



# Global Information Assurance Certification Paper

Copyright SANS Institute  
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

## Interested in learning more?

Check out the list of upcoming events offering  
"Intrusion Detection In-Depth (Security 503)"  
at <http://www.giac.org/registration/gcia>

## **Paul Crutchfield**

GIAC Certified Intrusion Analyst (GCIA)  
Practical Assignment (SANS 2000 Monterey)  
December 19, 2000

### **Assignment 1 – Network Detects**

- Detect #1 – “An RPC service attack”
- Detect #2 – “Web server logs”
- Detect #3 – “Port 515 scans”
- Detect #4 – “synscan”

### **Assignment 2 – Describe an Attack – ‘twinge.c’**

### **Assignment 3 – “Analyze This” Scenario**

EVENTS OF INTEREST [EOI]  
DEFENSIVE RECCOMENDATIONS

### **Assignment 4 – Analysis Process**

### **Appendix A – EOI by Source [MS Access Report]**

## **Assignment 1 – Network Detects**

[ [1](#) [2](#) [3](#) [4](#) ]

### **Detect #1 – “An RPC service attack”**

#### **1. Source of Trace**

(Toby Miller)

A little RPC and 9088. Checked many places for this port and has come up short. Anyone have any ideas?

```
WHOIS 24.3.24.41
@Home Network (NETBLK-ATHOME) ATHOME 24.0.0.0 - 24.23.255.255
@Home Network (NETBLK-MD-COMCAST-CATV-1) MD-COMCAST-CATV-1
```

```
Nov 11 16:49:53 socretes kernel: Packet log: input REJECT eth1 PROTO=6
24.3.24.41:1218 xxx.xxx.xxx.xxx:111 L=60 S=0x00 I=28466 F=0x4000 T=53
SYN (#9)
Nov 11 17:35:57 socretes kernel: Packet log: input DENY eth1 PROTO=6
24.3.24.41:3936 xxx.xxx.xxx.xxx:9088 L=60 S=0x00 I=31177 F=0x4000 T=53
SYN #107)
Nov 11 17:36:00 socretes kernel: Packet log: input DENY eth1 PROTO=6
24.3.24.41:3936 xxx.xxx.xxx.xxx:9088 L=60 S=0x00 I=31326 F=0x4000 T=53
SYN (#107)
Nov 11 17:36:06 socretes kernel: Packet log: input DENY eth1 PROTO=6
24.3.24.41:3936 xxx.xxx.xxx.xxx:9088 L=60 S=0x00 I=31847 F=0x4000 T=53
```

SYN (#107)

[[Global Incident Analysis Center: 11/16/00](#)]

## 2. Detect was generated by:

This detect was generated from the packet filtering capability available in linux kernels. Logging is sent to syslog (stored in `/var/log/messages` on redhat). The log contains date/time, hostname, action, and details of the packet that matched a rule. For example, take the first entry:

```
Nov 11 16:49:53 socretes kernel: Packet log: input REJECT eth1 PROTO=6
24.3.24.41:1218 xxx.xxx.xxx.xxx:111 L=60 S=0x00 I=28466 F=0x4000 T=53
SYN (#9)
```

Nov 11 16:49:53	date/time
socretes	hostname
input	direction of traffic
REJECT	action
eth1	interface
PROTO=6	protocol (TCP)
24.3.24.41:1218	source IP:source port
xxx.xxx.xxx.xxx:111	destination IP:destination port
L=60	packet length
S=0x00	type of service flags
I=28466	sequence number
F=0x4000	fragment offset
T=53	time to live (TTL)
SYN	flags – SYN flag set
(#9)	this packet matched rule #9

(A good reference for the log format is available [here](#).)

## 3. Probability the source address was spoofed

This attack depends on a 3-way TCP handshake. The first step of the attack is to connect to TCP port 111, the portmapper, to obtain the mapping of RPC services to ports. Because of this, the probability of spoofing is extremely low.

## 4. Description of attack:

By exploiting a format string vulnerability in `rpc.statd`, an attacker can gain remote access to a root login interactive shell. CVE reference: [CVE-2000-0666](#).

## 5. Attack mechanism:

This is a format string vulnerability in `rpc.statd`. A format string vulnerability exploits weaknesses in C program functions for printing output. The `printf()` function is used by

programmers for output with formatting. Many parts of programs pass user data straight to the printf function without any checks. Users are able to insert formatting syntax through this input to manipulate areas of memory. The statd service has one such vulnerability in its call to syslog().

If the attacker knows which port statd is running on, the exploit can be targeted immediately to that port. In most cases the portmapper must be referenced. The portmapper gives a list of mappings for all RPC based services to ports. portmapper runs on TCP/UDP port 111.

In this attack, the attacker first tries to determine which port statd is running on with a connection to port 111. This connection matches a rule in the packet filter rule set (#9) and is REJECTed. (A REJECT differs from DENY in that an ICMP message is sent to the source saying the packet has been dropped for administrative reasons) The attack has failed at this point.

If the connection were permitted, the list of portmappings would be returned to the attacker. At this point, the attacker would determine the port for statd and connect to it. Upon connection, a string is sent to the statd service. The exploit code is contained in this string which exploits the formatting in the syslog() call.

This statd log entry provided by Laurie (see Correlations below) provides a higher level of fidelity, and thus more insight into the exploit:

```
Aug 12 06:56:58 hostp statd[284]: statd: attempt to create
"/var/statmon/sm/%08x %08x %08x %08x %08x%08x %08x %08x
%08x %08x %08x %08x %08x %08x %0242x%n%055x%n%012x%n%0192x
%nK^v ^ ( ^ ^ . #^1 F'F* FF+, NV1@/bin/sh -c echo "9088
stream tcp nowait root /bin/sh -i" >> /tmp/m;
/usr/sbin/inetd /tmp/m;"
```

The format string exploit code is located at the beginning. The tail end of the code attempts to connect an interactive shell (`/bin/sh -i`) to port 9088. This is done by creating a custom `inetd.conf` file to pass as an argument to `/usr/sbin/inetd`. This would allow unauthenticated, root shell access.

## 6. Correlations:

Lots of correlation data available for this one. First, a search for “rpc 9088” on [www.google.com](http://www.google.com) turned up:

<http://www.sans.org/y2k/081600.htm>

Laurie[@.edu] submitted a kernel log, with one of the detects showing an attacker trying to use the statd vulnerability to create a custom `inetd.conf` with `/bin/sh` tied to port 9088.

<http://lists.insecure.org/incidents/2000/Aug/0162.html>

A reply from Andreas Östling warning a user to check port 9088 after a post containing the syslog entry for a statd attack.

<http://lists.insecure.org/incidents/2000/Sep/0057.html>

Matthew Caldwell reports seeing similar scans going to ports 9704, 5000, and 9088 following an attempt to exploit statd.

A further search with “statd port” turns up a CERT advisory:

<http://www.cert.org/advisories/CA-2000-17.html>

“CERT® Advisory CA-2000-17 Input Validation Problem in rpc.statd”

## 7. Evidence of active targeting:

There is some evidence of active targeting. The attacker follows up the portmapper connection with 9088 connections, so it's not just a port 111 scan. However, the fact that the statd port is never obtained since port 111 is blocked reveals that some automation is involved. If this were a manual exploit attempt, the attacker would have known that the statd exploit was not successful, and /bin/sh was not available on 9088. Therefore the subsequent connection attempts to 9088 indicate some level of automation. A determination of whether this were part of a wider scale scan would require correlation with perimeter based logs – firewalls, NIDS located at gateways, etc.

## 8. Severity:

$(\text{Criticality} + \text{Lethality}) - (\text{System} + \text{Net Countermeasures}) = \text{Severity}$

Criticality **3**, the detect did not include information on the target, however, since it's a Unix box it's assumed to have some useful purpose other than a desktop

Lethality **5**, this attack would allow unauthenticated, remote root access

Countermeasures

System **5**, kernel level packet filtering has blocked this

Network **0**, the attack made it all the way through the network to the target operating system

$(3 + 5) - (5 + 0) = 3$

## 9. Defensive recommendation:

Turn off RPC services if not needed. There are several vulnerabilities associated with RPC based services. If these services are not needed, they should be turned off. If they are needed, they should be protected by TCP Wrappers or kernel packet filtering (which is done in this case). Linux based systems should upgrade the nfs-utils RPM to v1.9.1 or greater to mitigate this particular vulnerability. Connections to the portmapper,

TCP/UDP 111, should also be blocked at the gateway firewall. Rarely are RPC services used across the gateway.

**10. Test question:**

True or false: RPC based services run on port 111.

Answer: False, port 111, portmapper, is referenced for RPC service to port mappings only.

© SANS Institute 2000 - 2002, Author retains full rights.

## Detect #2 – “Web server logs”

### 1. Source of Trace

(Roland Grefer)

Apparently somebody tried an exploit targeted at IIS earlier this week on our web server:

```
[#1]          Date:             11/05/2000
              Access from:        209.19.244.162
              Documents Not Found:
              /scripts/..Å../winnt/system32/cmd.exe?/c dir c:\
```

Those are some quite interesting special characters used in there ... On another occasion somebody else was kind enough to leave the following trace

```
[#2]          Date:             10/08/2000
              Access from:        162.33.194.162
              Documents Not Found:
              http://www.rusftpsearch.net/cgi-
              bin/pst.pl?pstmode=writeip&psthost=12.34.56.78&pstport=80
```

where 12.34.56.78 was our web server's address.

[\[Global Incident Analysis Center: 11/14/00\]](#)

### 2. Detect was generated by:

Detects were obtained from web server logs. The logs indicate the date, source, and URL of missing documents requested. Web server logs are useful in intrusion detection as they report anomalous behavior. In this case, the logs reported these sources trying to access documents that don't exist on the server. This behavior warrants the attention of an analyst.

### 3. Probability the source address was spoofed

[#1] This attack depends on a 3-way TCP handshake. Upon establishing the port 80 connection to the webserver, the attacker requests a document with a malformed URL. If successful, the contents of c:\ is returned to the source. Based on this, spoofing is highly unlikely.

[#2] The 3-way handshake is required for this attack as well. A port 80 connection must be established before, the pst.pl GET request is sent. Spoofing is highly unlikely.

### 4. Description of attack:

[#1] By using the extended UNICODE “/” or “\” in a GET URL, an attacker can execute random commands. In this case, the attacker was attempting to execute a shell (cmd.exe) and list the contents of the C:\ directory. CVE reference: [CAN-2000-0884](#).

[#2] The source is infected with the RingZero Trojan. This Trojan randomly scans for anonymous web proxies, that is, proxies that do not authenticate and can therefore be used to hide a user's true identity. (No CVE reference)

## 5. Attack mechanism:

[#1] This is an attack targeted at IIS web servers that use an extended UNICODE set. The attacker first establishes a TCP connection with the web server. A GET request is then sent. In this request, the extended UNICODE representation of “/” and “\” are used. [Research](#) from rain forest puppy indicates that “IIS seems to decode UNICODE at the wrong instance (after path checking, rather than before).” As a result, unauthenticated users can access any files and run executables from their browsers.

As illustrated in the [securityfocus.com exploit](#) area, this vulnerability can be used in many ways. In one example, TFTP was used to transfer a file onto the web server. This was done using a GET URL with the TFTP command and syntax to transfer a text file. This was verified with a directory listing and a “type” command to view the contents of the text file—both done with GET commands to the webserver.

[#2] RingZero is a Trojan that infects windows based systems. It's purpose is to use the victim's machine to scan for anonymous web proxies. The virus is spread through email attachments. When run, an [executable](#) is extracted which registers itself to run each time windows is started. This executable scans IP addresses at random. It is looking at ports 80, 3128, and 8080. If a port is found to be active, a GET URL is sent to the target (the URL appears in the web server log). If the target is an anonymous proxy, this request will be forwarded to [www.rusftsearch.net](#), where the pst.pl script will record the target's IP address and port.

## 6. Correlations:

Both of these detects are widespread. The IIS UNICODE [#1] vulnerability surfaced in October of this year. It has generated substantial discussion on several mailing lists and web sites.

RingZero [#2] surfaced over a year ago, but is still alive in the wild. In fact, an altavista search for parts of the URL's will often return web server usage pages. These pages often show the pst.pl as a high request script or document not found.

## 7. Evidence of active targeting:

RingZero randomly selects addresses for scanning. This would not be considered active targeting. On the other hand, the IIS UNICODE detect could be considered active targeting. More information is needed to make a determination. The attacker may be



using a script that simply scans the internet for vulnerable servers. On the other hand, the attacker could be targeting the network and simply checking the web server for this specific vulnerability. To determine active targeting in this case, other logs should be reviewed for this class C network (209.19.44). Firewalls with a “deny everything” policy programmed in would be a good source for this. If other logs are not available, active targeting should be assumed.

## 8. Severity:

$(\text{Criticality} + \text{Lethality}) - (\text{System} + \text{Net Countermeasures}) = \text{Severity}$

### [#1]

Criticality **3**, this is a web server – it probably holds information which could cause harm to the organization if the confidentiality, integrity, or availability (aka. CIA triad) were affected

Lethality **0**, this has some assumption built in: from the wording of the notes which precede the detect, the target web server is not IIS based

*Countermeasures*

System **3**, security posture (OS patches, TCP wrappers, etc.) for this system is not known, however, logging is enabled on the web server

Network **0**, the attack made it through to the web server, so it is not blocked at the network level

$(3 + 0) - (3 + 0) = 0$

### [#2]

Criticality **3**, this is a web server – it probably holds information which could cause harm to the organization if the confidentiality, integrity, or availability (aka. CIA triad) were affected

Lethality **0**, the URL was logged as a document not found on the server, this is not a proxy server

*Countermeasures*

System **3**, security posture (OS patches, TCP wrappers, etc.) for this system is not known, however, logging is enabled on the web server

Network **0**, the attack made it through to the web server, so it is not blocked at the network level

$(3 + 0) - (3 + 0) = 0$

## 9. Defensive recommendation:

At the network, intrusion detection could catch these with a simple string match on TCP traffic with a destination of port 80. The arachNIDS archive currently contains three signatures for the IIS UNICODE [#1] attack: [IDS432](#), [IDS433](#), [IDS434](#).

Example snort rule given in [IDS434](#):

```
alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: "IDS434/web-iis-unicode-traversal-backslash"; flags: AP; content: "..|25|c1|25|9c"; nocase;)
```

This rule alerts on TCP traffic from outside the network destined to any host inside the network, port 80. It looks for the ACK and PUSH flags on a TCP packet and the culprit extended UNICODE characters within the payload. (This assumes the GET requests are sent on the 3<sup>rd</sup> step of the TCP handshake with the ACK flag set and a PUSH flag indicating no more data is coming. Web communications rarely send ACKs by themselves, but rather piggyback the flags with data.)

arachNIDS does not contain a RingZero [#2] signature, but one can easily be derived:

```
alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: "ringzero"; flags: AP; content: "http://www.rusftpsearch.net/cgi-bin/pst.pl"; nocase;)
```

Moving closer to the application itself, host based IDS is also available. There is a product called [Perfecto](#) that will act as a proxy in front of your web server and defend against web-based attacks. [Perfecto](#) learns what appropriate behavior is for web applications, or forms, and makes decisions based on this. Both of these URL's would be considered deviant from expected behavior. The GET request would be blocked, logged, and reported to the administrator.

#### 10. Test question:

Which logs would contain the best level of fidelity for analysis of attacks on cgi-bin based web applications?

- a) Firewall
- b) NIDS
- c) Web server
- d) TCP wrappers

Answer: **c**, web server logs provide more insight into the GET/POST requests used in cgi-bin scripts.

### Detect #3 – “Port 515 scans”

#### 1. Source of Trace

(Patrick Prue)

Been logging a bunch of these scans over the last 3 days from various sources.

Was wondering what they were possibly looking for on port 515 . Haven't really been able to find anything with regards to what it possibly is .

```

[**] IDS06 - MISC-Source Port Traffic 20 TCP [**]
10/31-18:47:21.973424 208.184.219.253:20 -> x.x.x.2:515
TCP TTL:244 TOS:0x0 ID:59824
**S***** Seq: 0x1000000 Ack: 0x0 Win: 0x3FFF

[**] IDS06 - MISC-Source Port Traffic 20 TCP [**]
10/31-18:47:53.157274 208.184.219.253:20 -> x.x.x.4:515
TCP TTL:244 TOS:0x0 ID:25472
**S***** Seq: 0x1000000 Ack: 0x0 Win: 0x3FFF

[**] IDS06 - MISC-Source Port Traffic 20 TCP [**]
10/31-18:52:02.641247 208.184.219.253:20 -> x.x.x.20:515
TCP TTL:244 TOS:0x0 ID:12802
**S***** Seq: 0x1000000 Ack: 0x0 Win: 0x3FFF

[**] IDS06 - MISC-Source Port Traffic 20 TCP [**]
10/31-19:03:59.876453 208.184.219.253:20 -> x.x.x.66:515
TCP TTL:244 TOS:0x0 ID:9143
**S***** Seq: 0x1000000 Ack: 0x0 Win: 0x3FFF

```

[[Global Incident Analysis Center: 11/2/00](#)]

## 2. Detect was generated by:

These are snort logs. Apparently Patrick has a signature setup to alarm on incoming connections with low numbered source ports. In this case, port 20, the FTP data port, is being used.

[**] IDS06 - MISC-Source Port Traffic 20 TCP [**]	<- Name of the signature
10/31-18:52:02.641247 208.184.219.253:20 -> x.x.x.20:515	<- Date-time Source IP:port ->
Target IP:port	
TCP TTL:244 TOS:0x0 ID:12802	<- Protocol, Time to live value,
type of service flags, fragment ID	
**S***** Seq: 0x1000000 Ack: 0x0 Win: 0x3FFF	<- flags (SYN),
sequence number, ack. number, window size	

## 3. Probability the source address was spoofed

This attack depends on a 3-way TCP handshake. The first step of the attack is to connect to TCP port 515, the Unix print spooler port. At a minimum, the source would need to receive a SYN-ACK to determine if this service is running. Because of this, the probability of spoofing is extremely low.

## 4. Description of attack:

This is a recon attempt for hosts servicing TCP port 515. The attacker is attempting to map the target network. By using source port 20, some perimeter devices may let these packets in. From the [whitehats.com](#) archive:

This problem is due to design flaws in the consideration of some packetfilters. A typical ruleset would have a rule to allow FTP response traffic, by allowing traffic in that had the source port 20. However, an attacker can easily set their source port to 20 using a tool like netcat, and circumvent this type of primitive filter.

Port 20 is used for FTP data. FTP commands are passed on port 21. When a file is transferred, a connection is established on port 20. The target initiates the connection back to the source. For a firewall to intelligently allow this connection, it must be stateful. In other words, it knows an FTP session has been established between the source (internal host, protected by firewall) and the target. Based on this “stateful” knowledge, a firewall can make an intelligent decision to let this connection attempt through. The attacker is hoping there are no stateful inspection devices like this on the network.

#### **5. Attack mechanism:**

TCP port 515 is the Unix print spooler port. This port is used for network printing between windows hosts. Win2k also supports this network printing as well.

Microsoft TCP/IP Printing Services, aka Print Services for Unix, allows an attacker to cause a denial of service via a malformed TCP/IP print request.

[CVE-2000-0232](#) [securityfocus.com](http://securityfocus.com)

#### **6. Correlations:**

Search of source address returned no results. Neither could I find any other scans of port 515 using source port 20. Just other scans for port 515:

11/30: [security@auckland](mailto:security@auckland) reports incoming 515 connections from the @home network.

11/28: [security@auckland](mailto:security@auckland) reports incoming 515 connections from 128.255.130.28.

11/27: Stephen Odak reports incoming 515 connections.

No significant correlations available at this time.

#### **7. Evidence of active targeting:**

This is a scan of the network to find open 515 ports. Chances are if this source were targeting this network in general, other alarms would be triggered as well. The attacker would be looking for several vulnerabilities to exploit remotely. Based on the data available for analysis, this is a general scan for port 515 vulnerabilities through a range of network addresses. There is no evidence of active targeting.

#### **8. Severity:**

(Criticality + Lethality) - (System + Net Countermeasures) = Severity

Criticality 2, if the mapping is successful, the attacker would have knowledge of possible denial of service targets

Lethality 3, this is a denial of service attack

Countermeasures

System 0, it is unknown whether the target network has this service running anywhere

Network 3, snort picked up the mapping attempt, but it is unknown if these were blocked on the network

$$(2 + 3) - (0 + 3) = 2$$

**NOTE:** Based on the sophistication of the attacker (by using source port 20), I'd add a point or two to this!

### 9. Defensive recommendation:

This attack should be blocked at the perimeter. In most instances, there will not be a reason to let print requests through the perimeter. A firewall with a deny all but what is explicitly permitted would satisfy this. See [SANS Top 10](#), Appendix B, for further information on how and what to block at the perimeter. Further, based on the sophistication of this attack, a watchlist for this source network (208.184.219) is recommended.

### 10. Test question:

A stateless perimeter security device would have trouble blocking which packet?

- a. x.x.x.x:20 -> x.x.x.2:515 Flags: SYN
- b. x.x.x.x :8821 -> x.x.x.2:515 Flags: SYN
- c. x.x.x.x:22112 -> x.x.x.2:515 Flags: FIN, SYN
- d. x.x.x.x:80 -> x.x.x.2:1824 Flags: SYN

Correct answer: **a**, incoming port 20 connections need to be established for FTP data transfers. To successfully defend, yet provide FTP communications, the device must have knowledge of (or keep the "state" of) current outgoing FTP connections.

## Detect #4 – "synscan"

### 1. Source of Trace

(Laurie@.edu)

Bell Global Network Operations, Ottawa Ontario, CA (again)

```
Jan 19 22:10:07, Jan 21 15:37:48
Jun 29 05:56:35 207.236.111.226:21 -> z.y.w.98:21 SYNFIN **SF****
Jun 29 05:56:51 207.236.111.226:21 -> z.y.w.98:21 SYNFIN **SF****
Jun 29 05:57:54 207.236.111.226:21 -> z.y.w.98:21 SYNFIN **SF****
```

```
Jun 29 05:57:57 207.236.111.226:20755 -> z.y.w.98:21 SYN **S*****
Jun 29 05:58:02 207.236.111.226:1628 -> z.y.w.98:53 UDP
Jun 29 05:58:10 207.236.111.226:21 -> z.y.w.98:21 SYNFIN **SF*****
Jun 29 05:58:11 207.236.111.226:22819 -> z.y.w.98:21 SYN **S*****
```

```
-----
[**] SCAN-SYN FIN [**]
06/29-05:56:34.871172 207.236.111.226:21 -> z.y.w.98:21
TCP TTL:27 TOS:0x0 ID:39426
**SF***** Seq: 0x7E45DE1F Ack: 0x50A11826 Win: 0x404
00 00 00 00 00 00 .....
[**] SCAN-SYN FIN [**]
06/29-05:56:50.946947 207.236.111.226:21 -> z.y.w.98:21
TCP TTL:27 TOS:0x0 ID:39426
**SF***** Seq: 0x1C4719EB Ack: 0x1E77A02F Win: 0x404
00 00 00 00 00 00 .....
[**] SCAN-SYN FIN [**]
06/29-05:57:53.886933 207.236.111.226:21 -> z.y.w.98:21
TCP TTL:27 TOS:0x0 ID:39426
**SF***** Seq: 0x2627C01C Ack: 0x61572CE9 Win: 0x404
00 00 00 00 00 00 .....
[**] SCAN-SYN FIN [**]
06/29-05:58:10.093075 207.236.111.226:21 -> z.y.w.98:21
TCP TTL:27 TOS:0x0 ID:39426
**SF***** Seq: 0x43F05EC6 Ack: 0x6F270653 Win: 0x404
00 00 00 00 00 00 .....
```

[\[Global Incident Analysis Center: 7/2/00\]](#)

## 2. Detect was generated by:

These are snort logs. The first set of entries are fast alerts. This is an abbreviated format that provides a timestamp, source and destination, as well as the options that were included in the rule (in this case, the SIN and FIN flags being set simultaneously). The second set of logs provides more detail – the entire packet.

## 3. Probability the source address was spoofed

This attack depends on a 3-way TCP handshake. The source is trying to determine if the FTP service, port 21, is running on the target. The source depends on receiving a response (in the form of a RST) from the target to determine this. Because of this, the probability of spoofing is extremely low.

## 4. Description of attack:

This is a network recon attempt. The source is attempting to map well-known ports on the target network, in this case, FTP and DNS.

## 5. Attack mechanism:

There are two suspicious characteristics present in this attack:

1. Source port = Target port
2. SYN **AND** FIN flags are set

The first characteristic is not exactly illegal, but would not be found in normal TCP/IP traffic. The SYNFIN flags should never be set together. There is no reason for this. The response a target gives to these illegal packets can reveal the target's architecture and O.S. type. RFC's do not specify how stack programmer's should handle these illegal packets. Therefore, different operating systems behave differently and send different responses when receiving these illegal packets. Some may send a RST, some may send nothing. By poking and prodding with illegal packets like this, the attacker can reference a database of known responses to determine the target O.S. This is the basis of TCP/IP fingerprinting methods found in tools such as 'nmap' and 'queso'.

## 6. Correlations:

The trace provided in the arachNIDS synscan signature definition, [IDS441](#), exhibit the same qualities as our trace.

```
11/30-17:54:10.623674 10.200.1.100:218 -> 10.1.1.38:218
TCP TTL:42 TOS:0x0 ID:39426
**SF*** Seq: 0x55A0EF7B Ack: 0x40F9DC84 Win: 0x404
```

- source port, destination port are same
- ID = 39426
- Window size = 0x404

## 7. Evidence of active targeting:

With the data given, there is no evidence of active targeting. This is probably part of a larger range scan.

## 8. Severity:

$(\text{Criticality} + \text{Lethality}) - (\text{System} + \text{Net Countermeasures}) = \text{Severity}$

Criticality 2, if the mapping is successful, the attacker would have knowledge of possible denial of service targets

Lethality 0, this is just a probe/mapping attempt.

Countermeasures

System 0, it is unknown whether the target network has this service running anywhere

Network 3, snort picked up the mapping attempt, but it is unknown if these were blocked on the network

$(2 + 0) - (0 + 3) = -1$

## 9. Defensive recommendation:

There are many IDS signatures for catching illegal packets or suspicious traffic such as this. Snort signature - [IDS441](#):

```
alert TCP $EXTERNAL any -> $INTERNAL any (msg: "IDS441/probe-Synscan-Portscan";
id: 39426; flags: SF;)
```

#### 10. Test question:

Which of the following packets is illegal?

- a. x.x.x.x:20 -> x.x.x.2:515 Flags: SYN, ACK
- b. x.x.x.x :515 -> x.x.x.2:515 Flags: ACK
- c. x.x.x.x:22112 -> x.x.x.2:515 Flags: FIN, SYN
- d. x.x.x.x:80 -> x.x.x.2:515 Flags: SYN

Answer: c, the SYN and FIN flags should never be set together.

### Assignment 2 – Describe an Attack – ‘twinge.c’

[twinge.c](#) [source: technotronic.com]

‘twinge’ generates most ICMP types to send to a remote host with spoofed sources. These ICMP packets may have adverse affects on some TCP/IP stack implementations – specifically Win32. This technique could be used for Denial of Service [DoS] attacks. From the source:

```
/*
twinge.c - by sinkhole@dos.org [6/99]

this cycle through all the possible icmp types and subtypes and
send to target host, 1 cycle == 1 run thru all of em

Crashes almost all Windows boxes over a LAN.

DISCLAIMER:
This is a PoC (Proof Of Concept) program for educational purposes
only. Using this program on public networks where other people
are affected by your actions is HIGHLY ILLEGAL and is not what
this is made for.

for without help from ryan this wouldnt have been coded. =)
*/
```

Code was compiled and executed on a Redhat 6.2 box. Target was a Win2k Professional workstation. The generated ICMP packets did not appear to have an affect on the target. Apparently Microsoft has addressed these issues in their latest OS release as these ICMP packets were known to wreak havoc on Win32 boxes in the past.

#### Breakdown of ‘tcpdump –nvvx’ output:

```
21:17:06.709813 eth0 > 13.129.59.21 > 192.168.142.1: icmp: echo reply (ttl 255, id
61978)
4500 001d f21a 0000 ff01 3285 0d81 3b15
```



c0a8 8e01 0000 ffff 0000 0000 00

13.129.59.21 spoofed source IP  
192.168.142.1 target

**IP:**

4500 001d version, length, TOS, total header length  
f21a 0000 identification, flags, fragment offset  
ff01 3285 TTL, protocol, header checksum  
0d81 3b15 source IP  
c0a8 8e01 destination IP

**IP:**

0000 ffff message type, message code type, checksum  
0000 0000 00 type specific parameter [32 bits], data

**Start of tcpdump capture:**

Kernel filter, protocol ALL, datagram packet socket  
tcpdump: listening on all devices

**Type #0: Echo Reply:**

21:17:06.709813 eth0 > 13.129.59.21 > 192.168.142.1: icmp: **echo reply** (ttl 255, id 61978)

4500 001d f21a 0000 ff01 3285 0d81 3b15  
c0a8 8e01 0000 ffff 0000 0000 00

[ ICMP Header: Message type: 00, code type: 00, Checksum: ffff ]

**Types #1 and #2 are Unassigned:**

21:17:06.750523 eth0 > 111.142.235.73 > 192.168.142.1: icmp: **type-#1** (ttl 255, id 11128)

4500 001d 2b78 0000 ff01 e6e5 6f8e eb49  
c0a8 8e01 0100 feff 0000 0000 00

21:17:06.771878 eth0 > 161.173.209.126 > 192.168.142.1: icmp: **type-#2** (ttl 255, id 15660)

4500 001d 3d2c 0000 ff01 bcdd a1ad d17e  
c0a8 8e01 0200 fdff 0000 0000 00

**Type #3: Network Unreachable:**

21:17:06.786327 eth0 > 32.42.4.52 > 192.168.142.1: [icmp] (ttl 255, id 19880)  
4500 001d 4da8 0000 ff01 fb2f 202a 0434  
c0a8 8e01 0300 fcff 0000 0000 00 [network unreachable]

21:17:06.800118 eth0 > 32.42.4.52 > 192.168.142.1: [icmp] (ttl 255, id 19880)  
4500 001d 4da8 0000 ff01 fb2f 202a 0434  
c0a8 8e01 0301 fcfe 0000 0000 00 [host unreachable]

21:17:06.809113 eth0 > 32.42.4.52 > 192.168.142.1: [icmp] (ttl 255, id 19880)  
4500 001d 4da8 0000 ff01 fb2f 202a 0434  
c0a8 8e01 0302 fcfd 0000 0000 00 [protocol unreachable]

21:17:06.819585 eth0 > 32.42.4.52 > 192.168.142.1: [icmp] (ttl 255, id 19880)  
4500 001d 4da8 0000 ff01 fb2f 202a 0434  
c0a8 8e01 0303 fcfc 0000 0000 00 [port unreachable]

21:17:06.825515 eth0 > 32.42.4.52 > 192.168.142.1: [icmp] (ttl 255, id 19880)  
4500 001d 4da8 0000 ff01 fb2f 202a 0434  
c0a8 8e01 0304 fcfb 0000 0000 00 [packet to big, DF bit

set]  
21:17:06.842751 eth0 > 32.42.4.52 > 192.168.142.1: [icmp] (ttl 255, id 19880)  
4500 001d 4da8 0000 ff01 fb2f 202a 0434  
c0a8 8e01 0305 fcfa 0000 0000 00 [source route failed]

21:17:06.852835 eth0 > 32.42.4.52 > 192.168.142.1: [icmp] (ttl 255, id 19880)  
4500 001d 4da8 0000 ff01 fb2f 202a 0434

```

c0a8 8e01 0306 fcf9 0000 0000 00 [destination network
unknown error]
21:17:06.861397 eth0 > 32.42.4.52 > 192.168.142.1: [|icmp] (ttl 255, id 19880)
4500 001d 4da8 0000 ff01 fb2f 202a 0434
c0a8 8e01 0307 fcf8 0000 0000 00 [destination host
unknown error]
21:17:06.870986 eth0 > 32.42.4.52 > 192.168.142.1: [|icmp] (ttl 255, id 19880)
4500 001d 4da8 0000 ff01 fb2f 202a 0434
c0a8 8e01 0308 fcf7 0000 0000 00 [source host isolated
error]
21:17:06.877339 eth0 > 32.42.4.52 > 192.168.142.1: [|icmp] (ttl 255, id 19880)
4500 001d 4da8 0000 ff01 fb2f 202a 0434
c0a8 8e01 0309 fcf6 0000 0000 00 [destination network
admin prohibited]

```

Type #4: Source Quench:

```

21:17:06.896608 eth0 > 119.231.110.104 > 192.168.142.1: icmp: source quench
Offending pkt: [|ip] (ttl 255, id 52060)
4500 001d cb5c 0000 ff01 bb89 77e7 6e68
c0a8 8e01 0400 fbff 0000 0000 00

```

Type #5: Redirect:

```

21:17:06.906971 eth0 > 152.191.132.29 > 192.168.142.1: [|icmp] (ttl 255, id 8176)
4500 001d 1ff0 0000 ff01 3069 98bf 841d
c0a8 8e01 0500 faff 0000 0000 00 [network error]
21:17:06.913510 eth0 > 152.191.132.29 > 192.168.142.1: [|icmp] (ttl 255, id 8176)
4500 001d 1ff0 0000 ff01 3069 98bf 841d
c0a8 8e01 0501 fafe 0000 0000 00 [host error]
21:17:06.922702 eth0 > 152.191.132.29 > 192.168.142.1: [|icmp] (ttl 255, id 8176)
4500 001d 1ff0 0000 ff01 3069 98bf 841d
c0a8 8e01 0502 fafd 0000 0000 00 [TOS and network error]
21:17:06.932742 eth0 > 152.191.132.29 > 192.168.142.1: [|icmp] (ttl 255, id 8176)
4500 001d 1ff0 0000 ff01 3069 98bf 841d
c0a8 8e01 0503 fafc 0000 0000 00 [TOS and host error]

```

Type #6: Alternate Host Address:

```

21:17:06.942121 eth0 > 199.173.70.82 > 192.168.142.1: icmp: type-#6 (ttl 255, id
61007)
4500 001d ee4f 0000 ff01 70e6 c7ad 4652
c0a8 8e01 0600 f9ff 0000 0000 00

```

Type #7: Unassigned:

```

21:17:06.952554 eth0 > 32.82.18.71 > 192.168.142.1: icmp: type-#7 (ttl 255, id
31792)
4500 001d 7c30 0000 ff01 be6c 2052 1247
c0a8 8e01 0700 f8ff 0000 0000 00

```

Type #8: Echo:

```

21:17:06.963567 eth0 > 118.242.97.123 > 192.168.142.1: icmp: echo request (ttl
255, id 22260)
4500 001d 56f4 0000 ff01 3dd4 76f2 617b
c0a8 8e01 0800 f7ff 0000 0000 00

```

Type #9: Router Advertisement:

```

21:17:06.972993 eth0 > 202.194.65.49 > 192.168.142.1: icmp: router advertisement
lifetime 0 0: [size 0] (ttl 255, id 32294)
4500 001d 7e26 0000 ff01 e31b cac2 4131
c0a8 8e01 0900 f6ff 0000 0000 00

```

Type #10: Router Selection:

```

21:17:06.982902 eth0 > 6.229.235.101 > 192.168.142.1: icmp: router solicitation
(ttl 255, id 51447)
4500 001d c8f7 0000 ff01 b1f3 06e5 eb65
c0a8 8e01 0a00 f5ff 0000 0000 00

```

Type #11: Time exceeded:

```

21:17:06.992532 eth0 > 214.101.213.90 > 192.168.142.1: [|icmp] (ttl 255, id 57760)
4500 001d e1a0 0000 ff01 dfd4 d665 d55a
c0a8 8e01 0b00 f4ff 0000 0000 00 [TTL expired]
21:17:07.001628 eth0 > 214.101.213.90 > 192.168.142.1: [|icmp] (ttl 255, id 57760)
4500 001d e1a0 0000 ff01 dfd4 d665 d55a
c0a8 8e01 0b01 f4fe 0000 0000 00 [fragment reassembly
timeout]

```

Type #12: Parameter Problem:

```

21:17:07.015131 eth0 > 22.16.121.4 > 192.168.142.1: icmp: parameter problem -
octet 0 Offending pkt: [|ip] (ttl 255, id 63583)

```

```

4500 001d f85f 0000 ff01 e5c1 1610 7904
c0a8 8e01 0c00 f3ff 0000 0000 00

Type #13: Timestamp:
21:17:07.016078 eth0 > 23.1.38.121 > 192.168.142.1: icmp: time stamp request (ttl
255, id 24182)

4500 001d 5e76 0000 ff01 d145 1701 2679
c0a8 8e01 0d00 f2ff 0000 0000 00

Type #14: Timestamp reply:
21:17:07.017355 eth0 > 92.249.166.45 > 192.168.142.1: icmp: time stamp reply (ttl
255, id 10783)

4500 001d 2a1f 0000 ff01 3ff0 5cf9 a62d
c0a8 8e01 0e00 f1ff 0000 0000 00

Type #15: Information Request:
21:17:07.018175 eth0 > 8.222.246.98 > 192.168.142.1: icmp: information request
(ttl 255, id 8432)

4500 001d 20f0 0000 ff01 4d05 08de f662
c0a8 8e01 0f00 f0ff 0000 0000 00

Type #16: Information Reply:
21:17:07.018771 eth0 > 240.223.235.87 > 192.168.142.1: icmp: information reply
(ttl 255, id 64098)

4500 001d fa62 0000 ff01 969b f0df eb57
c0a8 8e01 1000 efff 0000 0000 00

Type #17: Address Mask Request:
21:17:07.019669 eth0 > 27.120.76.77 > 192.168.142.1: icmp: address mask request
(ttl 255, id 65234)

4500 001d fed2 0000 ff01 069e 1b78 4c4d
c0a8 8e01 1100 eeff 0000 0000 00

```

### **Observations:**

- spoofed source IP and IP ID change when type increments, but not when type code increments.
- Changes TTL to 255 compared to standard 64 (see below).
- Payload [parameters and data] stays constant: 0000 0000 00

A normal echo request/reply (ping) generated from Redhat 6.2 box to target:

```

22:35:20.345043 eth0 > 192.168.142.128 > 192.168.142.1: icmp: echo request (ttl 64, id
40801)

4500 0054 9f61 0000 4001 3d75 c0a8 8e80
c0a8 8e01 0800 dbac 6c06 0000 78d7 3e3a
0938 0500 0809 0a0b 0c0d 0e0f 1011 1213
1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
3435 3637

22:35:20.346345 eth0 < 192.168.142.1 > 192.168.142.128: icmp: echo reply (ttl 128, id
11992)

4500 0054 2ed8 0000 8001 6dfe c0a8 8e01
c0a8 8e80 0000 e3ac 6c06 0000 78d7 3e3a
0938 0500 0809 0a0b 0c0d 0e0f 1011 1213
1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
3435 3637

```

### **Defensive recommendations:**

Block ICMP at the firewall. Note, however, that blocking type 3 (dest. unreachable) and type 11 (time exceeded) may cause problems with IP communications.

Possible Snort signature: **[untested]**

```

alert ICMP $EXTERNAL any -> $INTERNAL any (msg: "icmp-unassigned-1"; itype: 1;)

```

## Assignment 3 – “Analyze This” Scenario

This report is based on Snort alarm data recorded from 11 Aug through 14 Sep, or **68,597** alarms total.

**NOTE!** Your sensor appeared to be missing several days of log data. Specifically, 12-14 Aug, 1 Sep, and 4 Sep.

### Update on past reporting

In past reports, `MY.NET.5.37` has been identified as possibly compromised. However, lack of activity in our data logs show the security issues on this host have been addressed. There is still ‘watchlist’ activity to `MY.NET.100.230`, but this appears to be normal mail traffic. We are still catching wu-ftp exploits. This vulnerability has serious consequences (root access). We are still seeing RPC traffic as well. RPC is plagued with vulnerabilities and should be blocked at the perimeter as there are very few cases where RPC communications are needed across the internet. No alarms were seen for the Trinoo server at `MY.NET.97.112`.

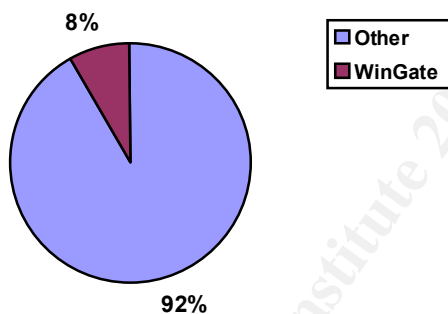
## EVENTS OF INTEREST [EOI]

Source	whois	Target	Activity	Date	Recommended action
<a href="#">128.61.105</a>	Georgia Institute of Technology	MY.NET.218	Recon	8-Sep	Implement watchlist this source
<a href="#">130.149.41</a>	Technische Universitaet Berlin, Germany	MY.NET.223, MY.NET.217	Recon	17-Sep	Implement watchlist this source
<a href="#">141.223.124</a>	Pohang Institute of Science and Technology, Republic of Korea	MY.NET.100, MY.NET.6		19-Sep	Check targets for compromise; block source; implement watchlist
			<b>SUN RPC ACCESS</b>		
<a href="#">151.196.73</a> .119	Windermer Information Systems Technology, Annapolis, MD	MY.NET.253	Recon	8-Sep	Implement watchlist this source
<a href="#">18.116.0</a>	Massachusetts Institute of Technology	MY.NET.100, MY.NET.15, MY.NET.6	Sun RPC access, Recon	18-Aug	Check targets for compromise; block source; implement watchlist
<a href="#">192.55.91</a>	NASA Lewis Network Control Center	MY.NET.5	Recon	19/20-Aug	Implement watchlist this source
<a href="#">202.187.24</a>	Universiti Tun Abdul Razak [APNIC]	MULTIPLE	Recon	MULTIPLE	Implement watchlist this source
<a href="#">205.128.11</a> .157	HeadHunter.net, Atlanta, GA	MY.NET.1	Recon	MULTIPLE	Implement watchlist this source
<a href="#">205.188.153</a>	America Online, Inc	MULTIPLE	Sun RPC access	MULTIPLE	Check targets for compromise; block source; implement watchlist
<a href="#">205.188.179</a>	America Online, Inc	MY.NET.217	Sun RPC access	15/16-Aug	Check targets for compromise; block source; implement watchlist
<a href="#">207.151.147</a>	Los Nettos, Marina del Rey, CA	MY.NET.60	Recon	16-Aug	Implement watchlist this source
<a href="#">209.160.238</a>	Brooks Fiber Properties, Inc., Sacramento, CA	MULTIPLE	Sun RPC access	19-Aug	Check targets for compromise; block source; implement watchlist
<a href="#">209.218.228</a> .201	RND Networks, Mahwah, NJ	MY.NET.1	Recon	11/15-Aug	Implement watchlist this source
<a href="#">210.100.199</a>	Korea Telecom	MY.NET.100, MY.NET.6	Sun RPC access	3-Sep	Check targets for compromise; block source; implement watchlist
<a href="#">210.101.101</a>	Korea Telecom	MY.NET.6	Sun RPC access, Recon	2-Sep	Check targets for compromise; block source; implement watchlist
<a href="#">210.61.144</a>	Abnet Information Co., Ltd,	MULTIPLE	Exhaustive Recon	11-Sep	Source has performed an exhaustive SIN-I

	Taiwan					on the entire network – recommend block implement watchlist this source
<a href="#">212.204.196</a>	The Netherlands	MY.NET.6	Sun RPC access	2/7-Sep		Check targets for compromise; block source implement watchlist
<a href="#">213.25.136</a>	E-SOLUTIONS, Poland	MULTIPLE	<b>Exhaustive Recon</b>	7-Sep		Source has performed an exhaustive SIN-I on the entire network – recommend block implement watchlist this source
<a href="#">216.15.191.130</a>	Twistedhumor.com, San Diego, CA	MY.NET.6, MY.NET.253	Recon		MULTIPLE	Implement watchlist this source
<a href="#">24.17.189.83</a>	@Home Network	MY.NET.150.24, MY.NET.99.104, MY.NET.202.190, MY.NET.202.202	<b>wu-ftpd exploit attempts</b>	8-Sep		Source has attempted known exploits to the server. Check targets for compromise. Recommend block and/or implement watchlist this source
<a href="#">24.180.134</a>	@Home Network	MY.NET.208	Recon	11-Sep		Implement watchlist this source
<a href="#">24.23.198</a>	@Home Network	MY.NET.217	Recon	17/18-Aug		Implement watchlist this source
<a href="#">24.3.161</a>	@Home Network	MY.NET.145	Recon	5,13,13-Aug		Implement watchlist this source
<a href="#">24.68.58</a>	@Home Network	MY.NET.210, MY.NET.217	<b>Tiny Fragments</b>	11,13-Sep		Suspect denial of service attempts against source. Recommend block and/or implement watchlist this source.
<a href="#">62.76.42</a>	SSAU - Samara State Aerospace University, Russia	MY.NET.1, MY.NET.212	<b>Tiny Fragments</b>	14-Sep		Suspect denial of service attempts against source. Recommend block and/or implement watchlist this source.
<a href="#">63.226.208</a>	Netpoint, Springville, UT	MY.NET.253	Recon	2-Sep		Implement watchlist this source
<a href="#">64.80.63</a>	CollegePark/LexingtonCrossing, Gainseville, FL	MULTIPLE	Recon		MULTIPLE	Implement watchlist this source

### Anonymous Web Proxy (WinGate)

6193 alerts



These alerts indicate possible anonymous web proxies setup inside your network. An anonymous web proxy is used to hide the sources true identity when they surf to internet sites. There are several Trojans that automatically scan the internet for these proxies. This is the primary cause for what may be considered false positives. However, the top 3 targets in this category indicate possible proxy configurations and should be investigated immediately:

SOURCE	TARGET	DESCRIPTION
MULTIPLE	MY.NET.60.11	Target is being used as an anonymous web proxy, port 1080 <b>Outgoing telnets</b> , 8/11-16:11
MULTIPLE	MY.NET.60.8	Target is being used as an anonymous web proxy, port 1080 <b>Incoming telnets</b> from 159.226.45.108 [8/11 01:27-02:25]
MULTIPLE	MY.NET.100.2	Several hosts from 151.17.144 network attempted port 1080 access

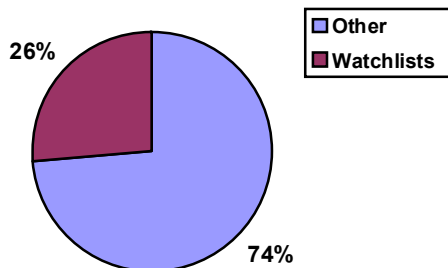
Top WinGate targets (10 or more hits)

<b>10</b>	MY.NET.185.20	<b>12</b>	MY.NET.202.118	<b>15</b>	MY.NET.97.115	<b>20</b>	MY.NET.98.162
	MY.NET.97.156		MY.NET.217.46		MY.NET.97.187	<b>21</b>	MY.NET.98.185
	MY.NET.97.182		MY.NET.98.131	<b>16</b>	MY.NET.97.169	<b>22</b>	MY.NET.97.192
	MY.NET.98.199		MY.NET.98.137		MY.NET.98.157	<b>23</b>	MY.NET.98.124

- |    |  |    |  |    |   |     |                                 |
|----|--|----|--|----|---|-----|---------------------------------|
| 11 | MY.NET.152.45<br>MY.NET.98.108<br>MY.NET.98.127<br>MY.NET.98.129 | 13 | MY.NET.98.197<br>MY.NET.203.218<br>MY.NET.60.38<br>MY.NET.97.226<br>13 MY.NET.98.170 | 17 | MY.NET.53.48<br>MY.NET.97.215<br>MY.NET.98.106  | 31  | MY.NET.98.111<br>MY.NET.222.198 |
|    |  | 14 | MY.NET.201.50  | 18 | MY.NET.97.119<br>MY.NET.97.237<br>MY.NET.98.193 | 39  | MY.NET.60.16                    |
|    |  |    |  | 19 | MY.NET.98.130                                   | 91  | <b>MY.NET.100.2</b>             |
|    |  |    |  |    |   | 139 | <b>MY.NET.60.8</b>              |
|    |  |    |  |    |   | 177 | <b>MY.NET.60.11</b>             |

### Watchlists

24,754 alerts



Watchlists raise alerts based on source IP alone. They are usually implemented for networks that have known to be hostile. There appear to be two watchlists configured:

12.179.44.0 - 212.179.44.63

ISDN Net Ltd.  
Israel

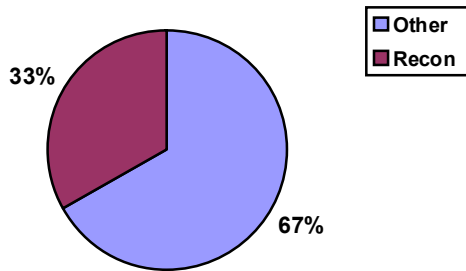
159.226.0.0  
Academy of Sciences (NET-NCFC)

The Computer Network Center Chinese  
Beijing, China

SOURCE	TARGET	DESCRIPTION
159.226.45.108	MY.NET.6.7	These sources are telnet'ing into the targets. [11, 16 Aug] This indicates the target may be compromised! Sources (internal) appear to have telnet connections to the targets. This is suspicious. Sources should be inspected for tampering. Outgoing telnet AND pop3 (mail). Inspect source for tampering. Incoming connections on port 4407. This port has some correlation associated with Trojan type activity. The target should be inspected. Multiple connections to port 8765. There is a known vulnerability associated with this port – Ultraseek search engine (from Infoseek) buffer overflow. Is the target running this?
159.226.45.3	MY.NET.60.8	
MY.NET.96.168	212.176.58.2	
MY.NET.60.11	159.226.41.166	
MY.NET.163.32	159.226.45.3	
212.179.29.150	MY.NET.53.28	
MULTIPLE	MY.NET.70.33	
159.226.5.152	MY.NET.100.165	Source attempted HTTP (web) connections to target. No other traffic. Several ident (113) connections with no mail traffic. Ident is a simple authentication mechanism usually seen with mail traffic. Ident can also be used to enumerate usernames for information gathering. Sources attempted SHTTP connections to target. SHTTP traffic is encrypted and could be used to tunnel attacks through. Target appears to FTP out, which is followed by several connections to higher ports (32766,68,70). Suspicious.
159.226.63.200	MY.NET.100.230	
212.179.61.247	MY.NET.5.27	
159.226.22.44	MY.NET.253.112	
159.226.114.29	MY.NET.162.199	

### Network Recon (Scans) [Statistics only, ref. EOI by Source for details]

34,626 alerts



Attackers perform network scanning as a form of recon. This information can be used in subsequent attacks. By exploiting weaknesses in protocols (ie. Different flag settings with TCP packets), attackers can glean much information from a network and sometimes avoid IDS and firewalls. The intrusion detection sensor picks up the following recon attempts:

- NMAP TCP Ping - uses the TCP protocol vs. ICMP to determine if a hosts exists [138 alerts]
- Null Scan - uses TCP as well, no flags set [181 alerts]
- NMAP Fingerprint - uses TCP, set all flags, observe response which varies by O.S. [64 alerts]
- Queso Fingerprint - uses TCP, set SYN flag and reserved, observe response [54 alerts]
- SYN-FIN - uses TCP, set SYN and FIN (illegal), observe response [5457 alerts]
- Portscans - if a source attempts a predetermined threshold [7 connections in 2 seconds] of connections, this plug-in raises an alert [28,732 alerts]

### ***Other Notes of Interest***

#### “nbtstat” attempts

These are attempts to connect to the NetBIOS ports of the targets and view a list of shares.

DATE	SOURCE	TARGET
9 Sep	129.37.160.81	MY.NET.100.130
11 Aug	168.167.8.12	MY.NET.253.24
11 Aug	205.229.90.194	MY.NET.181.37
11 Aug	24.28.62.226	MY.NET.70.121
11 Aug	64.7.58.194	MY.NET.20.10
11 Aug	206.171.108.1	MY.NET.6.7
11 Aug	209.150.98.231	MY.NET.130.91
18 Aug	4.17.88.66	MY.NET.6.15
11 Aug	62.136.168.18	MY.NET.70.121
11 Aug	207.79.66.3	MY.NET.253.53
11 Aug	207.79.66.3	MY.NET.253.53
19 Aug	129.37.161.200	MY.NET.100.130
11 Aug	131.118.254.222	MY.NET.6.7
11 Aug	168.143.29.9	MY.NET.60.17
11 Aug	166.72.86.217	MY.NET.100.230
11 Aug	166.72.86.217	MY.NET.100.165

#### “tiny fragments”

Fragmented traffic is commonly associated with attacks. These are normally attacks against TCP/IP stacks themselves or attempts to “fly under the radar” of perimeter security. The following sources were observed sending fragmented traffic.

212.160.15.85  
 213.132.131.201  
 24.68.58.96  
 62.76.42.17

62.76.42.18

### “SMTP source connections”

These hosts attempted connections originating from source port 25, SMTP. This is done as an attempt to penetrate firewall configurations that permit port 25 through. A true internet client/server program would initiate IP communication using a high port, 1024. This activity is suspicious:

DATE	SOURCE	TARGET
17 Aug	206.46.170.21	MY.NET.97.181
10 Sep	156.40.66.2	MY.NET.253.53

### “Happy 99” Worm

It appears the “[Happy 99](#)” worm has been passed on your network.:

DATE	SOURCE	TARGET
16 Aug	128.8.198.101	MY.NET.6.35
20 Aug	24.2.2.66	MY.NET.179.80

### High port to high port connections.

Trojan horses will often use high ports for communication. No other traffic preceded these connections. Therefore they are suspicious.

SOURCE	PORT	TARGET	PORT	DATE
212.179.44.62	30246	MY.NET.15.41	6690	8/17, 00-01
	30945			
212.179.62.74	1984	MY.NET.224.78	6346	9/3, 07:30
212.179.27.6	1948	MY.NET.253.105	26411	9/3, 13:22
212.179.61.5	21263	MY.NET.220.42	2367	9/6, 08:41
		MY.NET.204.150	2667	9/13, 15:09
212.179.47.30	6346	MY.NET.223.62	2995	9/6, 18:47
			1283	9/6, 22:59
212.179.127.45	1063	MY.NET.202.58	6688	9/12, 10:20

### **False Positives**

These hosts were observed passing what was considered normal traffic. For this analysis, they were assumed to have the indicated functions.

MY.NET.253.41, .42, .43	- Mail servers [7,670 Watchlist alerts] *** NOTE: In classifying these as false positives, we assume the IDS is not responsible for <b>virus</b> traffic
NAPSTER TRAFFIC	- the following hosts appear to be running napster (ports 6699, 6700): [3,126 Watchlist alerts] MY.NET. 181.87, 221.94, 205.254, 157.200, 208.178
REALAUDIO MY.NET.101.192	- outgoing on port 7070 [30 alerts] - this appears to be a network management stations using SNMP. The SNMP string is set to the default, “public”. This is a serious security risk and should be changed immediately. [922 alerts]
ICQ TRAFFIC (205.188.153.x)	- ICQ is an internet chat client that defaults to port 4000 and 32771. Port 32771 falls within the range of RPC clients. The “Attempted Sun RPC high port access” is triggered when ports within this range are accessed. RPC services that run on these ports have several vulnerabilities associated with them. However, because these



originate from the icq network (205.188.153), these are considered false positives. [1985 alerts] [Reference [Lenny Zelster](#) practical]

### ***Analyst Observations***

- MY.NET.6, .15, .100 are getting RPC attacks – different sources, same target set
- MY.NET.97, .98, .101 are on the external side of the snort box
- The following hosts have isolated incoming mail traffic (port 25) from watchlist sources. Are these really mail servers?

MY.NET.1.14	5 Sep, 09:55
MY.NET.6.34	7 Sep, 02:42
MY.NET.1.2	7 Sep, 03:11
MY.NET.110.150	8 Sep, 08:15

### **DEFENSIVE RECCOMENDATIONS**

If you have not already done so, establish a network perimeter defense. This can be done with a firewall. The firewall should block all traffic except that which is specifically permitted. Your intrusion detection capability should be placed in front of the firewall. This will give you a situational awareness and let you know when someone is trying something on your network. Minimum recommendations for your policy (firewall ruleset):

- Block incoming TCP port 1080
- Block incoming TCP port 111
- Do not allow incoming FTP (TCP port 21) connections

Viruses have been seen passing into your network. If you do not already have virus software installed, this should be a priority. Virus scanning can be done at the desktop, the gateway, or both.

You should also address the use of what may be inappropriate software on your network. RealAudio, Napster, and ICQ are all in use on the network. These can pose security threats to not only the boxes they are installed on, but the entire network as well. A written or verbal policy should be established with your employees to preclude the use of this software. You may consider implementing security awareness training for your employees to curtail these problems.

### ***Assignment 4 – Analysis Process***

Analysis of bulk data is different than real-time analysis. The analyst can choose to either parse through all the logs as if real-time analysis were being performed. But this can be time consuming with the amount of data we are dealing with. A better approach is to start at a high level and drill down into the data. In this section, I will discuss the strategy I used to perform this analysis.

I borrowed the same approach that is used in the data warehousing community, specifically in OLAP. The challenge in OLAP is dealing with huge amounts of data to find trends and produce more meaningful data. In a data warehouse, dimensions are created. With this data I found four dimensions: source, target, time, and activity. Within each of these dimensions lie further elements, but I start by looking at different combinations of data within these dimensions.

Examples of questions that should be asked:

What source and target combinations produced the most activity?

How were the different types of activity (alarms) distributed over the number of days?

Was activity from a source observed over a range of days or just once day?

Were multiple alarms seen from a source? (ie. Evidence of active targeting?)

In order to answer these questions, we must look at the data as a whole. This can be done a number of ways. But the first step is to parse the logs into dimensions. To do this, I used a Perl script. Perl is ideal for this as that was the scripting language's original intent – the manipulation of text. Here is sample output from my parsed log:

```
09/11/2000,06,SYN-FIN scan!,210.61.144,125,21,MY.NET.6,117,21,2
08/20/2000,12,Watchlist 000222 NET-NCFC,159.226.63,190,1031,MY.NET.253,41,25,4
08/11/2000,02,Watchlist 000222 NET-NCFC,159.226.45,108,1054,MY.NET.60,8,23,45
09/05/2000,20,WinGate 1080 Attempt,207.126.106,118,1391,MY.NET.60,8,1080,1
```

As you can see, the script has produced a comma separated value (CSV) file. A CSV file can be fed into many programs such as Excel and Access. A breakdown of the format

```
08/18/2000,13,Watchlist 000222 NET-NCFC,159.226.63,190,113,MY.NET.253,41,48797,1

08/18/2000,13           date, hour
Watchlist 000222 NET-NCFC      alarm
159.226.63,190,113           source net, source host, source port
MY.NET.253,41,48797          target net, target host, target port
1                             count
```

The last field, count, is a count of all like lines. Since we are losing some fidelity in the logs, there will be many duplicate long entries. Those duplicates are combined into one entry with a total count.

Once this data is imported into Excel or Access, some interesting trends can be found. Two tools I would suggest becoming familiar within Excel are the AutoFilter and the PivotTable. AutoFilter allows you to select specific entries by putting pull downs on the headers. (which by the way will have to be created manually by inserting a row at line 1.) Now you can see all entries for a source network, or all alarms on a given day, etc. This is an efficient way of drilling down within the data.

Access is helpful as well. Attached in Appendix A is a sample Access report. The report gives all alarms for source with their corresponding targets and the dates. This allows the analyst to see active sources. So for instance, if a source has only null scan activity in

one day, it's not considered a high threat. But what if the source is targeting several networks, over the span of several days as was the case with 210.61.144. This trend would be hard to spot by just reviewing the snort logs. Another thing to look for is a source with several different types of activity. This could be evidence of active targeting by employing several different recon methods.

In short, by borrowing some ideas from the OLAP world, the month's worth of logs were easier to analyze. The basics are to break the data into dimensions: time, activity, source, destination. (which by the way should hold true for data from all sources, firewalls, IDS, etc.) Then analyze within those dimensions. This type of analysis can even be more effective than real-time analysis for activity spread over a range of days (such as a slow scan). You lose some fidelity, but you can always refer back to the original logs.

### **Appendix A – EOI by Source [MS Access Report]**

<b>SrcNet</b>	<b>Alarm</b>	<b>DstNet</b>	<b>Date</b>
128.138.14	Null scan!	MY.NET.179	9/13/200
128.153.151	Null scan!	MY.NET.205	9/3/2000
128.194.51	Null scan!	MY.NET.210	9/6/2000
128.211.224	Attempted Sun RPC high port access	MY.NET.98	8/20/200
128.226.152	Null scan!	MY.NET.206	9/8/2000
128.61.105	Null scan!	MY.NET.218	9/8/2000
128.61.59	Queso fingerprint		
128.8.198	Null scan!	MY.NET.217	8/17/200
129.2.146	Happy 99 Virus	MY.NET.6	8/16/200
129.37.160	Queso fingerprint	MY.NET.201	9/14/200
129.37.161	SMB Name Wildcard	MY.NET.100	9/9/2000
	SMB Name Wildcard	MY.NET.100	8/19/200

129.59.24	Null scan!	MY.NET.204	9/3/2000
129.93.214	Null scan!	MY.NET.223	9/6/2000
130.149.41	Null scan! Probable NMAP fingerprint attempt Queso fingerprint SYN-FIN scan!	MY.NET.217	8/17/200
130.239.11	Null scan!	MY.NET.181	8/17/200
130.239.142	Null scan!	MY.NET.223	9/12/200
130.49.220	Null scan!	MY.NET.226	9/12/200
131.118.254	SMB Name Wildcard	MY.NET.6	8/11/200
131.155.192	Null scan!	MY.NET.5	9/5/2000
132.199.220	Null scan!	MY.NET.205	9/5/2000
137.82.136	Null scan!	MY.NET.97	8/15/200
139.91.171	Null scan!	MY.NET.211	9/12/200
141.213.191	Attempted Sun RPC high port access	MY.NET.98	9/12/200
141.223.124	External RPC call	MY.NET.100 MY.NET.6	8/19/200
141.40.205	Null scan!	MY.NET.224	9/6/2000
147.126.59	Queso fingerprint	MY.NET.253	9/6/2000
150.216.127	Null scan!	MY.NET.206	9/11/200
151.196.73	NMAP TCP ping! Null scan!	MY.NET.253	9/8/2000

	Probable NMAP fingerprint attempt		
153.19.25	Null scan!	MY.NET.223	9/10/200
156.40.66	TCP SMTP Source Port traffic	MY.NET.253	9/10/200
161.31.208	External RPC call	MY.NET.6	9/10/200
162.33.184	SMB Name Wildcard	MY.NET.60	8/11/200
166.72.86	SMB Name Wildcard	MY.NET.100	8/11/200
168.143.29	SMB Name Wildcard	MY.NET.60	8/11/200
168.167.8	SMB Name Wildcard	MY.NET.253	8/11/200
18.116.0	External RPC call	MY.NET.6	8/18/200
	SYN-FIN scan!	MY.NET.100	
		MY.NET.15	
		MY.NET.6	
192.55.91	NMAP TCP ping!	MY.NET.5	8/19/200
			8/20/200
193.251.71	Null scan!	MY.NET.146	8/17/200
193.64.205	SUNRPC highport access!	MY.NET.211	9/6/2000
194.237.99	Null scan!	MY.NET.223	9/9/2000
194.94.18	Null scan!	MY.NET.220	9/6/2000
195.132.204	Null scan!	MY.NET.220	9/10/200
195.150.132	Null scan!	MY.NET.202	9/7/2000
2.2.2	NMAP TCP ping!	MY.NET.60	9/2/2000
200.145.151	Null scan!	MY.NET.221	9/3/2000

200.52.201	Null scan!	MY.NET.217	8/11/200
202.187.24	NMAP TCP ping!	MY.NET.1	9/12/200
		MY.NET.179	9/7/2000
		MY.NET.60	9/2/2000
			9/3/2000
205.128.11	NMAP TCP ping!	MY.NET.1	8/11/200
			8/16/200
			8/17/200
			8/18/200
			8/20/200
205.188.153	Attempted Sun RPC high port access	MY.NET.105	8/17/200
			9/8/2000
		MY.NET.206	9/3/2000
	Attempted Sun RPC high port access	MY.NET.217	9/9/2000
		MY.NET.218	9/11/200
		MY.NET.219	9/2/2000
		MY.NET.220	9/7/2000
		MY.NET.222	9/5/2000
			9/6/2000
		MY.NET.53	9/9/2000
205.188.179	Attempted Sun RPC high port access	MY.NET.217	8/15/200
			8/16/200
205.188.4	SUNRPC highport access!	MY.NET.210	9/8/2000
205.229.90	SMB Name Wildcard	MY.NET.181	8/11/200
206.171.108	SMB Name Wildcard	MY.NET.6	8/11/200
206.46.170	TCP SMTP Source Port traffic	MY.NET.97	8/17/200
207.151.147	NMAP TCP ping!	MY.NET.60	8/16/200
	Null scan!		
	Probable NMAP fingerprint attempt		



MY.NET.120

MY.NET.13

MY.NET.130

MY.NET.138

MY.NET.139

MY.NET.140

MY.NET.141

MY.NET.142

MY.NET.143

MY.NET.144

MY.NET.145

MY.NET.146

MY.NET.15

MY.NET.150

MY.NET.151

MY.NET.152

MY.NET.153

MY.NET.154

MY.NET.155

MY.NET.156

MY.NET.157

MY.NET.158

MY.NET.159

MY.NET.160

MY.NET.161

MY.NET.162

MY.NET.163

MY.NET.17

MY.NET.178

MY.NET.179

MY.NET.18

MY.NET.180

MY.NET.181

MY.NET.182

MY.NET.183

MY.NET.184

MY.NET.185

MY.NET.186

MY.NET.188

SYN-FIN scan!

9/11/200



SYN-FIN scan!

9/11/200

MY.NET.190  
MY.NET.198  
MY.NET.199  
MY.NET.2  
MY.NET.20  
MY.NET.200  
MY.NET.201  
MY.NET.202  
MY.NET.203  
MY.NET.204  
MY.NET.205  
MY.NET.206  
MY.NET.207  
MY.NET.208  
MY.NET.209  
MY.NET.21  
MY.NET.210  
MY.NET.211  
MY.NET.212  
MY.NET.213  
MY.NET.214  
MY.NET.215  
MY.NET.216  
MY.NET.217  
MY.NET.218  
MY.NET.219  
MY.NET.220  
MY.NET.221  
MY.NET.222  
MY.NET.223  
MY.NET.224  
MY.NET.225  
MY.NET.226  
MY.NET.227  
MY.NET.228  
MY.NET.229  
MY.NET.230  
MY.NET.231  
MY.NET.232

© SANS Institute 2000 - 2002, Author retains full rights.

		MY.NET.25	
		MY.NET.253	
		MY.NET.254	
		MY.NET.26	
		MY.NET.4	
		MY.NET.5	
		MY.NET.53	
		MY.NET.54	
		MY.NET.6	
		MY.NET.60	
		MY.NET.68	
	SYN-FIN scan!	MY.NET.69	9/11/2000
		MY.NET.7	
		MY.NET.70	
		MY.NET.71	
		MY.NET.75	
		MY.NET.85	
		MY.NET.9	
		MY.NET.94	
		MY.NET.97	
		MY.NET.98	
		MY.NET.99	
211.111.108			
	Null scan!	MY.NET.224	9/10/2000
212.160.15			
	Tiny Fragments - Possible Hostile	MY.NET.160	8/19/2000
212.204.196			
	SUNRPC highport access!	MY.NET.6	9/2/2000
			9/7/2000
212.33.70			
	Null scan!	MY.NET.206	9/3/2000
213.132.131			
	Tiny Fragments - Possible Hostile	MY.NET.203	9/8/2000
213.228.1			
	Queso fingerprint	MY.NET.219	9/14/2000
213.25.136			
	SYN-FIN scan!	MY.NET.1	9/7/2000
		MY.NET.10	
		MY.NET.11	

		MY.NET.12	
		MY.NET.13	
	SYN-FIN scan!	MY.NET.14	9/7/2000
		MY.NET.15	
		MY.NET.17	
		MY.NET.18	
		MY.NET.2	
		MY.NET.20	
		MY.NET.21	
		MY.NET.25	
		MY.NET.26	
		MY.NET.4	
		MY.NET.5	
		MY.NET.53	
		MY.NET.54	
		MY.NET.6	
		MY.NET.60	
		MY.NET.68	
		MY.NET.69	
		MY.NET.7	
		MY.NET.70	
		MY.NET.71	
		MY.NET.75	
		MY.NET.85	
		MY.NET.9	
213.56.48			
	Null scan!	MY.NET.201	9/2/2000
213.6.43			
	Null scan!	MY.NET.208	9/11/200
213.8.52			
	NMAP TCP ping!	MY.NET.60	8/17/200
216.123.60			
	Null scan!	MY.NET.202	9/9/2000
216.123.63			
	Queso fingerprint	MY.NET.75	8/15/200
216.15.191			
	Queso fingerprint	MY.NET.253	9/9/2000
			9/11/200
			9/12/200

			9/13/200
		MY.NET.6	9/9/2000
			9/11/200
216.161.190	Null scan!	MY.NET.226	9/11/200
216.164.133	SMB Name Wildcard	MY.NET.60	8/11/200
216.181.188	Probable NMAP fingerprint attempt	MY.NET.6	8/15/200
216.63.200	Null scan!	MY.NET.203	9/5/2000
24.112.241	Null scan!	MY.NET.201	8/20/200
24.113.80	Null scan!	MY.NET.203	9/6/2000
24.115.96	Null scan!	MY.NET.222	9/9/2000
24.13.104	Null scan!	MY.NET.253	8/15/200
24.160.189	Null scan!	MY.NET.220	9/6/2000
24.164.181	Null scan!	MY.NET.217	8/18/200
24.17.189	Possible wu-ftpd exploit -	MY.NET.150	9/8/2000
		MY.NET.202	
	site exec - Possible wu-ftpd exploit -	MY.NET.150	
		MY.NET.202	
		MY.NET.99	
24.180.134	NMAP TCP ping!	MY.NET.208	9/11/200
	Null scan!		
	Probable NMAP fingerprint attempt		
24.180.196	Null scan!	MY.NET.217	9/6/2000
24.19.101	Null scan!	MY.NET.222	9/5/2000
24.19.244	Queso fingerprint	MY.NET.162	9/3/2000

24.2.2	Happy 99 Virus	MY.NET.179	8/20/200
24.200.201	Null scan!	MY.NET.162	8/16/200
24.201.116	Null scan!	MY.NET.209	9/12/200
24.201.209	SYN-FIN scan!	MY.NET.202	9/2/2000
24.22.125	Null scan!	MY.NET.223	9/9/2000
24.23.198	Null scan!	MY.NET.217	8/17/200
			8/18/200
	Probable NMAP fingerprint attempt		
24.232.79	Null scan!	MY.NET.206	9/6/2000
24.24.137	Queso fingerprint	MY.NET.219	9/3/2000
24.28.33	Null scan!	MY.NET.224	9/13/200
24.28.62	SMB Name Wildcard	MY.NET.70	8/11/200
24.29.7	Null scan!	MY.NET.218	9/9/2000
24.3.161	Queso fingerprint	MY.NET.145	9/5/2000
			9/12/200
			9/13/200
24.6.140	Null scan!	MY.NET.130	9/9/2000
24.68.58	Tiny Fragments - Possible Hostile	MY.NET.210	9/13/200
		MY.NET.217	9/11/200
24.72.23	Null scan!	MY.NET.213	9/6/2000
24.72.8	Null scan!	MY.NET.213	9/6/2000
24.8.241	Null scan!	MY.NET.70	8/19/200

24.91.58	Null scan!	MY.NET.221	9/14/200
24.92.174	Null scan!	MY.NET.217	9/11/200
24.92.188	Null scan!	MY.NET.106	9/14/200
4.17.88	SMB Name Wildcard	MY.NET.6	8/18/200
62.10.136	Null scan!	MY.NET.212	9/13/200
62.136.168	SMB Name Wildcard	MY.NET.70	8/11/200
62.2.64	Null scan!	MY.NET.218	9/9/2000
62.76.42	Tiny Fragments - Possible Hostile	MY.NET.1 MY.NET.212	9/14/200
63.144.227	Null scan!	MY.NET.208	9/8/2000
63.226.208	NMAP TCP ping! Null scan! Probable NMAP fingerprint attempt	MY.NET.253	9/2/2000
64.7.58	SMB Name Wildcard	MY.NET.20	8/11/200
64.80.63	Queso fingerprint	MY.NET.201 MY.NET.204 MY.NET.208 MY.NET.209 MY.NET.210 MY.NET.217	9/7/2000 9/8/2000 9/7/2000 9/11/200 9/7/2000 9/8/2000
	Queso fingerprint	MY.NET.223 MY.NET.224	9/10/200 9/10/200 9/11/200
MY.NET.101	SMB Name Wildcard	MY.NET.101	8/15/200

			8/16/200
			8/17/200
			8/19/200
			8/20/200
			9/2/2000
			9/3/2000
			9/7/2000
			9/9/2000
			9/10/200
			9/11/200
			9/12/200
			9/13/200
MY.NET.97	SNMP public access	MY.NET.101	8/15/200
			8/16/200
			8/20/200
			9/9/2000
			9/11/200
MY.NET.98	SNMP public access	MY.NET.101	8/17/200
			8/19/200
			8/20/200
			9/2/2000
			9/3/2000
			9/7/2000
			9/10/200
	SNMP public access	MY.NET.101	9/12/200
			9/13/200

© SANS Institute 2000 - 2002, Author retains full rights.

# Upcoming Training

Click Here to  
**{Get CERTIFIED!}**



Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC503: Intrusion Detection In-Depth	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201709,	Sep 11, 2017 - Oct 18, 2017	vLive
Baltimore September 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Scottsdale SEC503	Scottsdale, AZ	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Boston SEC503	Boston, MA	Oct 09, 2017 - Oct 14, 2017	Community SANS
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced