsar Systems.

Which

There are a lot of systems to choose from. Michael Sobirey's Intrusion Detection Systems page lists 92 host and network IDS as of June 2000. More probably have been added to the market since and a few on that list are probably out of business. Stephen Northcutt's "Network Intrusion Detection – An Analyst's Handbook" [3] gives a good list of available systems as well as analysis of those listed. A recent comparison of several systems in tests at DePaul University done by Network Computing Magazine is given here.

As there are so many systems to choose from, the capabilities and architecture vary widely, and the price for these systems is significant; it is best to do a lot of research and evaluations in your environment before purchase and implementation. You should also be aware of developing standards for correlation and interoperability between vendor products – both IDS and non-IDS. Common Intrusion Detection Framework (CIDF) and Common Intrusion Specification Language (CISL) are now proposed as the Intrusion Detection Working Group (IDWG) of the IETF. Checkpoint has their OPSEC standard[11]. A chapter covering these proposed solutions is in Stephen Northcutt's[3] book.

How

In most networked systems now traffic and traffic rates are way up from those of a few years ago. Computer systems are faster, have more disk and memory, and run faster network interface cards. GigE and 100Mbit full duplex are the norm with applications that take full advantage of these advances in technology. With the traffic and traffic rates up, the strain on NIDS is greater. Of course the intruder has benefited from the same technology. While the technology advances have benefited NIDS as well, there are other issues brought by the traffic increases. One is capacity to store all that traffic for full analysis. A port scan, as an example, might occur over several days or weeks. Thus to catch such a port scan the traffic for that period of time must be kept available for analysis and the NIDS must rerun the port scan analysis over that period. For a port scan the traffic storage could be reduced to just the IP and protocol headers. Now consider an event like BIND version query. Now we should analyse traffic, including the packet payload, from that source IP and that source IP network for some period of time to see if any subsequent traffic is generated to exploit any vulnerabilities found by the version query. Other examples abound, but the requirement would seem to be all the packets with the entire payload for a long time. The more probe points in the enterprise, the more data is collected, and correlation across the enterprise multiplies the resources needed in terms of both storage of the packets and the correlated analysis of that packet data. Sophisticated techniques like backdoor detection[6] and detection of stepping stones[7] also require a large volume of data for all traffic. This just covers the detection of intrusions. For the use of that packet data in legal proceedings then the added tasks of protecting the NIDS data from alteration, tracking the data, and keeping the ability to sanitize that data while preserving the ability of the data to serve in legal proceedings becomes a requirement that must be dealt with by policies and procedures before the resources are needed for those proceedings[13].

On the other hand, you now have a system capable of packet payload capture for the network you are monitoring for intrusion detection. Now you have the issue of privacy. This data can give who visits what web sites, what employees put into health care forms, what stock prices the company executives track, etc. You also have all this data available for discovery in any legal proceedings, meaning you could have to hand over this data to lawyers on the opposing side in these proceedings. Privacy was the overriding issue until September 11, now national security brings the question of balance of these two issues[4,5]. This means that your security policy (you do have a security policy AND it covers your intrusion detection systems, right?) must be intergraded with privacy policies (both employee and partner) and record retention policies. If you don't have an employee privacy policy you probably have governmental regulations similar to the United States's Electronic Communications Privacy Act (ECPA). For multi-national companies need to be aware of not only laws in all the nations they operate, but also the laws and regulations concerning how privacy issues are handled across national boundries[14].

As with most things, it is a question of balance. We configure our snort NIDS machines to very high security. Only the administrators with NIDS responsibilities are given accounts on these machines. We use agrus for our packet capture facility since it does not capture packet data. When we need to have packet capture we run snort in packet capture mode until the problem is resolved. Data is encrypted and the data is not allowed to leave the machine by policy. This gives most of the capability needed that utilizes full packet capture and does address the privacy issues.

Another important issue in NIDS deployment and correlation is time stamping. All systems must be synchronized to an accurate reference clock. This is important even if your company is in a single location with no branch or international offices since the intrusion data you collect will benefit the intrusion effort at large if sent to a Computer Incident Response Team (CIRT) like SANS incidents.org.

The resources needed by the system running the NIDS need careful attention as well. Statefull TCP packets require a large amount of memory. Keeping logs requires a fast disk I/O subsystem. Of course the network interface card needs to handle the traffic load. With NIDS doing context checking and/or correlation the CPU needs to be fast enough to handle the analysis. Reassembly of fragmented packets and any decryption of encrypted traffic by the NIDS burden the CPU as well. Over sizing the system for NIDS has the advantage of surviving to some degree an attempt by an intruder to overload the NIDS.

Another aspect to consider is tracking the incidents[12]. Addressing each significant incident is important, but tracking elements of each incident like incident source and resolution will give you a sense of the health of your network.

Once you have a detect, then what? Some NIDS send alerts via pager, Simple Network Management Protocol (SNMP) traps, or create a popup display on a management console. The human element then comes into play in evaluating the event and taking the appropriate actions. Some work has been done on systems which take actions depending on the event, they can modify firewall rules or router ACLs to disable a problem node as an example.

References

1. Amoroso, Edward. “Intrusion Detection”. Intrusion.Net Books. 1999
2. Escamilla, Terry. “Intrusion Detection”. Wiley & Sons. 1998
3. Northcutt, Stephen. “Network Intrusion Detection An Analyst’s Handbook”. New Riders. 1999
4. Sanborn, Stephanie; Linderholm, Owen; and Moore, Cathleen. “Privacy concerns raised”. Infoworld. 09.19.01
5. Gartner Research. “Caught in the Net: Online monitoring and employee privacy”. TechRepublic article.
6. Zhang, Yin; Paxson, Vern. “Detecting Backdoors”. Proceedings 9th USENIX Security Symposium. USENIX
7. Xhang, Yin; Paxson, Vern. “Detecting Stepping Stones”. Proceedings 9th USENIX Security Symposium. USENIX
8. Duncan, Jim; Farrow, Rik. “Handling Computer and Network Security Incidents”. Tutorial 9th USENIX Security Symposium, USENIX
9. Yocom, Betsy; Brown, Kevin; Van Derver, Dan. “Cisco offers wire-speed intrusion detection”. Network World. Dec 18, 2000
10. “New Directions in Intrusion Detection”. Information Security Magazine. Aug, 2001
11. Fratto, Mike. “OPSEC Sets Standard for Integration”. Network Computing Magazine December 4, 2000
12. Frandsen, Mike. “Incident Tracking”. SANS Institute Information Security Reading Room
13. Ceresini, Tom. “Maintaining the Forensic Viability of Logfiles”. SANS Institute Information Security Reading Room
14. Thibodeau, Patrick. “Foreign Laws Alter IT Privacy Policies”. ComputerWorld.

Assignment 2 – Network Detects

A Description of our network:

The network consists of a local LAN and a company WAN behind two Checkpoint Firewall-1 Version 4.0 SP6 firewalls. There is an extranet between those firewalls with connections to other company's networks via dedicated routers. The internal LAN/WAN accesses the Internet via proxy servers which Network Address Translate (NAT) the internal IP address range to a subnet of the Internet Service Provider (ISP).

A managed Virtual Private Network (VPN) service exists in the extranet as do several other servers.

A Solaris 8 Sun machine running argus Version 2.0.1.beta.1 and snort Version 1.8-beta4 (Build18) is on a read-only interface on the wire between the Internet router and the outermost firewall. A second Solaris 8 Sun machine configured the same way was added to the WAN link recently.

For this paper the IP addresses for the ISP provided subnet will be denoted as a.b.c.x/28. The Internet router knows of 9 addresses in use for this subnet.

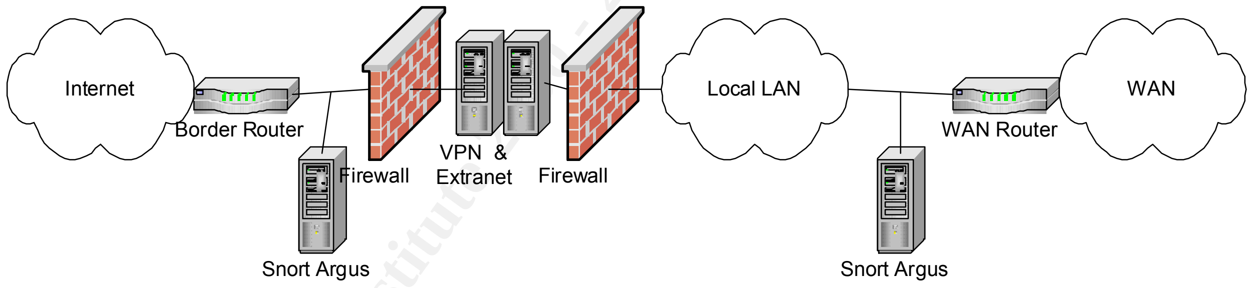
The company LAN/WAN will be denoted as w.x.y.z/29.

The extranet will be denoted as j.k.l.m/28.

Internet IP addresses will be the actual addresses unless otherwise noted.

Local time is GMT-9.

Both snort and argus output formats have changed from the previous versions, as has the snort rule format.



Snort is run in daemon mode via a command similar to:

snort –Afull –b –c <config file> -d –D –i <interface> -l <log subdir> -e –o

This gives:

- -A Alert mode as full, thus an alert to the alert file with IP header information as well
- -b log packets in tcpdump format
- -c configuration file to use to define plugins, rules and other configuration options. where relevant to the detect, the appropriate portion of the configuration file will be given
- -d dump the application layer data
- -D run in daemon mode
- -i interface specifying the read-only interface probing the Internet connection
- -l logfile location the alert file, the portscan.log and the binary tcpdump files
- -e log and display the link layer packet information
- -o change rule order to Pass->Alert->Log from Alert->Pass->Log

Snort's ruleset is as distributed from snort.org with a few additions from whitehats.com.

Argus is run in daemon mode as well with a command similar to:
argus –w <logfile> -i <interface>
thus specifying the log file location and name and the network interface.

Assignment 2 – Network Detects

Detect 1

snort alert file
```
[**] SCAN SYN FIN [**]
[Classification: Attempted Information Leak] [Priority: 3]
06/29/01-02:10:36.944959 0:E0:1E:83:D5:48 -> 8:0:20:D0:BB:23 type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.18:111 TCP TTL:16 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS198]

[**] SCAN SYN FIN [**]
[Classification: Attempted Information Leak] [Priority: 3]
06/29/01-02:10:37.013162 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.21:111 TCP TTL:16 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS198]

[**] SCAN SYN FIN [**]
[Classification: Attempted Information Leak] [Priority: 3]
06/29/01-02:10:37.123839 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.23:111 TCP TTL:16 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS198]

[**] SCAN SYN FIN [**]
[Classification: Attempted Information Leak] [Priority: 3]
06/29/01-02:10:37.124335 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.22:111 TCP TTL:16 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS198]

[**] SCAN SYN FIN [**]
[Classification: Attempted Information Leak] [Priority: 3]
06/29/01-02:10:37.127274 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.24:111 TCP TTL:21 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS198]

[**] SCAN SYN FIN [**]
[Classification: Attempted Information Leak] [Priority: 3]
06/29/01-02:10:37.278142 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.19:111 TCP TTL:21 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS198]

[**] SCAN SYN FIN [**]
[Classification: Attempted Information Leak] [Priority: 3]
06/29/01-02:10:37.283233 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.25:111 TCP TTL:16 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS198]

[**] SCAN SYN FIN [**]
[Classification: Attempted Information Leak] [Priority: 3]
06/29/01-02:10:37.584156 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.20:111 TCP TTL:21 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS198]
```

[**] Name of Alert [**]
[Classification given to this alert in the rule definition]   [Priority: Priority rank given by rule definition]
date-time   Source MAC address <direction>   Destination MAC address   type: IP   len: 60 bytes
Source IP Address : Port   <direction>   Destination IP Address: Port   TCP Protocal Type of Service   IP Packet Length   Datagram Length
TCP Flags   Sequence Number   Acknowledge Number   Window size   TCP datagram length
[Notes   defined in rule definition]

The rule for this alert from the scan.rules file:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN SYN FIN";flags:SF;
reference:arachnids,198; classtype:attempted-recon;)
Note: Line wrapped for readability
```

The rule is in two parts. The rule header and rule options (if any) enclosed in parentheses

alert	Send to alert file
tcp	Packet protocol
$EXTERNAL_NET	snort variable specifying any external net address defined as !$HOME_NET on our network
any	any port
$HOME_NET	snort variable specifying our internal net as comma separated CIDR (Classless Inter-Domain Routing) notation
any	any port
msg	message text to name or identify rule
flags	TCP flag bits Syn and Fin in this case
reference	Location to seek more information arachnids database at whitehats.com
classtype	Text added to alert

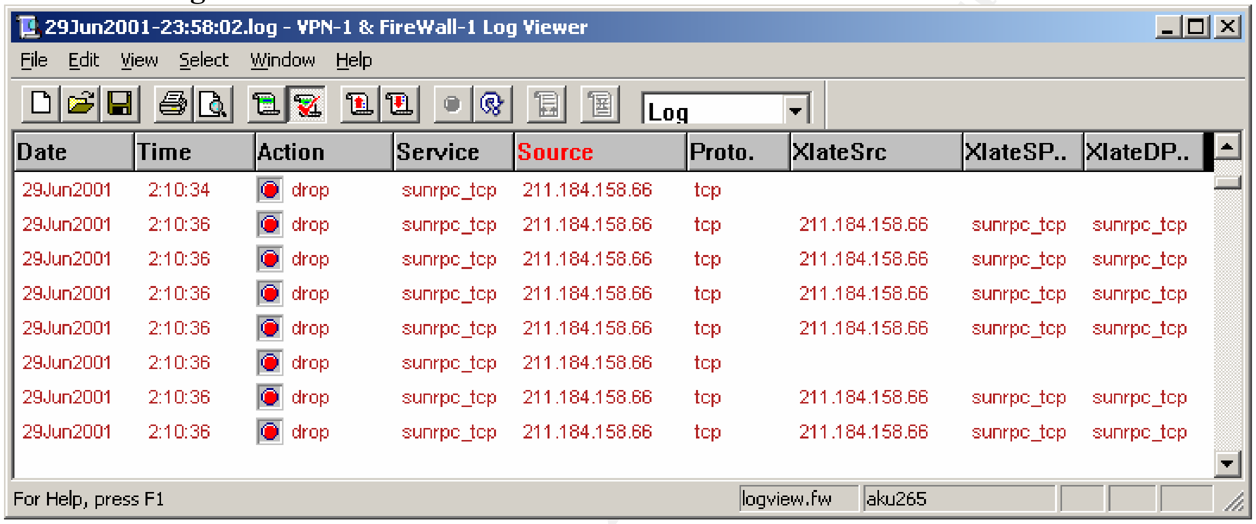
snort binary tcpdump

snort –dev –r <snort_binary_logfile>

-d   dump application layer data
-e   display link level packet headers
-v   verbose
-r   read tcpdump formatted logfile

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

06/29/01-02:10:36.944959 0:E0:1E:83:D5:48 -> 8:0:20:D0:BB:23 type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.18:111 TCP TTL:16 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20

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

06/29/01-02:10:37.013162 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.21:111 TCP TTL:16 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20

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

06/29/01-02:10:37.123839 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.23:111 TCP TTL:16 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20

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

06/29/01-02:10:37.124335 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.22:111 TCP TTL:16 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20

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

06/29/01-02:10:37.127274 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.24:111 TCP TTL:21 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20

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

06/29/01-02:10:37.278142 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.19:111 TCP TTL:21 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/29/01-02:10:37.283233 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.25:111 TCP TTL:16 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20

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

06/29/01-02:10:37.584156 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
211.184.158.66:111 -> a.b.c.20:111 TCP TTL:21 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x3F1E3DD9  Ack: 0x2C2F6F87  Win: 0x404  TcpLen: 20

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

Firewall-1 Log



Date	Time	Action	Service	Source	Proto.	XlateSrc	XlateSP..	XlateDP..
29Jun2001	2:10:34	● drop	sunrpc_tcp	211.184.158.66	tcp			
29Jun2001	2:10:36	● drop	sunrpc_tcp	211.184.158.66	tcp	211.184.158.66	sunrpc_tcp	sunrpc_tcp
29Jun2001	2:10:36	● drop	sunrpc_tcp	211.184.158.66	tcp	211.184.158.66	sunrpc_tcp	sunrpc_tcp
29Jun2001	2:10:36	● drop	sunrpc_tcp	211.184.158.66	tcp	211.184.158.66	sunrpc_tcp	sunrpc_tcp
29Jun2001	2:10:36	● drop	sunrpc_tcp	211.184.158.66	tcp	211.184.158.66	sunrpc_tcp	sunrpc_tcp
29Jun2001	2:10:36	● drop	sunrpc_tcp	211.184.158.66	tcp			
29Jun2001	2:10:36	● drop	sunrpc_tcp	211.184.158.66	tcp	211.184.158.66	sunrpc_tcp	sunrpc_tcp
29Jun2001	2:10:36	● drop	sunrpc_tcp	211.184.158.66	tcp	211.184.158.66	sunrpc_tcp	sunrpc_tcp

29Jun2001-23:58:02.log - VPN-1 & FireWall-1 Log Viewer
File Edit View Select Window Help
Log
For Help, press F1
logview.fw
aku265

1. Source of Trace: Our network

2. Detect was generated by: snort intrusion detection and Firewall-1

3.Probability the source address was spoofed: Low. This is a reconnaissance probe and the attacker wants the result returned. There is a slight possibility the source node is masquerading as the source IP if that IP address is not currently in use or is down for some reason. If the site was responding to ICMP traceroute we could determine the current number of hops away from our network. If the characteristics of the site could be also determined we can make a good guess at the initial time to live (ttl) of the site. A comparison of the initial ttl minus the ttl at arrival compared to the number of hops from the traceroute would be a better indicator of the probability of spoofing.

4. Description of attack: As we see all 9 of our addresses on this subnet probed we can assume this is a SYN/FIN scan for sunrpc_tcp or portmapper. SYN is a tcp header flag indicating the request for the start of a three-way handshake. FIN is a tcp header flag indicating the request for the graceful termination of an established TCP session. SYN and FIN in the same packet is not a normal occurrence which usually indicates a crafted packet. Normal requests to the portmapper service come from ephemeral ports on the client. In these packets the source port is 111 as well. More indication of a crafted packet and also indicates the source machine is probably not running portmapper. This could mean a non-UNIX machine or a UNIX machine without the portmapper service configured or running. The sequence numbers are the same for each of the packets. Again not a normal occurrence and yet another indication of crafted packets, the same with the acknowledge number. The datagram length of 40, the IP length of 20, and the tcp length of 20 would be normal for a SYN or a FIN packet. If any data was part of the packet it could possibly provide more information on the method or intent of the attack.

5. Attack mechanism: The consensus on the Internet is that synscan is the tool with this attack signature. Another possibility is Idlescan. Synscan will drop the initial SYN/FIN connection if it receives a reset (RST) and then attempt a connect as a normal connection. The tool can be configured to go to any port/service but does have the ID of 39426, window size of 1028, etc. The reported ttl is set to 42 so the source is about 30 hops from our network. So why is this scan after portmapper or sunrpc_tcp? SANS Institute's incidents web page shows portmapper to be one of the top 3 ports reported in terms of incidents. Portmapper is the key to several portmapper exploits AND provides information on many remote procedure call (rpc) exploits.
Using whois that IP address shows to be Haenam Songji Elementary School in Korea. At the time of the scan the local time in Korea would have been 7pm Saturday June 30. I do not know the structure of the school systems in Korea, but I doubt the school was in session at that local time.

6. Correlations:
http://www.sans.org/y2k/111600.htm
http://www.sans.org/y2k/112700-1400.htm
Search SANS for 39426, an internet search for SYN/FIN portmapper, or search http://www.incidents.org/

7. Evidence of active targeting: As this scan did hit all our viable addresses on the subnet this would appear to be a scan of at least that subnet. The other subnets of that class C address space are not allocated to us so we cannot determine how wide this scan might be. As the firewall dropped all of these packets the attacker apparently moved on to other sites as we saw no other traffic from this IP address or this subnet in any time period before or after the scan.

8. Severity: -3
(Critical + Lethal) – (System – Net Countermeasures)
(4+3) – (5+5)
Critical 4 - Servers in this subnet are our interface to the Internet
Lethal 3 - If portmapper was running, portmapper reconnaissance can lead to several other methods/means of attack
System 5 - Machines in this subnet do not run portmapper
Countermeasures 5 – scan packets dropped and logged by firewall rules

9. Defensive recommendations: Continued monitoring of snort alerts and firewall logs. Consider secure portmapper and IPSec in filter mode.

10. Multiple choice test question:
Which packet characteristic should bring similar packets to your attention?

a) SYN/FIN tcp flags both set
b) Source and destination port of 111
c) Ack numbers the same for each packet in the scan
d) IP sequence numbers the same for each packet in the scan
e) all of the above

Answer: e

Detect 2

snort alert file

```
[**] ICMP Nmap2.36BETA or HPING2 Echo  [**]
[Classification: Attempted Information Leak] [Priority: 3]
06/25/01-22:12:07.544973 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
209.193.18.199 -> a.b.c.26 ICMP TTL:44 TOS:0x0 ID:60587 IpLen:20 DgmLen:28
Type:8  Code:0  ID:50255   Seq:256  ECHO
[Xref => http://www.whitehats.com/info/IDS162]

[**] ICMP Nmap2.36BETA or HPING2 Echo  [**]
[Classification: Attempted Information Leak] [Priority: 3]
06/25/01-22:12:13.585159 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
209.193.18.199 -> a.b.c.26 ICMP TTL:44 TOS:0x0 ID:26002 IpLen:20 DgmLen:28
Type:8  Code:0  ID:50255   Seq:512  ECHO
[Xref => http://www.whitehats.com/info/IDS162]

[**] ICMP Nmap2.36BETA or HPING2 Echo  [**]
[Classification: Attempted Information Leak] [Priority: 3]
06/25/01-22:12:25.646034 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
209.193.18.199 -> a.b.c.26 ICMP TTL:44 TOS:0x0 ID:25274 IpLen:20 DgmLen:28
Type:8  Code:0  ID:50255   Seq:1024  ECHO
[Xref => http://www.whitehats.com/info/IDS162]
```

snort log file

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

06/25/01-22:12:07.544973 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
209.193.18.199 -> a.b.c.26 ICMP TTL:44 TOS:0x0 ID:60587 IpLen:20 DgmLen:28
Type:8  Code:0  ID:50255   Seq:256  ECHO

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

06/25/01-22:12:13.585159 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
209.193.18.199 -> a.b.c.26 ICMP TTL:44 TOS:0x0 ID:26002 IpLen:20 DgmLen:28
Type:8  Code:0  ID:50255   Seq:512  ECHO

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

06/25/01-22:12:25.646034 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
209.193.18.199 -> a.b.c.26 ICMP TTL:44 TOS:0x0 ID:25274 IpLen:20 DgmLen:28
Type:8  Code:0  ID:50255   Seq:1024  ECHO

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

The rule for this alert from the icmp.rules file:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Nmap2.36BETA or HPING2 Echo ";itype:8;dsize:0;
reference:arachnids,162; classtype:attempted-recon;)
```

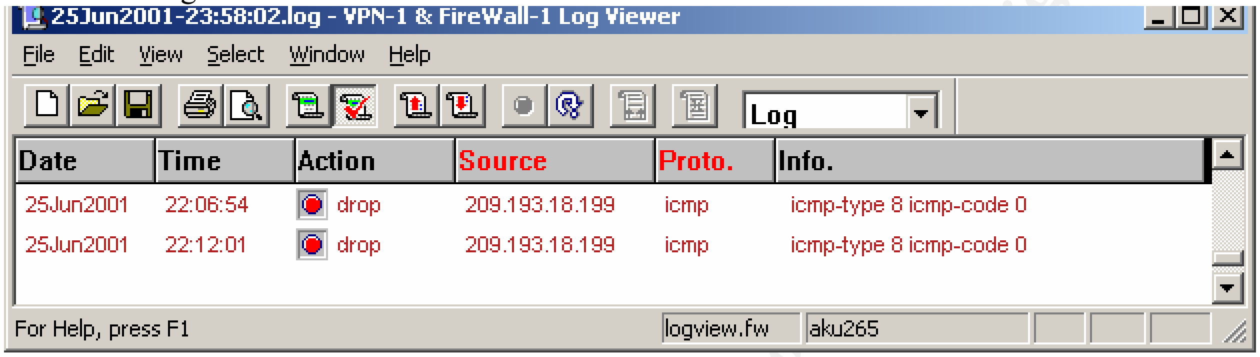
Note: Lines wrapped for readability

The rule is in two parts. The rule header and rule options in parenthesis

alert	Send to alert file
icmp	Packet protocol
$EXTERNAL_NET	snort variable specifying any external net address defined as !$HOME_NET on our network
any	place holder usually specifies port icmp has no port in header so it is ignored for icmp rules

$HOME_NET	snort variable specifying our internal net as a comma separated CIDR notation
any	place holder
msg	message text to name or identify rule
itype	icmp message type
dsize	datagram size
reference	Location to seek more information arcachnids database at whitehats.com
classtype	Text added to alert

Firewall-1 Log


25Jun2001-23:58:02.log - VPN-1 & FireWall-1 Log Viewer
File Edit View Select Window Help
Log
For Help, press F1
logview.fw
aku265

Date	Time	Action	Source	Proto.	Info.
25Jun2001	22:06:54	drop	209.193.18.199	icmp	icmp-type 8 icmp-code 0
25Jun2001	22:12:01	drop	209.193.18.199	icmp	icmp-type 8 icmp-code 0

1. Source of Trace: Our network

2. Detect was generated by: snort intrusion detection and Firewall-1

3. Probability the source address was spoofed: Low. While ICMP is typically used in attacks where the IP source is spoofed, those packets usually have a payload. This is more likely an nmap or hping packet so the attacker will wish to see the response or lack thereof. Of course ping can generate a packet with no payload or data as well. What makes spoofing more unlikely is the other packets from this source address that arrived in this time frame.

4. Description of attack: First we need to answer why snort saw 3 packets and Firewall-1 only two and at differing times. When this detect occurred, our firewall that logged these drops was having time server (xntpd) problems.

This IP address resolves to ***209-193-18-199-cdsl-rb1.anc.acsalaska.net*** so an address given out by Dynamic Host Configuration Protocol (DHCP) to a Digital Subscriber Line (DSL) customer of acsalaska.net. The a.b.c.26 address is our firewall interface to the internet. Shortly after these pings we saw this in our snort portscan log

```
Jun 25 22:15:59 209.193.18.199:1423 -> a.b.c.26:1032 SYN ******S*
Jun 25 22:15:56 209.193.18.199:1414 -> a.b.c.26:3006 SYN ******S*
Jun 25 22:15:56 209.193.18.199:1415 -> a.b.c.26:2011 SYN ******S*
Jun 25 22:15:56 209.193.18.199:1416 -> a.b.c.26:1511 SYN ******S*
Jun 25 22:15:56 209.193.18.199:1417 -> a.b.c.26:3421 SYN ******S*
Jun 25 22:15:56 209.193.18.199:1418 -> a.b.c.26:578 SYN ******S*
Jun 25 22:15:56 209.193.18.199:1419 -> a.b.c.26:1385 SYN ******S*
Jun 25 22:15:56 209.193.18.199:1420 -> a.b.c.26:832 SYN ******S*
Jun 25 22:15:56 209.193.18.199:1421 -> a.b.c.26:722 SYN ******S*
Jun 25 22:15:56 209.193.18.199:1422 -> a.b.c.26:5716 SYN ******S*
Jun 25 22:16:02 209.193.18.199:1424 -> a.b.c.26:3006 SYN ******S*
Jun 25 22:16:02 209.193.18.199:1425 -> a.b.c.26:2011 SYN ******S*
Jun 25 22:16:02 209.193.18.199:1426 -> a.b.c.26:1511 SYN ******S*
Jun 25 22:16:02 209.193.18.199:1427 -> a.b.c.26:3421 SYN ******S*
Jun 25 22:16:02 209.193.18.199:1428 -> a.b.c.26:578 SYN ******S*
Jun 25 22:16:05 209.193.18.199:1433 -> a.b.c.26:1032 SYN ******S*
Jun 25 22:16:02 209.193.18.199:1429 -> a.b.c.26:1385 SYN ******S*
Jun 25 22:16:02 209.193.18.199:1430 -> a.b.c.26:832 SYN ******S*
Jun 25 22:16:02 209.193.18.199:1431 -> a.b.c.26:722 SYN ******S*
Jun 25 22:16:02 209.193.18.199:1432 -> a.b.c.26:5716 SYN ******S*
Jun 25 22:16:08 209.193.18.199:1434 -> a.b.c.26:3006 SYN ******S*
Jun 25 22:16:08 209.193.18.199:1435 -> a.b.c.26:2011 SYN ******S*
Jun 25 22:16:08 209.193.18.199:1436 -> a.b.c.26:1511 SYN ******S*
 <snip>
Jun 25 22:19:28 209.193.18.199:1765 -> a.b.c.26:13 SYN ******S*
Jun 25 22:19:28 209.193.18.199:1766 -> a.b.c.26:82 SYN ******S*
Jun 25 22:19:28 209.193.18.199:1767 -> a.b.c.26:533 SYN ******S*
Jun 25 22:19:28 209.193.18.199:1768 -> a.b.c.26:278 SYN ******S*
Jun 25 22:19:29 209.193.18.199:1769 -> a.b.c.26:480 SYN ******S*
Jun 25 22:19:31 209.193.18.199:1755 -> a.b.c.26:1497 SYN ******S*
Jun 25 22:19:31 209.193.18.199:1756 -> a.b.c.26:129 SYN ******S*
Jun 25 22:19:31 209.193.18.199:1757 -> a.b.c.26:835 SYN ******S*
Jun 25 22:19:31 209.193.18.199:1758 -> a.b.c.26:577 SYN ******S*
Jun 25 22:19:31 209.193.18.199:1759 -> a.b.c.26:354 SYN ******S*
Jun 25 22:19:31 209.193.18.199:1760 -> a.b.c.26:87 SYN ******S*
Jun 25 22:19:31 209.193.18.199:1761 -> a.b.c.26:828 SYN ******S*
Jun 25 22:19:31 209.193.18.199:1762 -> a.b.c.26:840 SYN ******S*
Jun 25 22:19:31 209.193.18.199:1763 -> a.b.c.26:1405 SYN ******S*
Jun 25 22:19:31 209.193.18.199:1764 -> a.b.c.26:1474 SYN ******S*
Jun 25 22:19:31 209.193.18.199:1765 -> a.b.c.26:13 SYN ******S*
Jun 25 22:19:31 209.193.18.199:1766 -> a.b.c.26:82 SYN ******S*
Jun 25 22:19:32 209.193.18.199:1767 -> a.b.c.26:533 SYN ******S*
Jun 25 22:19:32 209.193.18.199:1768 -> a.b.c.26:278 SYN ******S*
Jun 25 22:19:32 209.193.18.199:1769 -> a.b.c.26:480 SYN ******S*
Jun 25 22:19:32 209.193.18.199:1770 -> a.b.c.26:57 SYN ******S*
```

a classic nmap SYN scan. The format of snort V1.8 portscan log

```
date   time     source IP  direction target IP TCP flag name TCP flag bits
```

So what happened? Have you ever run nmap? With out the –P0 on the command line the program will attempt to ping from within nmap to the target IP address first. When this fails, as the firewall logs show it did, the nmap program notifies the user that the target IP may be filtered and offers up the suggestion as to the command line option to determine if this is true.

```
Note: Host seems down. If it is really up, but blocking our ping probes, try -P0
```

With the time differences in the icmp packets and the tcp SYN packets it appears this is what probably happened. The firewall logs show these tcp SYN packets were dropped as well. The time zone is ours so the attacker was on and scanning after TV prime time.

5. Attack mechanism: As the alert states, nmap version prior to 2.36 beta probably generated this scan. Using argus, the firewall logs, and syslog on the targeted firewall show no further activity from that source IP address in a timeframe of a week on either side of the time of the detect. Not much to the mechanism - get the code from the Internet on a very intuitive web page, build, target an IP address and await the result. The attacker most probably got the hint on how to more correctly probe the target IP address as well.

6. Correlations: CVE CAN-1999-0454 mentions nmap, but in an Operating System (OS) fingerprint mode. The snort alert is also reported to be triggered by napster which was viable during the time of the detect, though napster is more pervasive in the attempts of ICMP and is not followed closely by a TCP SYN scan from the same IP source address. Probably every IP address on the planet has received a nmap scan.

7. Evidence of active targeting: This is our inbound connection to the Internet, thus not the NAT address carried by our outbound packets. It was a single IP target address so they were targeting a specific IP target address. As they apparently went away and did not go after target IP addresses on the same subnet, or those we would have seen, they apparently had a list of IP addresses to target or the algorithm that generates target IP addresses is not localized.

8. Severity 3
(Critical + Lethal) – (System + Network Countermeasures)
(5 + 5) – (4 + 3)
Critical 5 - Our firewall
Lethal 5 - nmap a very powerful scan utility with user friendliness
System 3 - not all ports can be blocked
Countermeasures 4 – snort and firewall logs to capture the attempts, argus to see what other traffic was exchanged with the source IP address. tcpwrappers on the Solaris system hosting the firewall.

9. Defensive recommendations: Continued monitoring of snort alerts and firewall logs. Send details of incidents to incidents.org or other CERT. Start contacting ISP asap after such incidents.

10. Multiple choice test question:
Can a Microsoft Windows 2000 Professional PC trigger this snort alert? If so, how so?
a) yes
b) no
Answer: yes ping –l 0 <target IP> sends a code 8 icmp packet with 0 sized datagram.

Detect 3

snort alert file WAN link
```
[**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**]
09/12/01-18:53:21.226962 AA:0:4:0:1E:18 -> 8:0:20:D0:BB:23 type:0x800 len:0x46
10.5.20.254 -> 169.254.109.250 ICMP TTL:252 TOS:0x0 ID:54351 IpLen:20 DgmLen:56
Type:3  Code:13  DESTINATION UNREACHABLE: PACKET FILTERED
** ORIGINAL DATAGRAM DUMP:
169.254.109.250:137 -> 169.254.255.255:137 UDP TTL:126 TOS:0x0 ID:5 IpLen:20 Dgm Len:96
Len: 76
** END OF DUMP
```
snort alert file Internet link
```
[**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**]
09/12/01-18:53:21.226962 AA:0:4:0:1E:18 -> 8:0:20:D0:BB:23 type:0x800 len:0x46
10.5.20.254 -> 169.254.109.250 ICMP TTL:252 TOS:0x0 ID:54351 IpLen:20 DgmLen:56
Type:3  Code:13  DESTINATION UNREACHABLE: PACKET FILTERED
** ORIGINAL DATAGRAM DUMP:
169.254.109.250:137 -> 169.254.255.255:137 UDP TTL:126 TOS:0x0 ID:5 IpLen:20 Dgm Len:96
Len: 76
** END OF DUMP
```

The rule for this alert from the icmp.rules file

```
alert icmp any any -> !$HOME_NET any (msg:"ICMP Destination Unreachable
(Communication Administratively Prohibited)"; itype: 3; icode: 13;)
```

```
Send to the alert file packets if ICMP protocol from any IP address to any IP
address not in the local LAN IP space with ICMP type code 3 and subcode of 13
```

No entry in Firewall-1 logs

1. Source of Trace: Our network

2. Detect was generated by: snort intrusion detection

3. Probability the source address was spoofed: Low. This was a ICMP reject of a UDP packet to the broadcast address for the source network on netbios Name Service port. As such the source node will need to see the return traffic or the ICMP error code.

4. Description of the attack: As the date shows this was shortly after the terrorist activities in the United States. Being on heightened alert and seeing this unusual traffic on both the WAN (Wide Area Network) snort sensor and the Internet snort sensor gave cause to investigate further. The 169.254.109.250 IP address shows up as reserved in reverse lookup. The address the ICMP error is going to is also reserved for private address space. An ICMP error packet going to a private IP address indicating an administratively blocked NetBios Name Service packet to a broadcast address from a reserved IP address hitting our WAN and internet connection at the same time – looks like an incident. Further investigation shows the 169.254.0.0/24 IP address range is reserved for DHCP.
Thus most systems are configured to use an IP address in this range when configured to use DHCP and no DHCP (Dynamic Host Configuration Protocol) server responds with an IP address and other IP stack configuration parameters. As part of the Microsoft windows systems startup they request name service via port 137 on UDP protocol on the broadcast address for the network. This still leaves the question of why these packets are hitting our two snort sensors and why the ICMP error packets are going to the 10.5.20.254 IP address. Traceroutes to this 10. IP address did not reach the hosts. As the packets hit the sensors for about 20 minutes then stopped a traceroute to the 169.254.109.250 address also did not reach a source. After getting our telecom department involved we found the 10.5.20.254 address is a router interface on a new network segment at a remote facility on our WAN. So what was happening is a PC on the other side of the new network segment facing the 10.5.20.254 router interface was configured to have a DHCP reserved IP address. It then tried to establish connection via NetBios Name Services using the broadcast IP address. A router configured to block these send the ICMP error message to the last router to route the packets. As the route tables had not been configured to know of this new network segment the packets went out the firewall (no rules triggered for this outbound traffic) to the Internet where the Internet snort sensor picked them up.

5. Attack mechanism: Traffic to and from port 137 is normal and allowed on our WAN. Traffic to and from port 137 is blocked on the Internet firewall. Thus traffic using this port being seen as rejected to an unknown (at the time) private IP address from a reserved IP address to a reserved IP network address range broadcast address without triggering a firewall rule after the activities of September 11, 2001 was something to investigate. Once the knowledge of the location and use of the 10.5.20.254 IP address was gained the timing on the packets on the two snort sensors correlating with the varying datagram lengths yielded the plausible explanation for the traffic.

6. Correlations:
http://www.sans.org/y2k/072500-1200.htm
RAS Server Behavior When Configured to Use DHCP to Assign IP Addresses (Q216805)

7. Evidence of active targeting: None. This incident was the combination of misconfiguration of a PC and the lack of complete documentation of a router and network addition.

8. Severity: -4
(Critical + Lethal) – (System + Net Countermeasures)
Critical 0 – No server – a broadcast address not configured on our network
Lethal 0 - No explicit attack
System 1 – System configuration for DHCP
Network 3 – snort sensors
(0 + 0) – (1 + 3)

9. Defensive recommendations: Continued monitoring of snort alerts and firewall logs. Stress importance to telecommunications department of network changes being communicated to all support personnel.

10. Multiple choice test question:
What is the reserved IP address range for DHCP configurations?
a) none
b) 169.254.0.0/16
c) 10.0.0.0/8
d) 0.0.0.0/0
Answer: b

Detect 4

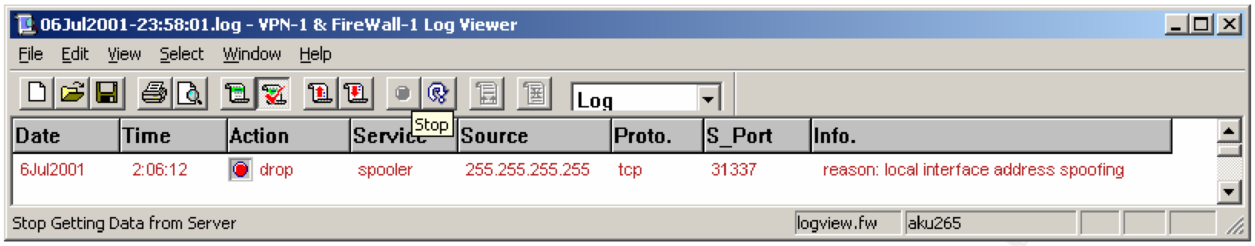
snort alert file

```
[**] BACKDOOR Q access [**]
07/06/01-02:06:11.961037 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
255.255.255.255:31337 -> a.b.c.19:515 TCP TTL:13 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
```

snort binary tcpdump

```
snort -dev -r<snort_binary_logfile>

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

07/06/01-02:06:11.961037 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x3C
255.255.255.255:31337 -> a.b.c.19:515 TCP TTL:13 TOS:0x0 ID:0 IpLen:20 Dgm Len:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
63 6B 6F                                          cko

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

Firewall-1 Log


06Jul2001-23:58:01.log - VPN-1 & FireWall-1 Log Viewer
File Edit View Select Window Help
Log
Stop
Stop Getting Data from Server
logview.fw
aku265

Date	Time	Action	Service	Source	Proto.	S_Port	Info.
6Jul2001	2:06:12	drop	spooler	255.255.255.255	tcp	31337	reason: local interface address spoofing

1. Source of trace: Our network

2. Detect was generated by: snort Intrusion Detection and Firewall-1

3. Probability the source address was spoofed: Almost certain. The source address of 255.255.255.255 is all ones in binary or the broadcast address in most IP implementations.

4. Description of attack: No correlation to other packets from the same source that we can tell as the source MAC address of the packet will be the last router and the IP address of 255.255.255.255 is obviously bogus. Attacks on port 515 of the TCP protocol are numerous but they all require a data payload. With a TcpLen of 20 this is not the case. Perhaps the packet crafter was hoping for a return ICMP to the source address which could be interpreted as an all network broadcast address. Other fields of the packet also indicate crafting: ID, Seq, Ack, Win all being 0. The TCP flags of Ack and Reset together are not standard if not an illegal combination. The various illegal or invalid values of the cited portions of the TCP packet may be set to illicit a response to the broadcast source address from a router, filtering device, or the targeted host. Another interesting aspect of this packet is the source port of 31337 (eleet) which is used by Back Orifice, though BO uses a UDP target port of 31337.

5. Attack mechanism: This packet was probably crafted with hping or similar tool.

6. Correlations: I could find only one reference to a source address of 255.255.255.255 in an incident, but it did have both Ack and Reset. A lot of mentions of destination address of 255.255.255.255 were found.

7. Evidence of active targeting: Difficult to say for sure. Only one of our Internet exposed IP addresses got this packet, but it was a highly used IP address, being that of our extranet. Thus that extranet IP address is the source of a lot of packets. It could be the IP address was gathered in some manner and a one off attempt was made to see the affect of the crafted packet. It could just as well be an IP picked at random or other selection of target IP address mechanism. As not many correlations could be found I would guess this was active targeting.

8. Severity: -1
(Critical + Lethal) – (System + Net Countermeasures)
(4 + 3) – (4 + 4)
Critical 4 – Extranet Server
Lethal 3 - No problem caused on our network, but such a severe crafted packet could be a

problem on some systems
System 4 - NT System
Countermeasures 4 – Caught by firewall, but routed by our routers

9. Defensive recommendations: Continued monitoring of snort alerts and firewall logs. Consider router acls and filters like ingres filtering for the broadcast address.

10. Multiple choice test question:
In this packet, now many fields were probably crafted?

a) 3
b) 5
c) 7
d) 6

For extra credit name them.
Answer: d
Source IP address, Source port, Sequence number, ACK number, Window size, TCP flags

Detect 5

snort alert file
```
[**] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 3]
09/04/01-06:00:36.509601 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x48
203.146.184.8:1440 -> a.b.c.23:53 UDP TTL:43 TOS:0x0 ID:64671 IpLen:20 Dgm Len:58
Len: 38
[Xref => http://www.whitehats.com/info/IDS278]
```

Snort binary tcpdump
```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
09/04/01-06:00:36.509601 0:E0:1E:83:D5:48 -> 8:0:20:C7:93:2C type:0x800 len:0x48
203.146.184.8:1440 -> a.b.c.23:53 UDP TTL:43 TOS:0x0 ID:64671 IpLen:20 Dgm Len:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4...........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```
Firewall-1 Logs
No records matching packet criteria

argus logs
```
04 Sep 01 06:00:36    udp   203.146.184.8.1440          ->    a.b.c.23.domain
       1       0        72           0           INT
04 Sep 01 06:00:36   icmp  a.b.c.23                 ->     203.146.184.8
          1       0        70           0          URP
lines wrapped for readability
```

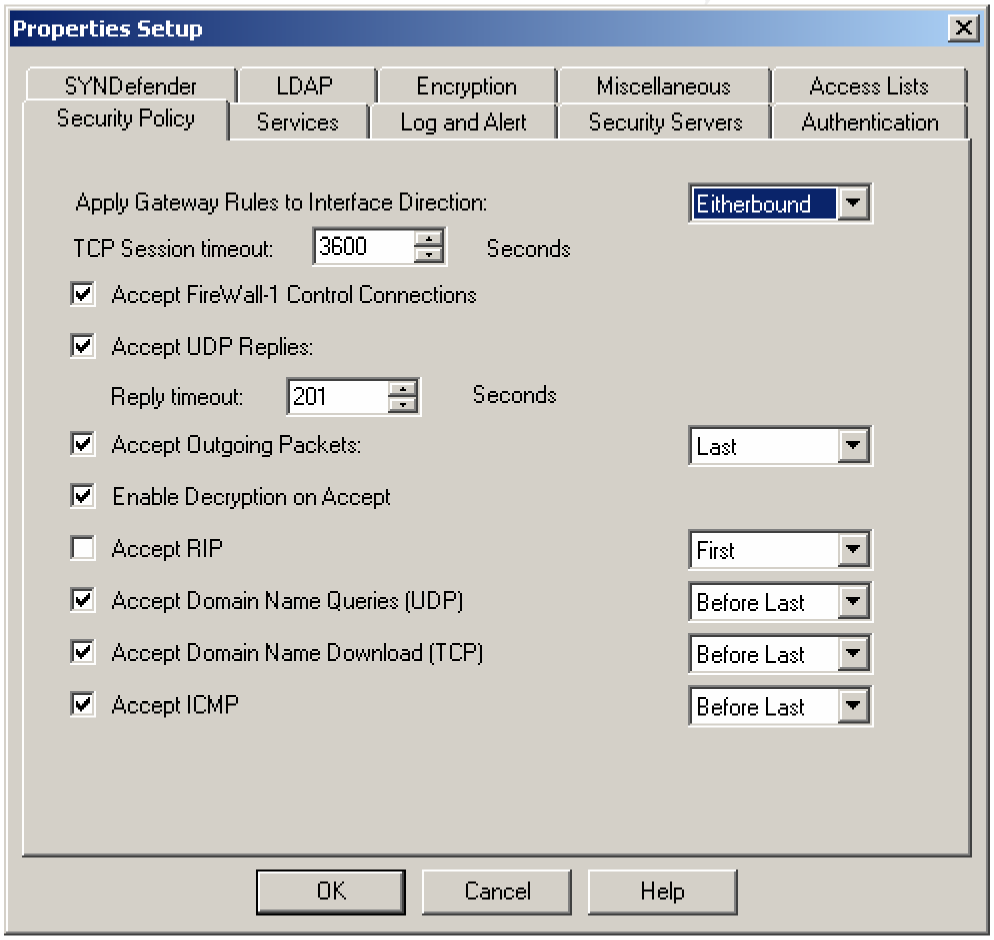
The ra command takes matching records from the argus binary log file. In the records above we have:
<date> <time> <protocol> <source IP.port> <direction> <target IP.port>
<packet counts> <byte counts> <status>

1. Source of trace: Our network

2. Detect was generated by: snort intrusion detection and argus

3. Probability the source address was spoofed: Low. This is a reconnaissance probe looking for the version of BIND running with very probable attempt at an exploit if the version returned is the values the attacker is looking for. Another possibility is information gathering for Internet statistics or similar. In either case, the response to the query is very probably desired at the source address.

4. Description of the attack: Vulnerabilities of BIND abound. The older the version of BIND, the greater the probability of vulnerability. Thus once the request for the version level of BIND has been seen, it would be good security practice to look for follow on traffic to BIND particularly from the source address or the source address (sub)network. That is why we ran a follow on search for traffic to the source IP address. The result for the previous day and subsequent week shows only one other packet – that an immediate ICMP unreachable port response, thus the attacker or information requestor can surmise the BIND service is not available on this IP address.
What is troubling for this probe is the lack of entries in the Firewall-1 logs. We block port 53 attempts to all but our BIND/DNS servers. Actually a pass for protocol 53 to the BIND/DNS servers with a deny all policy. The reason for this lack of entries in the logs is the Firewall-1 Version 4.0 rule 0 problem.
From the Security Policy GUI window Policy pulldown looking at Properties...



Apply Gateway Rules to Interface Direction: Eitherbound
TCP Session timeout: 3600 Seconds
[✓] Accept FireWall-1 Control Connections
[✓] Accept UDP Replies:
Reply timeout: 201 Seconds
[✓] Accept Outgoing Packets: Last
[✓] Enable Decryption on Accept
[ ] Accept RIP First
[✓] Accept Domain Name Queries (UDP) Before Last
[✓] Accept Domain Name Download (TCP) Before Last
[✓] Accept ICMP Before Last

Note the check box for Accept Domain Name Queries (UDP). Thus this and similar packets will be passed and not logged.

5. Attack mechanism: There are several ways to ask for the version of BIND. Both nslookup and dig can ask for this information.

nslookup
```
> set type=txt
> set class=chaos
> version.bind
```

dig
```
dig @ <server> txt chaos version.bind
```
Using O'Reilly's "DNS and BIND" by Paul Albitz and Cricket Lui and RFC 1035

The header contains the following fields:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ID															
QR	Opcode				AA	TC	RD	RA	Z			RCODE			
QDCOUNT															
ANCOUNT															
NSCOUNT															
ARCOUNT															

where:

ID

A 16 bit identifier assigned by the program that generates any kind of query. This identifier is copied the corresponding reply and can be used by the requester to match up replies to outstanding queries.

which indicates the ID for this query is 1234, so a program was probably written to send these version number queries. I could find no mention of methods of sending BIND version queries with an ID set to 1234.
The source IP shows to be from APNIC belonging to Roiet Pacnitchayakan Technology School in Thailand. If I did my math correctly, that puts the query sent about 9pm local Thailand time Sunday evening.

6. Correlations: None

7. Evidence of active targeting: The target IP address in this case is for inbound ssh traffic only. No outbound traffic is supposed to go out and argus shows none. Also no other Internet exposed IP addresses were hit. Thus this appears to be a random IP address selection.

8. Severity: 0
(Critical + Lethal) – (System + Net Countermeasures)
(3 + 2) – (4 + 1)
Critical 3 – Target IP address is in our extranet
Lethal 2 - Query was for version of BIND
System 4 – BIND not running on the target system
Countermeasures 1 – Firewall-1 not logging BIND udp packets

9. Defensive recommendations: Continued monitoring of snort alerts, firewall logs, and argus traffic patterns. Change Firewall-1 rule 0 parameters to log and block BIND udp packets.

10. Multiple choice test question:
What in the version query packet indicates the packet is not from another BIND/DNS server?

a) Server to server communications use TCP
b) Server to server communications use port 53 to port 53
c) Server to server communications use a different opcode

Answer: b

Assignment 3 – "Analyze This" Scenario

The assignment calls for 5 recent consecutive days. On the day I went to pickup the data files the most recent days were 09/29/2001 -> 01/03/2001. If this were a real assignment the most recent days of the time period contracted for would be what I would pick. I may loose information that might indicate a compromise on a system in a day's log before the most recent days, but intrusion detection is more concerned with now.
After retrieval and uncompression the files had all occurrences of MY.NET replaced with 0.0. This would make the standard 4 octets for the IP address and be an illegal address so probably not in the data otherwise. The 5 alert files were merged into one, then that file run through snortsnarf.pl version v010821.1 from Silicon Defense. That produced this table:

324223 alerts found using input module SnortFileInput, with sources:
Earliest alert at **00:00:03**.643183 *on 09/29/2001*
Latest alert at **23:50:34**.472240 *on 10/03/2001*

Signature	# Alerts	# Sources	# Destinations
WEB-CGI w3-msql access	1	1	1
INFO - Web File Copied ok	1	1	1
Virus - SnowWhite Trojan Incoming	1	1	1
MISC solaris 2.5 backdoor attempt	1	1	1
WEB-MISC webdist.cgi access	1	1	1
MISC PCAnywhere Startup	1	1	1
WEB-IIS anot.htr access	1	1	1

WEB-IIS iisadmpwd attempt	1	1	1
INFO - Web Cmd completed	1	1	1
IDS50/trojan_trojan-active-subseven	1	1	1
INFO - Web Dir listing	1	1	1
Virus - Naked Wife	1	1	1
WEB-CGI survey.cgi access	1	1	1
TCP SMTP Source Port traffic	1	1	1
WEB-MISC ~root	1	1	1
WEB-MISC /etc/passwd	1	1	1
WEB-IIS scripts-browse	1	1	1
IDS475/web-iis_web-webdav-propfind	1	1	1
ICMP Alternate Host Address (Undefined Code!)	1	1	1
WEB-CGI tsch access	1	1	1
WEB-CGI cvsweb.cgi access	2	1	1
WEB-MISC whisker splice attack	2	1	1
X11 xopen	2	1	1
Virus - Possible MyRomeo Worm	2	2	2
DDOS mstream handler to client	2	1	1
TELNET access	2	1	1
SYN-FIN scan!	2	2	2
Virus - Possible scr Worm	2	2	2
MISC Source Port 20 to <1024	2	1	1
WEB-CGI rsh access	3	3	1
WEB-IIS File permission canonicalization(Chinese charset)	3	3	3
RFB - Possible WinVNC - 010708-1	3	3	3
spp_http_decode: CGI Null Byte attack detected	3	2	2

SCAN XMAS	3	2	2
WEB-IIS view source via translate header	3	3	3
Attempted Sun RPC high port access	3	3	3
SNMP public access	3	2	1
WEB-IIS asp-dot attempt	3	1	1
INFO Outbound GNUTella Connect request	3	3	3
WEB-MISC Lotus Domino directory traversal	4	4	3
WEB-CGI ksh access	4	1	1
WEB-MISC compaq nsight directory traversal	4	3	3
SMTP chameleon overflow	4	4	4
INFO napster new user login	4	1	3
WEB-FRONTPAGE fourdots request	4	1	1
ICMP SRC and DST outside network	6	5	5
Tiny Fragments - Possible Hostile Activity	6	3	3
INFO Inbound GNUTella Connect request	6	6	4
SCAN FIN	7	5	5
X11 outgoing	7	6	7
Back Orifice	7	2	7
WEB-CGI csh access	7	6	3
EXPLOIT x86 NOPS	7	4	4
RPC tcp traffic contains bin_sh	8	6	6
WinGate 1080 Attempt	8	7	7
Port 55850 udp - Possible myserver activity - ref. 010313-1	8	2	4
Virus - Possible pif Worm	9	4	6
ICMP redirect (Host)	10	2	3
INFO - Possible Squid Scan	10	7	10

EXPLOIT x86 stealth noop	12	7	5
Russia Dynamo - SANS Flash 28-jul-00	12	9	6
BACKDOOR NetMetro File List	13	2	2
ICMP Destination Unreachable (Fragmentation Needed and DF bit was set)	14	14	2
connect to 515 from inside	14	4	4
x86 NOOP - unicode BUFFER OVERFLOW ATTACK	16	9	5
WEB-CGI redirect access	16	16	6
WEB-MISC L3retriever HTTP Probe	16	2	3
NMAP TCP ping!	18	12	11
WEB-IIS File permission canonicalization	18	9	17
SCAN Synscan Portscan ID 19104	19	19	16
WEB-FRONTPAGE shtml.dll	20	2	1
CS WEBSERVER - external ftp traffic	20	16	1
WEB-IIS _vti_inf access	22	18	5
WEB-FRONTPAGE fpcount.exe access	22	21	2
ICMP Echo Request Delphi-Piette Windows	23	3	23
BACKDOOR NetMetro Incoming Traffic	25	4	4
WEB-FRONTPAGE _vti_rpc access	32	26	7
beetle.ucs	33	7	7
INFO FTP anonymous FTP	34	18	20
EXPLOIT x86 setgid 0	34	34	32
WEB-MISC count.cgi access	34	29	2
connect to 515 from outside	37	1	37
ICMP Source Quench	39	18	6
ICMP Destination Unreachable (Protocol Unreachable)	39	24	25
MISC Large ICMP Packet	39	33	10

WEB-CGI scriptalias access	40	4	4
TFTP - External TCP connection to internal tftp server	40	6	6
TFTP - Internal UDP connection to external tftp server	46	4	11
EXPLOIT x86 setuid 0	55	49	45
INFO napster upload request	56	17	8
ICMP Echo Request L3retriever Ping	58	8	9
ICMP Echo Request BSDtype	60	17	28
WEB-MISC http directory traversal	67	48	5
High port 65535 udp - possible Red Worm - traffic	68	37	33
TELNET login incorrect	84	7	84
ICMP Echo Request Sun Solaris	98	17	65
Queso fingerprint	105	31	31
ICMP Echo Request CyberKit 2.2 Windows	135	44	15
ICMP Echo Request Windows	138	81	40
External RPC call	140	8	135
Port 55850 tcp - Possible myserver activity - ref. 010313-1	157	36	38
Watchlist 000222 NET-NCFC	180	27	71
EXPLOIT x86 NOOP	182	41	32
INFO Possible IRC Access	314	86	60
WEB-MISC 403 Forbidden	350	11	285
INFO Outbound GNUTella Connect accept	379	367	99
TCP SRC and DST outside network	399	126	289
SUNRPC highport access!	432	5	5
FTP DoS ftpd globbing	602	70	71
High port 65535 tcp - possible Red Worm - traffic	609	33	30
ICMP Destination Unreachable (Host Unreachable)	622	146	52

SCAN Proxy attempt	651	67	455
INFO napster login	692	40	102
WEB-IIS Unauthorized IP Access Attempt	722	7	60
ICMP Destination Unreachable (Network Unreachable)	837	3	255
ICMP traceroute	887	574	481
ICMP Fragment Reassembly Time Exceeded	927	84	98
INFO Napster Client Data	1202	173	468
Possible trojan server activity	1419	78	1208
TFTP - Internal TCP connection to external tftp server	1730	61	61
INFO Inbound GNUTella Connect accept	1755	140	1456
Watchlist 000220 IL-ISDNNET-990517	1799	307	176
CS WEBSERVER - external web traffic	2795	1592	1
ICMP Echo Request speedera	2989	1	1
SMTP relaying denied	3642	3	39
MISC source port 53 to <1024	3879	2003	11
MISC traceroute	4563	128	11
SMB Name Wildcard	4852	81	4065
UDP SRC and DST outside network	5504	23	34
ICMP Destination Unreachable (Communication Administratively Prohibited)	5782	31	312
ICMP Echo Request Nmap or HPING2	6128	77	189
WEB-MISC prefix-get //	6474	1312	5
INFO MSN IM Chat data	6825	634	497
Incomplete Packet Fragments Discarded	14601	18	15
MISC Large UDP Packet	25586	91	59
spp_http_decode: IIS Unicode attack detected	30472	5499	9543
IDS552/web-iis_IIS ISAPI Overflow ida nosize	38301	9822	21129

Null scan!	40144	104	72
WEB-MISC Attempt to execute cmd	102830	19460	30032

A quick calculation of the alerts per 24-hour day gives 64,885, or 2,701 per hour on average. As this data period covers the Code Red, Code Blue, Code Green, Code Rainbow, and nimda worm flurry this was expected.

First we will address the alerts that are usually benign, starting with the bottom of the table to get the greater numbers of alerts.

Null Scan!
The top talkers and top listeners shows two nodes produce most of these alerts
Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
3.1.3.7	39562	39562	1	1
65.66.70.18	150	153	1	1
64.160.24.3	104	105	1	1
216.62.157.134	29	30	1	1
162.83.140.139	24	25	1	1
63.204.104.205	21	22	1	1
63.204.251.128	19	19	1	1
141.133.192.101	15	15	2	2
207.46.203.12	12	12	1	1
65.64.75.201	9	9	1	1

Destinations receiving this attack signature

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
MY.NET.217.54	39562	39566	1	4
MY.NET.226.110	150	163	1	6
MY.NET.204.70	104	107	1	2
MY.NET.204.110	29	33	1	4

MY.NET.53.36	24	36	1	9
MY.NET.219.14	21	34	1	10
MY.NET.190.20	19	20	1	2
MY.NET.20.10	15	16	4	5
MY.NET.233.58	12	21	3	9
MY.NET.217.114	12	21	6	12

Thus most of the null scan traffic is between 3.1.3.7 and MY.NET.217.54. The class A 3.0.0.0/128 belongs to General Electric

```
General Electric Company (NET-GE-INTERNET)
  1 Independence Way
  Princeton, NJ 08540
  US
  Netname: GE-INTERNET
  Netblock: 3.0.0.0 - 3.255.255.255
  Coordinator:
    General Electric Company  (GET2-ORG-ARIN)  GENICTech@GE.COM
    518-612-6672
  Record last updated on 12-Nov-1998.
  Database last updated on 5-Oct-2001 23:18:41 EDT.
```

Null scans for reconnaissance are either horizontal (looking at many hosts) or vertical (looking at many ports on a specific host). In the case of these two nodes the source port is the same value and the destination ports are in a narrow range.

Correlation:

Global Incident Analysis Center: Detects Analyzed 9/20/00

Defensive recommendations:

At our site we see most null scans, i.e. no TCP flag bits set, are the initiation of VPN sessions. If this is the case here, then a pass rule can be used to not log/alert on this traffic on these two particular IP addresses. If not, then the full packet will need to be analyzed to determine the problem.

MISC. Large UDP Packet

This is hard to dismiss or include as the packet size that triggers the rule/alert is not given. It is interesting to note that 78% of these alerts also have source port of 0. Searching for large UDP packets with port of 0 yields some NFS implementations that use port 0, some SAMBA configurations that use port 0, and the use of port 0 in Operating System (OS) fingerprinting.

Analysis of the packet would be the next step. We could also increase the dss field in the rule to gain further understanding on this alert, as this would allow the other rules to trigger instead of the large UDP size. The top 5 talkers and listeners for this alert:

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
209.190.237.123	13224	13384	1	2
61.153.17.244	3330	3331	4	4
61.134.9.121	3051	3058	1	2
61.150.5.19	1466	1525	6	6
213.244.175.42	1206	1207	2	2

Destinations receiving this attack signature

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
MY.NET.70.134	13224	13235	1	4
MY.NET.111.142	3075	3143	2	7
MY.NET.153.185	1552	1557	2	7
MY.NET.112.244	1080	1082	2	4
MY.NET.153.199	843	849	1	4

which does indicate most of the traffic is between the top talker and top listener.

<u>Incomplete Packet Fragments Discarded</u>

Again we have almost all of the traffic generating this alert from two nodes, the top talker and the top listener.

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
207.199.100.7	13640	13640	1	1
218.2.4.102	331	382	1	2
MY.NET.217.66	319	334	1	12
211.110.11.194	78	223	1	1
61.150.5.19	59	1525	1	6

Destinations receiving this attack signature

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
MY.NET.60.11	13640	13779	1	9
MY.NET.144.51	331	1092	1	4
64.15.228.213	319	319	1	1
MY.NET.211.54	78	240	1	4
MY.NET.111.142	59	3143	1	7

The top talker belongs to Verio, Inc., a large Web hosting company. These alerts could indicate a problem in the network infrastructure. Again, more data is needed to resolve this alert type.

ICMP Destination Unreachable (Communication Administratively Prohibited)
These are normally a good thing. The top listeners are all on MY.NET. This can be caused by something trying all the MY.NET address space and the top destination not being assigned to a machine, the machine down when the traffic is seen (users turn the machine off to go home), or there is a packet filtering device on the network between the source and destination node.

MISC traceroute
This is usually considered a benign activity. Either the packet was intended to be a traceroute or the packet just happened to be the packet's original Time to Live (ttl) hops away.

MISC source port 53 to <1024
This is probably benign, but a list of BIND/DNS servers would be needed to determine. Most of the entries are port 53 to port 53 communications which is normal for BIND server to BIND server.

A few of the other alerts that are usually non problematic:
ICMP traceroute
ICMP Destination Unreachable (Network Unreachable)
ICMP Destination Unreachable (Host Unreachable)
WEB-MISC 403 Forbidden

Now to address the top alerts that are probably NOT to be dismissed.

WEB-MISC Attempt to execute cmd
This probably Code Red II.

WEB-MISC Attempt to execute cmd	19460 sources	30032 destinations

These numbers also indicate Code Red, Code Red II, one of the attack mechanisms of nimda, or similar. None of the source IPs is from MY.NET which would indicate MY.NET is not infected or the rule excludes HOME_NET as a source. As these have been covered well in the media and most security sites, we will refer the customer to those resources and move to other alerts.

Correlation:
[CERT Incident Note IN-2001-09: "**Code Red II**:" Another Worm **...**]
[incidents.org - By The SANS Institute: **Code Red (II)**]
[incidents.org - By The SANS Institute: **Code Red** Threat FAQ]
Defensive recommendation:
Keep patches up to date; monitor security related news groups, web sites, and mail lists; monitor system logs. Purchase, maintain, and use virus scanning tools. Purchase, maintain, and use email filtering tools.
Virus Busters

IDS552/web-iis_IIS ISAPI Overflow ida nosize
Again Code Red or similar attempt to exploit the Microsoft Internet Information Services (IIS) web server's Index Server Application Programming Interface (ISAPI) vulnerability.

IDS552/web-iis_IIS ISAPI Overflow ida nosize	9822 sources	21129 destinations

Correlation and Defensive recommendations as above (WEB-MISC Attempt to execute cmd)

spp_http_decode: IIS Unicode attack detected
This might be Code Blue or other attacks attempting to exploit the vulnerability of the double dot directory transversal when unicode character representations of / and/or \ are substituted.

spp_http_decode: IIS Unicode attack detected	5499 sources	9543 destinations

A good write up of this vulnerability is given at Security Focus.
Correlation and Defensive recommendations as above (WEB-MISC Attempt to execute cmd)

INFO MSN IM Chat data
Almost all the destination IP addresses are 64.4.0.0/128 which belongs to HotMail.
Correlation:
Instant Chat
Defensive recommendation:
Depending on site policy this is either accepted usage of the Internet connection and can be removed from the snort ruleset –or- a violation of policy and can be addressed by taking the list of source nodes from MY.NET and finding the users on those machines using the Microsoft HotMail service.

ICMP Echo Request Nmap or HPING2
The signature definition for this alert is an ICMP echo request with a datagram size of 0 bytes. The top talkers shows most of these from one IP on MY.NET and most others from other nodes on MY.NET.
Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
MY.NET.218.174	4204	4210	3	9
MY.NET.219.38	673	705	3	34
MY.NET.204.150	628	632	4	6

MY.NET.203.178	216	217	4	5
MY.NET.97.46	51	51	31	31

The native ping or source of ICMP echo requests may generate packets with datagram length of 0 bytes. To investigate further we would need to know the function of the top talker IP address.
Correlation:
Neohapsis Archives - Snort discussion - Re: [Snort-users] **...**
Defensive recommendation:
Though most implementations of ping can send 0 length datagrams via ICMP with command line switches or other means, certain versions of nmap do so as well. Then there are various tools which craft ICMP and other protocol packets. It would be good in investigate all instances of this alert. Also check other traffic from the alerted source IP address and IP addresses on the source IP addresses's network.

UDP SRC and DST outside network
This alert indicates the packet's source and destination IP addresses are not part of the snort environmental variable $HOME_NET.
The top 10 talkers and top 10 listeners:
Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
63.250.213.92	4595	4595	1	1
3.0.0.99	746	746	1	1
172.16.3.213	48	48	1	1
164.107.98.247	38	38	2	2
198.180.47.169	11	11	1	1
192.168.0.96	10	22	2	13
134.192.133.116	6	6	1	1
169.254.101.152	6	91	5	74
169.254.125.27	6	6	3	3
192.168.1.106	6	14	1	9

Destinations receiving this attack signature

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
233.28.65.178	4595	4595	1	1
10.0.0.1	746	746	1	1

172.16.1.103	48	48	1	1
164.107.3.40	34	34	1	1
198.180.47.156	11	11	1	1
24.3.0.36	6	6	1	1
134.192.128.52	6	6	1	1
168.95.192.1	6	6	1	1
206.27.242.2	5	5	3	3
208.194.25.201	5	5	1	1

Amazing that the top talker and top listener have the same number of alerts and alerts total. Looking at the raw alert file we see that 63.250.219.92:1031 attempts to contact 233.28.65.178:5779 from 10/3/2001 08:53:05 till 10/3/2001 16:59:33.

```
10/03-08:53:05.462524  [**] UDP SRC and DST outside network [**] 63.250.213.92:1031 ->
233.28.65.178:5779
10/03-08:53:13.009933  [**] UDP SRC and DST outside network [**] 63.250.213.92:1031 ->
233.28.65.178:5779
<SNIP>
10/03-16:59:31.677076  [**] UDP SRC and DST outside network [**] 63.250.213.92:1031 ->
233.28.65.178:5779
10/03-16:59:33.701284  [**] UDP SRC and DST outside network [**] 63.250.213.92:1031 ->
233.28.65.178:5779
```

The 63.50.213.92 address belongs to yahoo!

Yahoo! Broadcast Services, Inc. (NETBLK-NETBLK2-YAHOOBS)

2914 Taylor st

Dallas, TX 75226

US

Netname: NETBLK2-YAHOOBS

Netblock: 63.250.192.0 - 63.250.223.255

The 233.28.65.178 address is IANA reserved for MCAST-NET. Thus this traffic appears to be part of some multicast groups setup from Yahoo! Broadcast Services. Thus in this instance the source and destination IP addresses are not both outside.
Correlation:
IP **Multicast** - Webopedia Definition and Links
Defensive recommendation:
Refine rules to define and handle multicast address range.
As before, if this service is part of allowed Internet usage policy, the rule can be preceded with a pass rule.

SMB Name Wildcard
Almost all these alerts were generated from MY.NET to MY.NET. As such the SMB name wildcard is normal for sites with NetBIOS NameServices on port 137.
The external sources for this alert:

Sources external to MY.NET triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
130.67.86.110	2	2	1	1
213.104.229.41	2	2	1	1
217.84.132.60	2	2	1	1
194.74.28.2	2	2	1	1
210.12.16.34	2	2	1	1
209.85.131.101	1	1	1	1
61.217.162.92	1	1	1	1
217.34.238.242	1	2	1	2
208.150.174.5	1	1	1	1
130.13.79.179	1	1	1	1
213.213.52.183	1	1	1	1
130.13.109.192	1	1	1	1
130.67.65.141	1	1	1	1
130.49.117.94	1	1	1	1
130.67.75.118	1	1	1	1
63.25.67.101	1	1	1	1
213.93.204.109	1	1	1	1
66.137.164.185	1	1	1	1
61.75.72.1	1	1	1	1
66.105.84.187	1	1	1	1
130.13.120.21	1	1	1	1
130.130.130.13	1	1	1	1
211.248.158.2	1	1	1	1
213.65.235.9	1	1	1	1
130.157.57.210	1	1	1	1

10.0.0.3	1	1	1	1
172.150.1.250	1	1	1	1
61.74.194.11	1	1	1	1
4.3.104.165	1	1	1	1
217.128.207.212	1	1	1	1
130.130.3.2	1	1	1	1
217.98.153.138	1	9	1	7
211.74.233.17	1	1	1	1
128.134.126.1	1	1	1	1
195.63.12.154	1	1	1	1

Most of these are single alerts to single nodes on MY.NET. If the MY.NET site runs Windows NetBIOS name service to external nodes, this may be normal traffic. I do note a private IP address in the 10.0.0.3 source IP address which is one of the reserved private IP address ranges. Also the 217.98.15.138 address which triggered one alert to one destination for this signature, but triggered 9 alerts against 7 destinations otherwise.
Correlation:
ID FAQ - Port 137 Scan
Defensive recommendation:
The rule that generates this alert should be modified to alert on $EXTERNAL_NET -> $HOME_NET.
If possible block NetBIOS ports 137,138, and 139 at the border routers.

Analysis to this point has been done by quantity of the alert. There are too many alerts in this dataset to do an analysis on each and still be under 70 pages for the paper length. While not as objective as quantity of alerts, the subjective focus should be done for completeness. Alerts which should be analyzed with that selection criteria:
TFTP – Internal connection to external tftp server
Possible trojan server activity
High port 65535 tcp – possible Red Worm – traffic
FTP DoS ftpd globbing
SUNRPC high port access!
External RPC call
High port 65535 udp – possible Red Worm – traffic
TFTP – External TCP connection with internal tftp server
connect to 515 from outside
SCAN Sunscan Portscan ID 19104
ICMP redirect (Host)
Back Orfice

X11 outgoing
SCAN FIN
Tiny Fragments – Possible Hostile Activity
ICMP SRC and DST outside network
SNMP public access
Attempted Sun RPC high port access
MISC Source port 20 to < 1024
SYN-FIN scan!
X11 xopen
MISC Solaris 2.5 backdoor attempt

Port Scans

The port scan log files were analyzed similar to the alert files. MY.NET was replaced with 0.0 with a text editor. Then the 5 separate scan.<date> files were concatenated into a single scan file of about 23 MegaBytes. Then snortsnarf was run against this single scan file producing this result:
368407 alerts found using input module SnortFileInput, with sources:
Earliest alert at **00:00:02** *on 9/29/2001*
Latest alert at **23:55:13** *on 10/3/2001*

Signature (click for sig info)	# Alerts	# Sources	# Destinations	Name
TCP 2*S***AU scan	1	1	1	
TCP 21SFRP*U scan	1	1	1	
TCP *1S*RP*U scan	1	1	1	
TCP **S*RPAU scan	1	1	1	
TCP 21SFRP** scan	1	1	1	
TCP *1SFR*A* scan	1	1	1	
TCP 2*S*RPAU scan	1	1	1	
TCP 2*SF**A* scan	1	1	1	
TCP *1***PA* scan	1	1	1	
TCP 2*SFR**U scan	1	1	1	

TCP **SF**AU scan	1	1	1	
TCP 21S*R*** scan	1	1	1	
TCP *1*F*P*U scan	1	1	1	
TCP **SF*P** scan	1	1	1	
TCP 21***PA* scan	1	1	1	
TCP 2**F*PAU scan	1	1	1	
TCP 21*****U scan	1	1	1	
TCP 21*FR*A* scan	1	1	1	
TCP 2*S***** scan	1	1	1	
TCP *1S***AU scan	1	1	1	
TCP 2*SF*PA* scan	1	1	1	
TCP 2*SF*P** scan	1	1	1	
TCP *1**RP*U scan	1	1	1	
TCP *1S**P** scan	1	1	1	
TCP 21***PAU scan	1	1	1	
TCP 21**RP** scan	1	1	1	
TCP 21S***AU scan	1	1	1	
TCP **S***AU scan	1	1	1	
TCP ***FR*AU scan	1	1	1	
TCP 2***R*A* scan	1	1	1	

TCP ***FRPAU scan	1	1	1	
TCP 2***R**U scan	1	1	1	
TCP *1**RPA* scan	1	1	1	
TCP 21*FRPAU scan	1	1	1	
TCP **S**P** scan	1	1	1	
TCP **SFR*AU scan	1	1	1	
TCP **S*R*** scan	1	1	1	
TCP **SFR**U scan	1	1	1	
TCP 2****PAU scan	1	1	1	
TCP 2**FR*A* scan	1	1	1	
TCP *1SFR*AU scan	1	1	1	
TCP *1***P** scan	1	1	1	
TCP ****RP*U scan	1	1	1	
TCP 2*S*R*** scan	1	1	1	
TCP 21SFRPA* scan	1	1	1	
TCP *1*F**A* scan	1	1	1	
TCP 2*SFR*** scan	1	1	1	
TCP ****R**U scan	1	1	1	
TCP 21***P*U scan	1	1	1	
TCP 2**FR**U scan	1	1	1	

TCP 2***RP*U scan	1	1	1	
TCP 2*S**P** scan	1	1	1	
TCP 21SF**A* scan	1	1	1	
TCP *1**R*AU scan	1	1	1	
TCP 2*SFRP*U scan	1	1	1	
TCP 2**FR*AU scan	1	1	1	
TCP ***F***U scan	1	1	1	
TCP *1*F*P** scan	1	1	1	
TCP *1**R**U scan	1	1	1	
TCP *1SF*P*U scan	1	1	1	nmapID
TCP 21*F*P** scan	1	1	1	
TCP *1SFR**U scan	1	1	1	
TCP 21**R**U scan	1	1	1	
TCP *1SF***U scan	1	1	1	
TCP 21SFR**U scan	1	1	1	
TCP 2*S***A* scan	1	1	1	
TCP ***FR*A* scan	1	1	1	
TCP *1S***** scan	1	1	1	
TCP *1SF*P** scan	1	1	1	
TCP 2**FR*** scan	1	1	1	

TCP *1S**PA* scan	1	1	1	
TCP *1*FR*AU scan	1	1	1	
TCP 2******* scan	1	1	1	
TCP **S*RP*U scan	1	1	1	
TCP **S**PA* scan	1	1	1	
TCP 2*S*RP** scan	1	1	1	
TCP *1****A* scan	1	1	1	
TCP 21S*RPA* scan	1	1	1	
TCP 2***RPAU scan	1	1	1	
TCP **S*RP** scan	1	1	1	
TCP 21S**PA* scan	1	1	1	
TCP 21**R*** scan	1	1	1	
TCP **S*R*AU scan	1	1	1	
TCP 2*S*R*AU scan	1	1	1	
TCP *1S****U scan	1	1	1	
TCP 21****A* scan	1	1	1	
TCP 21S*RPAU scan	1	1	1	
TCP ****RPAU scan	1	1	1	
TCP 2*SFRPA* scan	1	1	1	
TCP 21*F**A* scan	2	2	2	

TCP *1S**PAU scan	2	1	2	
TCP 2*S*RPA* scan	2	2	2	
TCP 2**F*PA* scan	2	2	2	
TCP 21**RPAU scan	2	1	2	
TCP *1S*RP** scan	2	1	2	
TCP *1SFR*** scan	2	2	2	
TCP *1*FR*** scan	2	2	2	
TCP 21SF**** scan	2	1	2	
TCP 2*SFRPAU scan	2	2	2	
TCP 2*SFR*AU scan	2	2	2	
TCP *1*FRP** scan	2	2	2	
TCP **SFRP** scan	2	2	2	
TCP 21S***A* scan	2	1	1	
TCP *1S*RPAU scan	2	2	2	
TCP ***F*P** scan	2	2	2	
TCP **SFRPA* scan	2	1	2	
TCP **SFR*A* scan	2	2	2	
TCP 2**F**AU scan	2	1	2	
TCP *1S*R*A* scan	2	2	2	
TCP 2*SF**AU scan	2	1	2	

TCP 21*F*PA* scan	2	1	2	
TCP *1****AU scan	2	2	2	
TCP ***FRPA* scan	2	2	2	
TCP 21*FR*** scan	2	1	2	
TCP 21SFRPAU scan	2	2	2	
TCP *1**RPAU scan	2	2	2	
TCP 21S*R**U scan	2	1	2	
TCP 21****AU scan	2	2	2	
TCP 2**FRP** scan	2	2	2	
TCP 21*F*PAU scan	2	2	2	
TCP 21***P** scan	2	2	2	
TCP *1S*R**U scan	2	2	2	
TCP 21SF*P** scan	2	1	2	
TCP *1*F**** scan	2	1	2	
TCP *1SFRP** scan	2	1	2	
TCP 21SFR*AU scan	2	2	2	
TCP *1S*R*** scan	2	1	2	
TCP **SFR*** scan	2	1	2	
TCP 21**R*A* scan	2	2	2	
TCP *1*FRP*U scan	3	3	3	

TCP ****RP** scan	3	1	3	
TCP ***FR**U scan	3	1	3	
TCP **S****U scan	3	1	3	
TCP 21S*R*A* scan	3	3	3	
TCP *1*F*PAU scan	3	1	3	
TCP ***FRP*U scan	3	2	3	
TCP 21****** scan	4	2	4	
TCP 2**F**** scan	4	3	4	
TCP **SFRP*U scan	4	3	4	
TCP 2****P*U scan	4	2	4	
TCP 2*SFR*A* scan	4	4	4	
TCP *1****** scan	5	2	5	
TCP ***FRP** scan	5	5	5	NoACK
TCP **S*R*A* scan	5	5	5	
TCP 21*FR*AU scan	5	2	5	
TCP ***F**** scan	5	5	5	
TCP *******U scan	5	2	5	
TCP *1**R*** scan	6	6	6	
TCP 2*****A* scan	6	2	6	
TCP 21S***** scan	73	18	18	

TCP *****P** scan	247	152	50	VECNA
TCP **S***** scan	5103	65	3820	Syn Scan
TCP ******** scan	13106	49	75	Null Scan
UDP scan	349626	212	19452	

Well. No classic SYNFIN scans, but just about every other possible combination of TCP flag settings. First thought was a VPN or other tunneling mechanism that encrypts the packet below the IP level. Encrypted TCP flags would not be legal TCP flag combinations typically. However if this were a VPN in the classic sense, then there would be a VPN server or VPN software on several of MY.NET's IP addresses. In the first case most of the illegal TCP flag combinations would be to a small number of IP addresses on MY.NET. In the second case there would be larger numbers of the illegal flag combinations. Neither is the case. Other possibilities to explain the data might be a large IP address space so the probability of getting this many rogue packets is high. TCP ECN might also be a factor in the large number of TCP flag combinations.
Starting with the UDP scans taking the top 10 sources and top 10 destinations we get these tables:

349626 alerts with this signature using input module SnortFileInput, with sources:

Earliest such alert at **00:00:02** *on 9/29/2001*
Latest such alert at **23:55:13** *on 10/3/2001*

UDP scan	212 sources	19452 destinations

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
MY.NET.160.114	105810	105810	3788	3788
205.188.233.185	26287	26287	47	47
205.188.244.121	26158	26158	45	45
MY.NET.222.158	22650	22652	788	789
205.188.246.121	21954	21954	43	43
205.188.233.121	20145	20145	46	46
205.188.233.153	17503	17503	42	42
205.188.244.57	16751	16751	43	43

MY.NET.160.169	14221	14221	1894	1894
MY.NET.208.58	10939	10943	842	843

Destinations receiving this attack signature

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
166.84.159.101	8892	8892	2	2
MY.NET.184.23	7064	7064	6	6
MY.NET.145.166	6598	6598	6	6
24.178.16.42	6583	6583	1	1
MY.NET.151.85	6572	6572	6	6
MY.NET.178.94	6337	6337	6	6
199.173.16.53	5923	5923	31	31
MY.NET.69.221	5571	5571	6	6
MY.NET.184.40	5463	5463	6	6
131.204.196.244	5400	5400	1	1

The top source is on MY.NET and attempts 3788 destinations almost all from UDP port 777 to destination port 27005. I could find no correlation for this UDP port scan combination. IANA shows port 777 to be Multiling HTTP.
Of the 10 top sources, about half are on MY.NET. The rest are all assigned to nodes at spinner.com which shows to be a Internet radio service. These services can use UDP since music can loose packets and the overhead of connection build and teardown are avoided. This service downloads a player which I bet uses the destination port of 6970 which is the port on the destination nodes in MY.NET
For scans the important nodes are the destination nodes on MY.NET. Looking at the detail on the nodes in MY.NET in the top 10 destinations we see almost all are the UDP port that spinner.com uses.

```
Oct 1 08:56:21 205.188.233.121:10816 -> MY.NET3:6970 UDP
```

Thus these scans are probably not scans, but many of MY.NET nodes using the spinner radio service.

The TCP scans
We have covered the null scans previously in the alert analysis section. Now we look at the SYN scans.

5103 alerts with this signature using input module SnortFileInput, with sources:

Earliest such alert at **00:16:30** *on 9/29/2001*
Latest such alert at **23:52:20** *on 10/3/2001*

TCP **S***** scan	65 sources	3820 destinations

The top 10 sources and top 10 destinations:
Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
MY.NET.228.6	679	679	631	631
209.150.29.131	610	610	604	604
MY.NET.208.10	551	587	489	514
209.10.173.30	547	547	544	544
MY.NET.224.186	492	501	1	10
213.52.195.39	381	382	381	382
195.249.88.3	368	368	367	367
MY.NET.102.17	279	291	263	273
MY.NET.91.19	225	225	1	1
MY.NET.60.16	166	166	1	1

Destinations receiving this attack signature

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
198.162.1.150	492	492	1	1
12.154.160.11	225	225	1	1
63.94.219.182	217	217	2	2
MY.NET.5.13	89	89	2	2
MY.NET.99.85	13	13	1	1
216.177.89.34	9	18	8	10
194.251.249.103	6	180	2	7
35.11.129.193	3	3	2	2
62.64.250.19	3	3	1	1
100.176.65.3	3	3	1	1

Looking at sources first we see 60% on MY.NET. Those are using source port of 1214 for KaZaA services and/or destination port of 6346 for gnutella.

```
Sep 29 01:21:08 MY.NET.228.6:1212 -> 65.30.225.165:1214 SYN **S*****
Sep 29 01:21:09 MY.NET.228.6:1227 -> 24.222.165.212:1214 SYN **S*****
```

```
Sep 29 01:27:58 MY.NET.208.10:1202 -> 24.208.18.185:6349 SYN **S*****
Sep 29 01:27:59 MY.NET.208.10:1205 -> 24.176.134.247:6346 SYN **S*****
Sep 29 01:27:59 MY.NET.208.10:1183 -> 65.33.90.152:6346 SYN **S*****
```

The top outside source is going after the telnet port on most of address range of MY.NET

```
Oct 3 21:32:51 209.150.29.131:1144 -> MY.NET.1.8:23 SYN **S*****
Oct 3 21:32:51 209.150.29.131:1148 -> MY.NET.1.12:23 SYN **S*****
```

<SNIP>

```
Oct 3 21:40:12 209.150.29.131:2796 -> MY.NET.248.248:23 SYN **S*****
Oct 3 21:40:15 209.150.29.131:2721 -> MY.NET.248.174:23 SYN **S*****
```

The source IP shows to be:

```
rcode = 0 (Success), ancount=1
The following answer is not verified as authentic by the server:
131.29.150.209.IN-ADDR.ARPA 43200 IN  PTR        cz-cblk-150-29-131.cyberzone.net
For authoritative answers, see:
29.150.209.IN-ADDR.ARPA       43200 IN  NS         name1.cyberzone.net
29.150.209.IN-ADDR.ARPA       43200 IN  NS         name2.cyberzone.net
Additional information:
name1.cyberzone.net  43200 IN  A          216.238.98.22
name2.cyberzone.net  43200 IN  A          216.238.98.23
```

The second outside source is also after the telnet port on most of MY.NET

```
Oct 3 22:04:25 209.10.173.30:2232 -> MY.NET.1.100:23 SYN **S*****
Oct 3 22:04:25 209.10.173.30:2234 -> MY.NET.1.102:23 SYN **S*****
```

<SNIP>

```
Oct 3 22:10:17 209.10.173.30:3068 -> MY.NET.254.59:23 SYN **S*****
Oct 3 22:10:19 209.10.173.30:3237 -> MY.NET.254.226:23 SYN **S*****
```

and belongs to:

```
Globix Corporation (NETBLK-GLOBIXBLK3)
  295 Lafayette St- 3rd Fl
  NY, NY 10012
  US
  Netname: GLOBIXBLK3
  Netblock: 209.10.0.0 - 209.11.223.255
```

For the top two destinations on MY.NET
The first is a vertical scan against MY.NET.

```
Sep 29 09:18:40 195.96.104.252:2455 -> MY.NET.5.13:71 SYN **S*****
Sep 29 09:18:40 195.96.104.252:2476 -> MY.NET.5.13:92 SYN **S*****
Sep 29 09:18:41 195.96.104.252:2474 -> MY.NET.5.13:90 SYN **S*****
Sep 29 09:18:41 195.96.104.252:2462 -> MY.NET.5.13:78 SYN **S*****
```

<SNIP>

```
Sep 29 09:19:41 195.96.104.252:3324 -> MY.NET.5.13:1729 SYN **S*****
Sep 29 09:19:44 195.96.104.252:3362 -> MY.NET.5.13:1767 SYN **S*****
Sep 29 09:19:45 195.96.104.252:3370 -> MY.NET.5.13:1776 SYN **S*****
```

from

```
252.104.96.195.IN-ADDR.ARPA 86400 IN  PTR     3dyn252.delft.casema.net
For authoritative answers, see:
104.96.195.IN-ADDR.ARPA     86400 IN  NS      sun4000.casema.nl
104.96.195.IN-ADDR.ARPA     86400 IN  NS      ns1.casema.net
Additional information:
sun4000.casema.nl   86400 IN  A       195.96.96.97
ns1.casema.net      86400 IN  A       195.96.96.33
```

The other destination on MY.NET

```
Sep 29 14:22:39 164.77.118.159:38957 -> MY.NET.99.85:333 SYN **S*****
Sep 29 14:22:39 164.77.118.159:38957 -> MY.NET.99.85:143 SYN **S*****
Sep 29 14:22:39 164.77.118.159:38957 -> MY.NET.99.85:1662 SYN **S*****
Sep 29 14:22:40 164.77.118.159:38957 -> MY.NET.99.85:1356 SYN **S*****
Sep 29 14:22:40 164.77.118.159:38957 -> 0.0.99.85:1497 SYN **S*****
Sep 29 14:22:40 164.77.118.159:38957 -> MY.NET.99.85:772 SYN **S*****
Sep 29 14:22:41 164.77.118.159:38957 -> MY.NET.99.85:26208 SYN **S*****
Sep 29 14:22:41 164.77.118.159:38957 -> MY.NET.99.85:10005 SYN **S*****
Sep 29 14:22:42 164.77.118.159:38957 -> MY.NET.99.85:4144 SYN **S*****
Sep 29 14:22:42 164.77.118.159:38957 -> MY.NET.99.85:5236 SYN **S*****
Sep 29 14:22:42 164.77.118.159:38957 -> MY.NET.99.85:147 SYN **S*****
Sep 29 14:22:43 164.77.118.159:38957 -> MY.NET.99.85:2011 SYN **S*****
Sep 29 14:22:46 164.77.118.159:38957 -> MY.NET.99.85:714 SYN **S*****
```

In this vertical scan note the same source port on each scan. The source for this scan

```
        159.118.77.164.IN-ADDR.ARPA 86400 IN  PTR        as5300-s20-149.cnt.entelchile.net
        For authoritative answers, see:
        77.164.IN-ADDR.ARPA           86400 IN  NS         polux.entelchile.net
        77.164.IN-ADDR.ARPA           86400 IN  NS         castor.entelchile.net
        Additional information:
        polux.entelchile.net   86400 IN  A          200.72.1.254
        castor.entelchile.net  86400 IN  A          200.72.1.253
```

which shows up a lot in abuse notices on the web.

OOS Data

The packets in tcpdump format in these files are packets that are Out Of Specification (OOS). As such they do not meet IP specification for one or more reasons.

The out of specification could be from one or more reasons. They could be packets crafted to cause undesired behavior in the target node's IP stack. They could be packets that are part of an encrypted connection like a VPN. The packets could be corrupted by invalid implementations of IP stacks in the sending machines. The packets could be corrupted by network hardware along the communications path. I used to think this could not happen due to the checksums at the packet layers, but then I had the experience of troubleshooting a problem caused by such a device.

It should also be noted that specifications do change. What used to be reserved bits in the IP and TCP header are used by Explicit Congestion Notification (ECN) which can be turned on in Linux kernel version 2.4 and is described in relation to NIDS by this article at SANS.

A few examples of packets in the supplied OOS data files:

```
09/29-00:43:59.673856 64.218.132.33 -> MY.NET.220.2
```

```
TCP TTL:111 TOS:0x0 ID:25569  DF MF
Frag Offset: 0x0   Frag Size: 0x22
83 0E 1E CD A3 1E 05 9E 90 E4 C8 13 BD F1 0B 3A  ...............:
11 58 9F 83 D4 2D 81 88 95 35 25 74 EB 8D 49 6B  .X...-...5%t..Ik
A9 6B                                            .k


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
09/29-00:44:03.081131 64.218.132.33 -> MY.NET.220.2
TCP TTL:111 TOS:0x0 ID:25583  DF MF
Frag Offset: 0x0   Frag Size: 0x22
83 0E 1E CD A3 1E 05 9E 90 E4 C8 13 BD F1 0B 3A  ...............:
11 58 9F 83 D4 2D 81 88 95 35 25 74 EB 8D 49 6B  .X...-...5%t..Ik
A9 6B                                            .k
```
In this example we have a resend of a packet with a small fragment size (36 bytes) that should not be that small as the more fragments (MF) flag is set. The fragment offset is 0 so this should be the first fragment - which dos not match the small fragment size.
Then we have the do not fragment (DF) flag set on an obviously fragmented packet.
The source for this packet

adsl-64-218-132-33.dsl.rcsntx.swbell.net.

which looks to be a DSL site in the southwest United States.

Now consider this packet:
```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
09/30-04:40:43.148758 65.30.62.202:0 -> MY.NET.207.158:4043
TCP TTL:110 TOS:0x0 ID:7796  DF
21S*RPA* Seq: 0x18CA01B8   Ack: 0x3A80EF40   Win: 0x5018
38 DE 50 18 22 38 7F D7 00 00 6A BF 75 17 19 B5  8.P."8....j.u...
D5 11 9D 68 00 08                                ...h..
```
A reasonable TTL, TOS, ID, Sequence ID, ACK ID, and window size in this packet.
Problems are source port of 0 and illegal combinations of the TCP flags. Not only are the two reserved flag bits set, but SYN, Reset, Push, and ACK as well.
The source IP belongs to:

ROADRUNNER-CENTRAL (NETBLK-RR-CENTRAL-2BLK)

13241 Woodland Park Road

Herndon, VA 20171

US

Netname: RR-CENTRAL-2BLK

Netblock: 65.28.0.0 - 65.31.255.255

Maintainer: RRCT

Another packet from the OOS data:
```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/02-22:21:54.290049 216.140.133.51:4944 -> MY.NET.221.30:1214
TCP TTL:44 TOS:0x0 ID:5106  DF
21S***** Seq: 0x1F16CAEC   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1380 SackOK TS: 71772845 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
10/02-22:24:38.996867 216.140.133.51:6458 -> MY.NET.221.30:1214
TCP TTL:44 TOS:0x0 ID:38347  DF
21S***** Seq: 0x3E7FCC2E   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1380 SackOK TS: 71789312 0 EOL EOL EOL EOL

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

Here we have some TCP options in addition to the two reserved TCP flag bits. The TCP option of Maximum Segment Size (MSS) is an unusual value of 1380. Ethernet is usually 1460 and other transport media are in the range of SLIP (296) or a multiple of 512. We have a Timestamp (TS) option set so the sender wants to measure packet travel times. Selective Acknowledgement (SackOK) is an extended option specified as well.
The source IP belongs to
central.clearcommerce.com.
and the destination port of 1214 we have seen is KaZaA.

We could go on and on looking at these packets and explaining what makes them out of spec. To help the site, it would be responsible to do analysis on the number of OOS packets that make up source IP, destination IP, and destination ports. To facilitate this the Unix command line tools in Chris Baker's practical assignment were used.

Source IP counts:

```
count  IP
 72 MY.NET.238.138
 14 213.67.40.166
 12 195.55.94.196
 11 217.120.160.85
 11 206.228.117.239
  9 199.183.24.194
  8 198.186.202.147
  6 24.40.4.186
  4 216.140.133.51
  3 66.26.165.221
  3 4.42.55.192
  3 24.0.154.106
  3 208.178.176.216
  3 130.207.193.70
  3 MY.NET.218.186
  2 66.122.134.171
  2 65.69.153.233
  2 65.33.91.194
  2 65.164.16.45
  2 64.218.132.33
  2 64.167.150.89
  2 24.40.66.37
  2 24.141.66.254
  2 213.76.176.114
  2 213.23.43.118
  2 213.228.45.165
  2 212.185.230.110
  2 194.82.103.75
  2 128.205.180.213
  1 66.56.24.49
  1 66.50.1.39
  1 66.114.106.23
  1 65.97.26.22
  1 65.92.252.8
  1 65.33.129.59
  1 65.30.62.202
  1 64.216.142.156
  1 64.180.117.68
  1 64.161.29.209
```

```
   1 64.123.191.66
   1 63.151.108.214
   1 24.96.46.63
   1 24.65.213.111
   1 24.60.146.180
   1 24.51.109.164
   1 24.42.235.163
   1 24.37.46.237
   1 24.28.134.6
   1 24.253.102.88
   1 24.252.187.200
   1 24.201.84.47
   1 24.158.128.63
   1 217.86.1.116
   1 217.83.3.219
   1 216.15.205.2
   1 213.93.167.132
   1 213.67.51.173
   1 213.65.218.122
   1 213.45.53.46
   1 213.45.40.162
   1 213.40.8.61
   1 213.197.71.244
   1 213.132.130.123
   1 213.109.156.1
   1 212.76.47.137
   1 212.199.47.18
   1 203.54.198.44
   1 202.168.254.178
   1 200.32.79.67
   1 199.174.222.236
   1 195.70.114.210
   1 195.113.158.226
   1 193.231.20.2
   1 192.168.1.2
   1 155.245.210.15
   1 144.132.13.182
   1 128.54.180.165
   1 128.205.192.149
   1 128.138.222.101
   1 128.102.112.231
   1 MY.NET.236.78
   1 MY.NET.228.138
   1 MY.NET.226.14
   1 MY.NET.214.6
```

Destination IP counts

```
  33 MY.NET.230.70
  17 MY.NET.207.94
   9 MY.NET.221.30
   6 MY.NET.70.113
   5 MY.NET.220.2
   4 MY.NET.253.42
   3 24.49.188.83
   3 200.52.202.7
   3 172.129.137.69
   3 129.21.113.176
   3 MY.NET.253.51
   3 MY.NET.253.43
   3 MY.NET.236.238
   3 MY.NET.221.218
   3 MY.NET.219.38
```

```
3 MY.NET.219.146
3 MY.NET.218.254
2 63.42.206.43
2 62.248.157.5
2 24.18.3.208
2 216.206.234.148
2 213.46.196.66
2 207.46.203.25
2 198.82.71.155
2 144.118.214.171
2 142.177.38.206
2 MY.NET.60.14
2 MY.NET.253.53
2 MY.NET.253.41
2 MY.NET.253.125
2 MY.NET.235.26
2 MY.NET.226.58
2 MY.NET.226.218
2 MY.NET.224.14
2 MY.NET.222.74
2 MY.NET.213.122
2 MY.NET.209.6
2 MY.NET.208.186
2 MY.NET.207.158
2 MY.NET.206.114
2 MY.NET.204.70
2 MY.NET.201.238
2 MY.NET.182.91
2 MY.NET.179.77
2 MY.NET.100.165
1 66.27.134.77
1 65.30.113.92
1 65.208.203.28
1 65.14.147.182
1 65.0.7.224
1 65.0.196.139
1 64.71.165.130
1 64.45.130.174
1 64.250.146.23
1 64.231.128.194
1 64.229.232.147
1 64.180.61.157
1 64.12.184.141
1 63.178.206.33
1 63.173.7.217
1 63.127.60.59
1 4.60.173.191
1 4.33.224.14
1 24.82.11.234
1 24.78.14.141
1 24.5.117.35
1 24.28.231.106
1 24.27.209.189
1 24.251.55.87
1 24.24.11.239
1 24.218.173.53
1 24.200.86.68
1 24.147.70.254
1 24.136.53.172
1 24.13.113.158
1 24.102.37.134
1 217.80.244.57
1 216.232.87.241
1 216.220.186.192
```

```
   1 216.115.106.129
   1 213.112.203.234
   1 213.107.133.148
   1 209.73.225.17
   1 209.240.31.97
   1 208.45.242.225
   1 207.46.203.21
   1 207.171.37.138
   1 204.186.210.57
   1 172.183.162.138
   1 170.140.57.221
   1 162.33.157.42
   1 152.163.226.25
   1 132.248.129.230
   1 128.211.217.104
   1 MY.NET.97.202
   1 MY.NET.97.186
   1 MY.NET.97.150
   1 MY.NET.6.34
   1 MY.NET.6.14
   1 MY.NET.253.114
   1 MY.NET.253.106
   1 MY.NET.234.70
   1 MY.NET.234.106
   1 MY.NET.230.182
   1 MY.NET.225.214
   1 MY.NET.224.246
   1 MY.NET.223.26
   1 MY.NET.222.82
   1 MY.NET.220.142
   1 MY.NET.219.62
   1 MY.NET.217.242
   1 MY.NET.217.226
   1 MY.NET.216.58
   1 MY.NET.216.2
   1 MY.NET.215.6
   1 MY.NET.212.134
   1 MY.NET.210.66
   1 MY.NET.208.74
   1 MY.NET.208.10
   1 MY.NET.205.218
   1 MY.NET.205.190
   1 MY.NET.204.154
   1 MY.NET.201.74
   1 MY.NET.181.144
   1 MY.NET.150.41
   1 MY.NET.115.178
   1 MY.NET.106.139
   1 MY.NET.104.76
   1 MY.NET.102.17
   1 MY.NET.100.236
```

Not much to note there, mostly ones and twos except for the top few in each list.

Looking at the top counts of destination ports:

```
  46 6346
  38 1214
  33 3400
  16 25
  12 80
   5 113
   4 6347
   3 4310
   3 3143
```

```
3 1124
2 6699
2 412
2 3950
2 21536
2 2149
2 1835
2 1797
2 1
1 8888
<SNIP>
```

KaZaA at 1214, gnutella at 6346 - either not well-behaved applications or crafted packets attempting to take advantage of these services. All of the destination ports of 3400 hit MY.NET.230.70.

The other top destination ports like Simple Mail Transport Protocol (SMTP) 25, HTTP 80, and authentication service 113 are probably crafted packets after ports that are probably open to some degree on most border filtering devices. The nodes using port 3400 are all on MY.NET and come from a wide range of ports so a link graph does not help.

Summary

A review of the findings given for the five days analyzed should be undertaken knowing that this is a limited analysis of a limited amount of data. Local hosts that indicate a possible compromise should be addressed, but investigation should not be limited to the hosts showing problems in this sampling of the snort data.

It is worth noting that no MY.NET nodes show up in the source for the nimda and Code Red II alerts. There might be several reasons for this including the rule being specific to EXTERNAL_NET -> HOME_NET.

The point being any actions prompted by the conclusions drawn from the analysis of the data in this report should NOT be limited to the hosts and alerts given by the data.

The site has installed snort so at least the first step has been taken. Other items to consider, if they have not been already, to help with the NIDS task:

1) Implement a firewall
2) Add a packet capture facility like argus
3) Use scripts or tools to correlate the snort sensor with the firewall logs and the packet capture facility
4) Review router ACLs to make sure no information leak reaches outside nodes and standard security practices for router ACLs are in place
5) Work on ruleset to reduce false positives