Global Information Assurance Certification Paper

GIAC Intrusion Detection Curriculum
Practical Assignment
GCIA

Thierry Engels

February 2001

Assignment 1 : Network Detects

Network detect #1

```
4 23:59:53 curry snort[17326]: spp_portscan: PORTSCAN DETECTED from 209.223.45.87 (STEALTH)
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.2:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.3:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.4:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.5:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.11:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.13:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.14:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.15:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.18:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.16:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.17:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.19:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.21:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.20:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.24:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.26:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.28:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.29:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.30:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.31:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.32:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.34:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.33:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.36:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.37:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.38:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.39:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.41:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.42:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.43:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.44:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.45:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.46:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.48:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.49:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.50:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.51:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.52:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.66:53
4 23:59:53 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.67:53
4 23:59:55 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.201:53
4 23:59:55 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.200:53
4 23:59:55 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.202:53
4 23:59:55 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.204:53
4 23:59:55 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.203:53
4 23:59:55 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.205:53
4 23:59:55 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.206:53
4 23:59:55 curry snort[17326]: IDS198/SYN FIN Scan: 209.223.45.87:53 -> MY.NET.6.207:53
4 23:59:57 curry snort[17326]: IDS277/named-probe-iquery: 209.223.45.87:3170 -> MY.NET.6.4:53
4 23:59:57 curry snort[17326]: IDS277/named-probe-iquery: 209.223.45.87:3171 -> MY.NET.6.2:53
4 23:59:57 curry snort[17326]: NRB026/UDP DNS version asked through chaos: 209.223.45.87:3171 ->
MY.NET.6.2:53
4 23:59:57 curry snort[17326]: NRB026/UDP DNS version asked through chaos: 209.223.45.87:3170 ->
MY.NET.6.4:53
5 00:00:04 curry snort[17326]: spp_portscan: portscan status from 209.223.45.87: 52 connections across 48
hosts: TCP(50), UDP(2) STEALTH
5 00:00:15 curry snort[17326]: spp_portscan: End of portscan from 209.223.45.87: TOTAL time(4s) hosts(48)
TCP(50) UDP(2) STEALTH
```

1. Source of Trace:

This trace is coming from my home network.

2. Detect was generated by:

Snort intrusion detection system V. 1.6.3 with a specific rule :

```
alert TCP $EXTERNAL any -> $INTERNAL 53 (msg: "NRB026/UDP DNS version
asked trough chaos"; content: "version|04|bind"; nocase; )
```

Log format is :

```
<Date> <time> <hostname> <process>[<pid>]: <detect id>: <src-ip>:<port> ->
<dest-ip>:<port>
```

3. Probability the source address was spoofed:

Very low, as the attacker need to know which version of DNS we are running in order to organize the next phase.

4. Description of attack:

This is a reconnaissance scanning trying to fetch the version of BIND server we are running. This kind of probe is used as a pre attack reconnaissance in order to locate specific version of BIND server. Various attack are available on various version of BIND. Please see www.isc.org, www.cert.org and www.auscert.org for advisories).

5. Attack mechanism:

A sepcial zone called "bind" in the chaosnet class is used to derive some info about the BIND version running. This zone should be "secured" in order to block queries from "unsecured" networks. In the example below, the zone has been totally blocked (best to my point of view) :

```
zone "bind" chaos
 {
  type master;
  file "config/bind.db";
  allow-query { none; };
  allow-transfer { none; };
```

```
};
```

6. Correlations:

The detect is a well known recon activity. www.whitehats.com are publishing a snort rule which detects the same kind of activity. [IDS278]. See also [IDS277].

7. Evidence of active targeting:

Previous portscan where done on the whole subnet in order to find BIND servers (SYN-FIN scan). Then specific info where collected on those BIND servers: support of the Inverse Query and the BIND version number.

8. Severity:

(Critical + Lethal) – (System + Network Countermeasures) = Severity
(5 + 2) – (4 + 2) = 1

Critical: DNS is the basement of internet communication(5)
Lethal: attack is a reconnaissance scan (2)
System coutermesure: target is state-of-the-art secured(4)
Network coutermesure: traffic is allowed to target(2)

9. Defensive recommendation:

1. Latest version and patch of BIND installed
2. Block version number avilability
3. Hide DNS whith real authtoritative behind a firewall with no external exposure. Exposed DNS is just a slave. Will decrease the impact of an attack.

10. Multiple choice test question:

Why is BIND server a well known target of attacks ?

a) BIND is unable to handle huge amount of queries and will easily crash,
b) BIND is a public domain software with a lot of bugs, so its easy to hack,
c) BIND is the basement of the Internet communication, so the impact of an attack is critical.

Answer: C

Network detect #2

```
…
12:58:54 curry snort[446]: NRB027/TCP AXFR Transfert asked: SUSPECT.NET.208.139:52048 -> MY.NET.6.2:53
12:59:30 curry snort[446]: NRB027/TCP AXFR Transfert asked: SUSPECT.NET.208.139:52054 -> MY.NET.6.2:53
12:59:31 curry snort[446]: NRB027/TCP AXFR Transfert asked: SUSPECT.NET.208.139:52056 -> MY.NET.6.2:53
12:59:32 curry snort[446]: NRB027/TCP AXFR Transfert asked: SUSPECT.NET.208.139:52057 -> MY.NET.6.2:53
12:59:32 curry snort[446]: NRB027/TCP AXFR Transfert asked: SUSPECT.NET.208.139:52059 -> MY.NET.6.2:53
12:59:33 curry snort[446]: NRB027/TCP AXFR Transfert asked: SUSPECT.NET.208.139:52060 -> MY.NET.6.2:53
12:59:34 curry snort[446]: NRB027/TCP AXFR Transfert asked: SUSPECT.NET.208.139:52063 -> MY.NET.6.2:53
13:01:44 curry snort[446]: NRB027/TCP AXFR Transfert asked: SUSPECT.NET.208.139:52082 -> MY.NET.6.2:53
…

…
12:58:54.998753 P SUSPECT.NET.52048 > MY.NET.domain: P 667026896:667026923(27) ack 1645999448 win 8760 (DF)
12:59:30.806201 P SUSPECT.NET.52054 > MY.NET.domain: P 672056931:672056958(27) ack 1683674515 win 8760 (DF)
14:14:48.475509 P SUSPECT.NET.52534 > MY.NET.domain: P 1279398518:1279398544(26) ack 2153390574 win 8760 (DF)
14:14:49.606303 P SUSPECT.NET.52539 > MY.NET.domain: P 1279932940:1279932980(40) ack 2148740369 win 8760 (DF)
…

…
12:58:55.430 notify: info: Received NOTIFY answer (AA) from SUSPECT.NET.208.139 for "MY.ZONE IN SOA"
12:59:30.430 notify: info: Received NOTIFY answer (AA) from SUSPECT.NET.208.139 for "MYOTHERZONE IN SOA"
14:14:52.418 notify: info: Received NOTIFY answer (AA) from SUSPECT.NET.208.139 for "YETTANOTHERZONE IN SOA"
14:14:52.421 notify: info: Received NOTIFY answer (AA) from SUSPECT.NET.208.139 for "ONEMOREZONE IN SOA"
…
```

1. Source of Trace:

This trace is coming from my home network.

2. Detect was generated by:

Snort intrusion detection system V. 1.6.3 with a specific rule for the first part

```
alert TCP $EXTERNAL any -> $INTERNAL 53 (msg: "NRB027/TCP AXFR Transfert
asked"; content: "|00 00 FC 00 01|"; offset: 14; )
```

Log format is :

```
<time> <hostname> <process>[<pid>]: <detect id>: <src-ip>:<port> -> <dest-
ip>:<port>
```

Tcpdump for the second trace

Log format is :

```
<time> <src-ip>.<src-port> > <des-ip>.<dest-port>: <flags> <data-seqno>
ack  <ack> win <window> <urgent> <options>
```

BIND AXFR log for the latter
Log format is:

<time> notify: <priority>: Received NOTIFY answer (AA) from <src-ip> for <domain>

3. Probability the source address was spoofed:

The probability is very low because the three way handshake has been done.

4. Description of attack:

First of all, we are allowing AXFR to our known secondary DNS only. This trace show AXFR requests (and answers :-{{) with an unknown system. This has been flagged as suspect. We can't really call this an attack (at this time), but just some information collecting process which must catch our attention. The point is that we shouldn't allow unknown system to request our zone files.

If we look closer at the BIND log file, we can see that the AXFR has been triggered by our DNS (NOTIFY ANSWER). This is becoming really strange as SUSPECT.NET is not (as far as I know) a friend DNS.

I've immediately called the DNS manager at my site, and things become clearer : our upstream provider was merging with another one (SUSPECT.NET) becoming a bigger one (did you ever heard this kind of story ?) and the DNS manager were requested to register a new DNS as secondary.

5. Attack mechanism:

This was considered as information gathering at first, using zone tranfer requests. Now we know that this is normal traffic (secondary updating their cache of authoritative info).

6. Correlations:

None

7. Evidence of active targeting:

This traffic is directed to our primary DNS server only. So one can call this "targeted" traffic.

8. Severity:

(Critical + Lethal) – (System + Network Countermeasures) = Severity
(5 + 2) – (4 + 2) = 1

Critical: DNS is the basement of internet communication(5)
Lethal: attack is an information gathering process (2)
System coutermesure: target is state-of-the-art secured and should block unwanted AXFR request (4)
Network coutermesure: traffic is allowed to target(2)

9. Defensive recommendation:

1. Latest version and patch of BIND installed
2. Reply to AXFR query only to secondary servers
3. Hide DNS whith real authtoritative behind a firewall with no external exposure. Exposed DNS is just a slave. Will decrease the impact of an attack.

10. Multiple choice test question:

What is the best way to correlate logs files entries ?

a) Having all the log merged into one file, searching trough it
b) Having systems time synchro and searching trough their logs,
c) Using an expert system to do correlation trough the logs

Answer: b

Network detect #3

```
…
Jan 21 10:30:36 curry snort[17326]: IDS266/smtp-chameleon-overflow: 63.209.80.226:63307 -> MY.NET.6.28:25
Jan 22 19:57:35 curry snort[17326]: IDS266/smtp-chameleon-overflow: 63.209.80.226:63371 -> MY.NET.6.28:25
Jan 22 23:03:11 curry snort[17326]: IDS266/smtp-chameleon-overflow: 63.209.80.226:64353 -> MY.NET.6.28:25
Jan 23 19:30:13 curry snort[17326]: IDS266/smtp-chameleon-overflow: 63.209.80.226:61617 -> MY.NET.6.28:25
Jan 24 09:04:00 curry snort[17326]: IDS266/smtp-chameleon-overflow: 63.209.80.226:62594 -> MY.NET.6.28:25
Jan 26 06:29:59 curry snort[17326]: IDS266/smtp-chameleon-overflow: 63.209.80.226:64185 -> MY.NET.6.28:25
Jan 26 21:14:26 curry snort[17326]: IDS266/smtp-chameleon-overflow: 63.209.80.226:64758 -> MY.NET.6.28:25
Jan 27 02:14:24 curry snort[17326]: IDS266/smtp-chameleon-overflow: 63.209.80.226:64182 -> MY.NET.6.28:25
Jan 27 23:38:45 curry snort[17326]: IDS266/smtp-chameleon-overflow: 63.209.80.226:61550 -> MY.NET.6.28:25
Jan 27 23:42:51 curry snort[17326]: IDS266/smtp-chameleon-overflow: 63.209.80.226:62017 -> MY.NET.6.28:25
…
```

1. Source of Trace.

This trace is coming from my home network.

2. Detect was generated by:

Snort intrusion detection system V. 1.6.3 with a specific rule for the first part

```
alert TCP $EXTERNAL any -> $INTERNAL 25 (msg: "IDS266/smtp-chameleon-
overflow"; content: "HELP"; nocase; flags: AP; dsize: >500; depth: 10;)
```

Log format is :

```
<time> <hostname> <process>[<pid>]: <detect id>: <src-ip>:<port> -> <dest-
ip>:<port>
```

3. Probability the source address was spoofed:

The probability is very low because the three way handshake must be done in order to send the ftp HELP command.

4. Description of attack:

The chameleon SMTP deamon (NetManage) is vulnerable to some buffer overflow attacks. One of them is trough the HELP command. In this case, the attacker is sending a help command with a payload size greater than 500 bytes. This should crash the deamon. See http://www.insecure.org/sploits/netmanage.chameleon.overflows.html for more details about Chameleon holes.

5. Attack mechanism:

Connect to smtp server (telnet chameleon.smtp.server:25), issue the help command with more thant 500 bytes of data behind it :

```
%> telnet chameleon.smtp.server 25
220 chameleon.smtp.server  SMTP server (Chameleon)
HELP
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
(crash)
```

6. Correlations:

This issue has been covered in http://www.insecure.org/sploits/netmanage.chameleon.overflows.html It seems that a lot of SMTP are still vulnerable to this kind of hack regarding the amount of alerts triggered by SNORT about SMTP HELP commands with more thant 500 bytes of data.

7. Evidence of active targeting:

This traffic is directed to our SMTP servers, but those are not chameleon servers. So one can think of attack targeting SMTP servers only. I’ve not find any previous portscan activity with this source address.

8. Severity:

(Critical + Lethal) - (System + Network Countermeasures) = Severity
(4 + 5) - (5 + 1) = 3

Critical: The mail services is critical to our business(4)
Lethal: the attack will kill the SMTP daemon(5)
System coutermesure: SMTP HELP commands are disabled, our daemon is not subject to buffer overflow with the HELP command
Network coutermesure: traffic is allowed to target(1)

9. Defensive recommendation:

1. Do not run chameleon SMTP services. They are known to be buggy.
2. Do not allow HELP command on your SMTP services. SMTP servers know their jobs.
3. Run the IDS266 on your IDS just to flag networks sources in your black list

10. Multiple choice test question:

Why is it best to disable unused functionalities in exposed software ?

a) Because it consume memory,
b) Because every byte of code hide a bit of bugs.
c) Because it slow down the processing.

Answer: b

Network detect #4

```
… First, there is the three way handshake
21:43:15.026593 eth0 P MY.NET.16.100.1153 > MY.NET.16.201.tnet: S 30934592:30934592(0)
                                           win 16384 <mss 1460,nop,nop,sackOK> (DF)
21:43:15.029576 eth0 P MY.NET.16.201.tnet > MY.NET.16.100.1153: S 3920818296:3920818296(0)
                                           ack 30934593 win 4288 <mss 1460>
21:43:15.029693 eth0 P MY.NET.16.100.1153 > MY.NET.16.201.tnet: . 30934593:30934593(0)
                                           ack 3920818297 win 17520 (DF)

… Then the normal telnet session is running
21:43:15.035643 eth0 P MY.NET.16.201.tnet > MY.NET.16.100.1153: P 3920818297:3920818309(12)
                                           ack 30934593 win 4288
21:43:15.042312 eth0 P MY.NET.16.100.1153 > MY.NET.16.201.tnet: P 30934593:30934596(3)
                                           ack 3920818309 win 17508 (DF)

… (I've removed some packet just to keep you awake …)
21:43:24.852851 eth0 P MY.NET.16.201.tnet > MY.NET.16.100.1153: P 3920818411:3920818412(1)
                                           ack 30934655 win 4226
21:43:24.946050 eth0 P MY.NET.16.100.1153 > MY.NET.16.201.tnet: P 30934655:30934657(2)
                                           ack 3920818412 win 17405 (DF)
21:43:24.951239 eth0 P MY.NET.16.201.tnet > MY.NET.16.100.1153: P 3920818412:3920818414(2)
                                           ack 30934657 win 4224
21:43:25.055017 eth0 P MY.NET.16.201.tnet > MY.NET.16.100.1153: P 3920818414:3920818635(221)
                                           ack 30934657 win 4224
21:43:25.055150 eth0 P MY.NET.16.100.1153 > MY.NET.16.201.tnet: . 30934657:30934657(0)
                                           ack 3920818635 win 17182 (DF)
21:43:36.854037 eth0 P MY.NET.16.100.1153 > MY.NET.16.201.tnet: P 30934657:30934660(3)
                                           ack 3920818635 win 17182 (DF)

… Now, the strange activity is coming
21:43:36.854493 eth0 P MY.NET.16.201.tnet > MY.NET.16.100.1153: R 3920818635:3920818635(0)
                                           win 1024 [tos 0xf4]
21:43:36.861558 eth0 P MY.NET.16.201.tnet > MY.NET.16.100.1153: P 3920818635:3920818646(11)
                                           ack 30934660 win 4221
21:43:36.861714 eth0 P MY.NET.16.100.1153 > MY.NET.16.201.tnet: R 30934660:30934660(0)
                                           win 0
21:43:36.872706 eth0 P MY.NET.16.201.tnet > MY.NET.16.100.1153: R 3920818635:3920818635(0)
                                           win 1024 [tos 0xf4]
21:43:36.890574 eth0 P MY.NET.16.201.tnet > MY.NET.16.100.1153: R 3920818635:3920818635(0)
                                           win 1024 [tos 0xf4]
… (there are a lot of RESET packet sent)
```

1. Source of Trace.

This trace is coming from my lab's network.

2. Detect was generated by:

Tcpdump with the folowing format :

```
<time> <if> <promisc> <src-ip>.<src-port> > <des-ip>.<dest-port>: <flags>
<data-seqno> ack  <ack> win <window> <urgent> <options>
```

3. Probability the source address was spoofed:

The attacker doesn't need any answer and he doesn't want to be traced back, so : YES the source address is spoofed.

4. Description of attack:

This attack is used to shut down some TCP connections. This trace above doesn't show a huge amount of RESET packet sent from the (spoofed) source MY.NET.16.201.tnet.

5. Attack mechanism:

The attacket code is learning the sequence number of the TCP session and craft some RESET packet on the fly.

6. Correlations:

This hack has been played using a program called COUIC [Cutting Off Unwanted Ip Connections]. Thanks to Michel Arboi for this "academic" tool [http://michel.arboi.free.fr/couic.html], which can really make your network becoming crasy. It demonstrate how to shut down TCP connections on the fly. COUIC is also able to disturb UDP traffic by sending ICMP "Host unreachable / bad port".One should know that COUIC is really noisy.

7. Evidence of active targeting:

As this was ran in a test lab, it was really targeted. COUIC is able to shutdown TCP session either globally (it will shutdown every TCP session he will heard about) or sepcifically (you can tell COUIC wich session to shutdown, using the command line params).

8. Severity:

(Critical + Lethal) – (System + Network Countermeasures) = Severity
(4 + 1) – (0 + 0) = 5

Critical: COUIC can really disturb your network availability (4)

Lethal: COUIC will not destroy any system or data (1)
System coutermesure: hard to filter RESET packets (0)
Network coutermesure: hard to filter RESET packets (0)

9. Defensive recommendation:

Think of blocking RESET traffic coming, but will have some side effects too.

10. Multiple choice test question:

In a lan, spoofed ip packets can be differenciated from legitimate ones based on :

a) The physical network frame checksums
b) The physical network sequence numbers
c) The physical network addresses (MAC)

Answer c

Assignment 2 : "Analyze This" Scenario

Introduction

GIAC enterprises provides us with some data from one (of their) snort system running on MY.NET network. Unfortunately they didn't gives us the rule base used at the time of collect. As they have requested us to focus on the analysis of those files, many spects of security analysis will not be covered in this document [DR01].

As mentionned by GIAC enterprises, not all the data are available due to some power failures and disk space availability.This can reduce the ability to make corelations between scans, attacks and exploits. Further more this can lead to undetected suspicious activity. Wether or not the disk space un-availability is the result of some "pre" attack activity is a question left open [DR02].

In the report some references will be done to the "Defensives recommendations" that can be outlined out of this analysis. Those references will be marked as [DRxx].

Overview of the data set

The dataset is composed of 3 set of text (ASCII) files (mainly) coming from a snort system. (snort alerts logs, snort scans reports and some Out-Of-Spec files (illegal per RFC packet dumps)).

File format can be described using the folowing (simple) grammar (please have a look at the unix `date` man page for more info about date and time format description). Optionnal items are enclosed in "[[" "]]" and multiple occurence of optionnal items is noted as "[[" "]]*". Variables are prefixed with "$".

Set 1 : SnortA*.txt

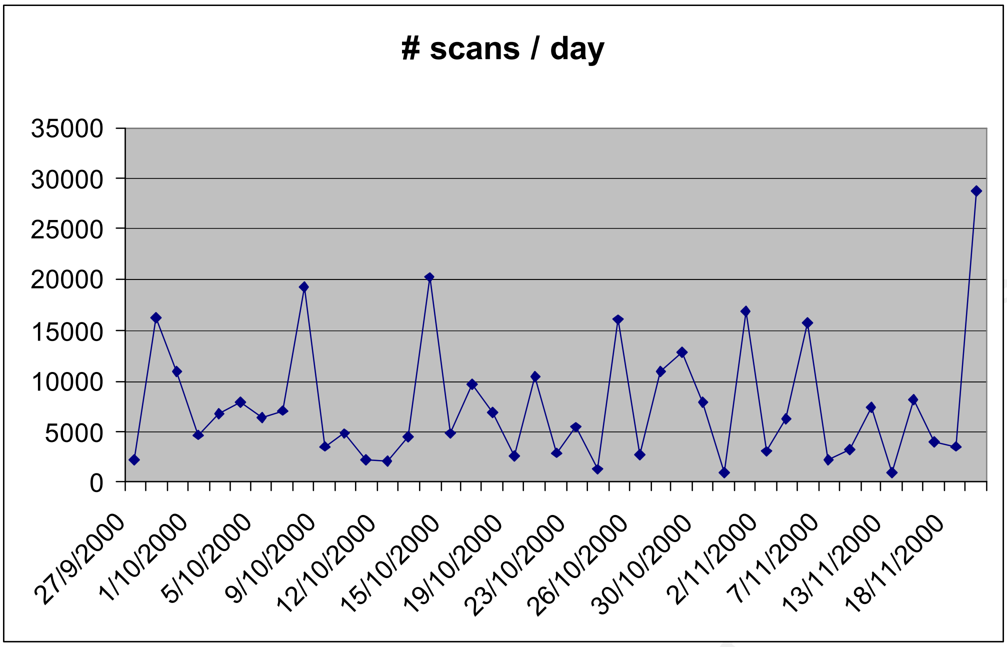
```
%m/%d-%k:%M:%S.$msec [**] $Alert-identification [**] [[ $src-ip:$src-port -
> $dest-ip:$dest-port ]]
```

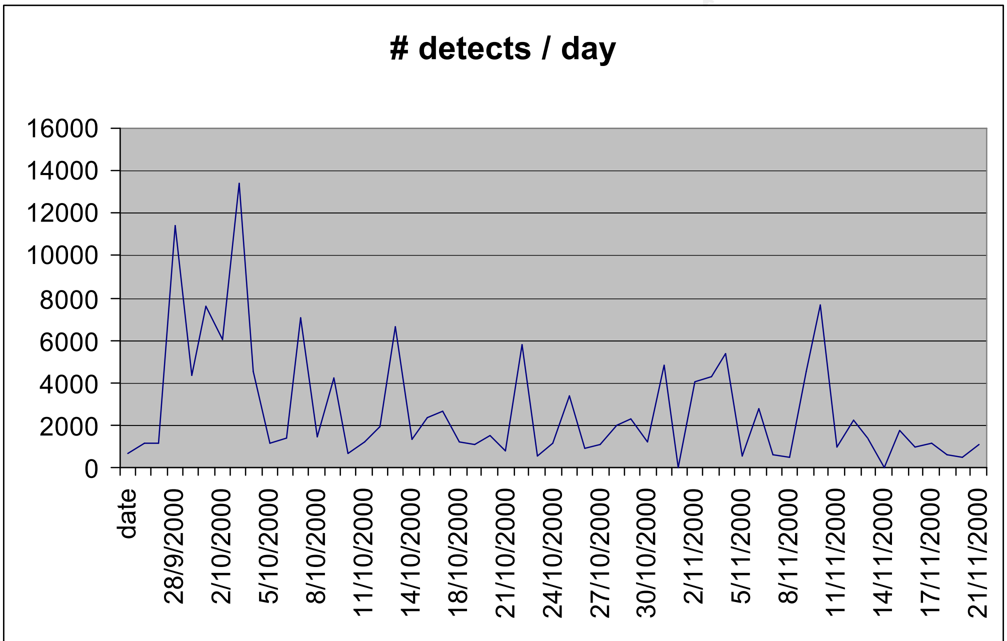
Set 2 : SnortS*.txt

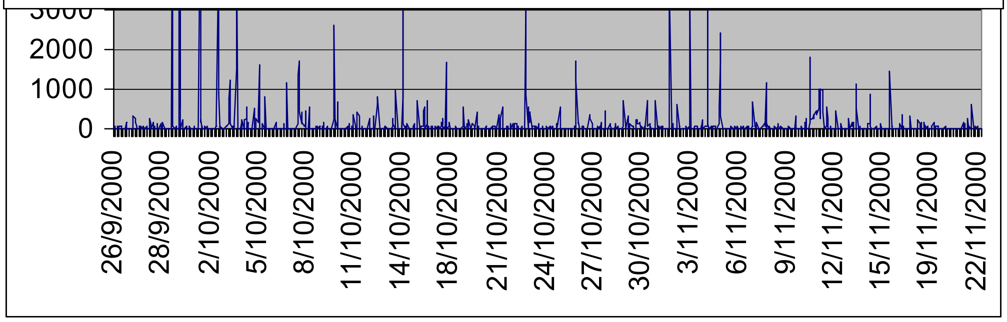
```
%b %d %k:%M:%S $src-ip:$src-port -> $dest-ip:$dest-port [[$flags]]*
```

Set 3 : Oos*.txt

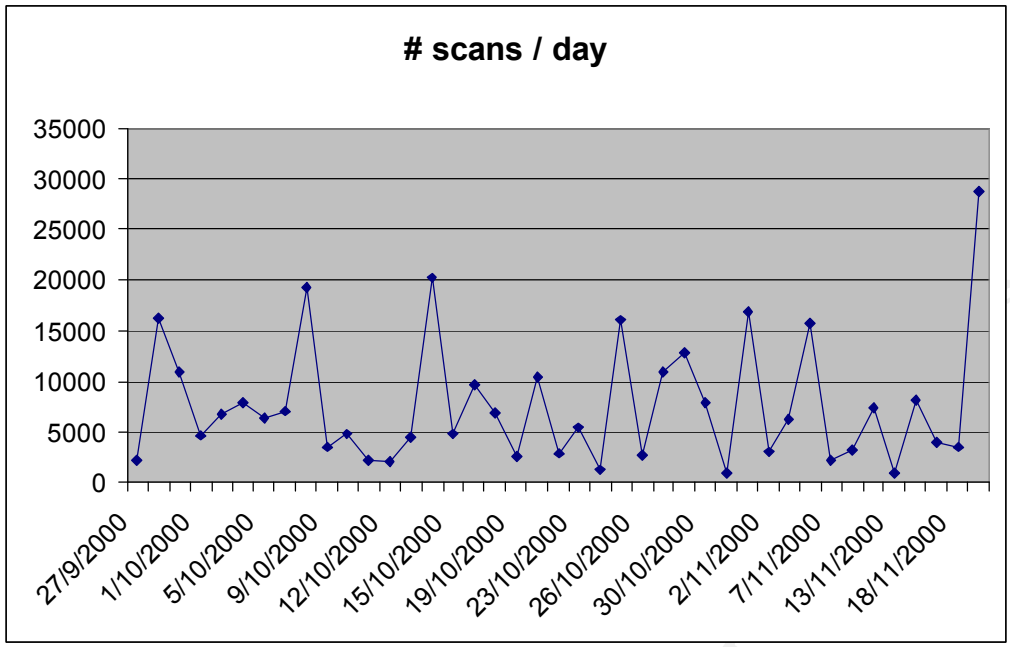
```
%m/%d-%k:%M:%S.$msec $src-ip:$src-port -> $dst-ip:$dest-port
$protocol TTL:$time-to-live TOS:$type-of-service ID:$id [[$Don't-fragment-
flag]]
$tcp-flags Seq: $tcp-seqence-number Ack: $tcp-acknowledgment-number Win:
```



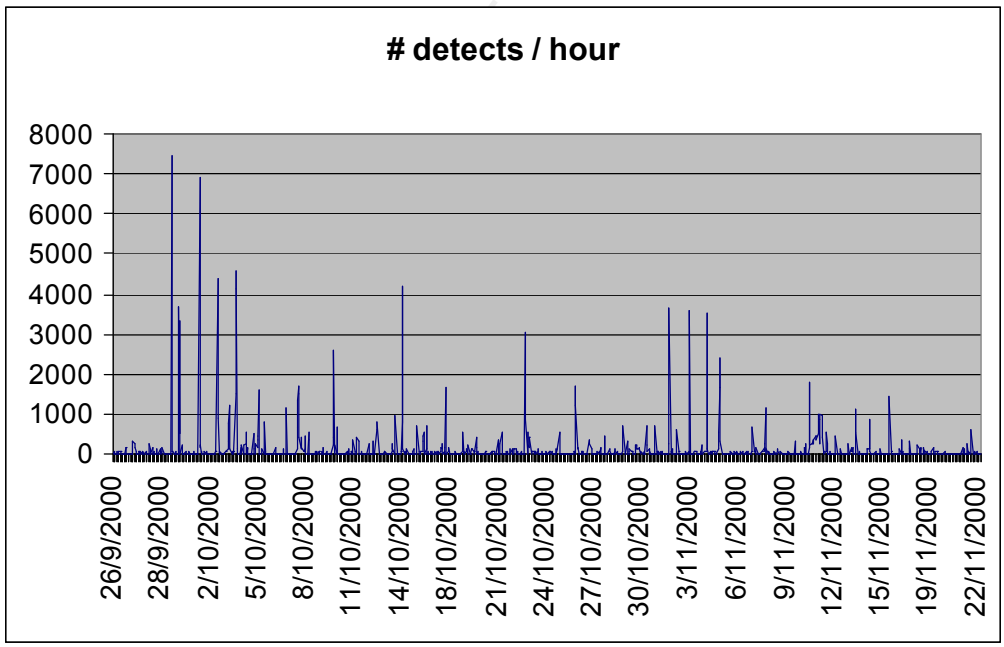


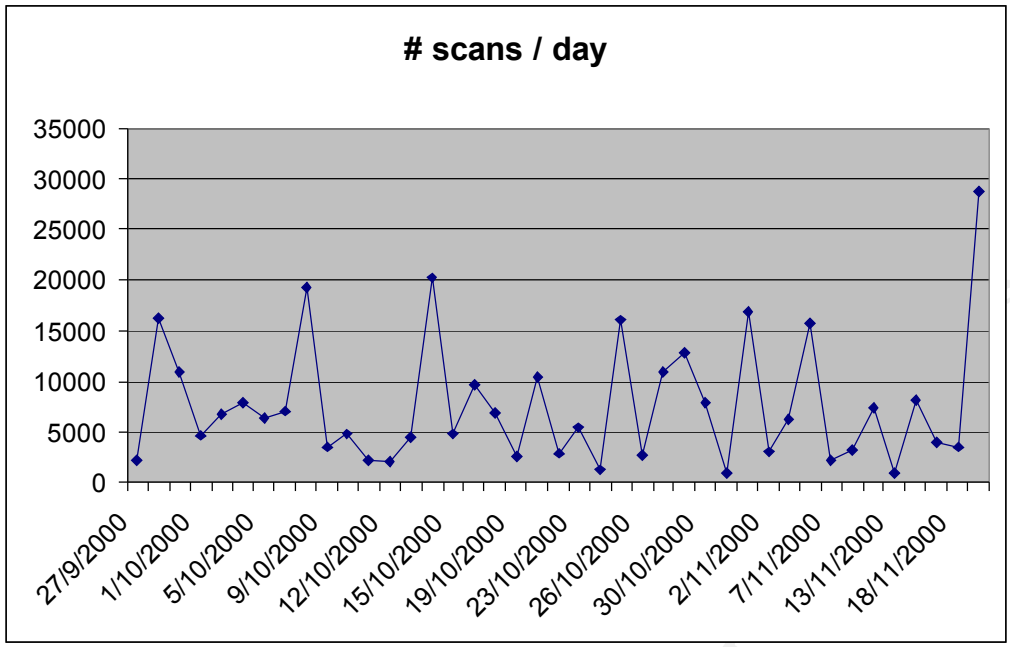


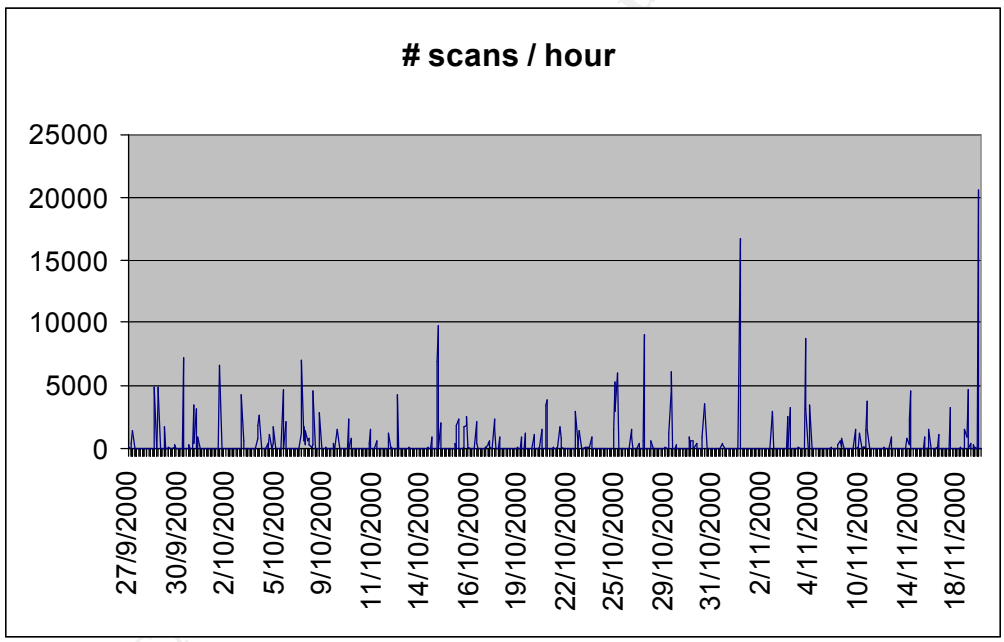

```
$tcp-window-size
[[ TCP Options => $tcp-options ]]
[[ $data ]]
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```



The dataset covers the period of 2000-sep-26 to 2000-nov-23 with some specific OOS data from 2000-aug-17. Distribution of the dataset can be graphed like this :







One can already see some hard time (7460 alerts/hour on 2000-sep-30 13:00 and 20652 scan hits on 2000-nov-23 19:00). The fairly high amount of suspicious traffic logged already drives us to a very first conclusion : The Giac Enterprises network seems to have some success (at least from the hacker point of view) and need a continuous security vigilance

The top talkers list

Here is the top-ten talker list based on ip sources :

#detects	Src-ip	Detect
7199	160.78.49.191(PARMANET)	SYN-FIN scan! only
6635	208.61.4.207(BELLSNET-BLK7)	SYN-FIN scan! only
6297	159.226.45.3(NCFC)	Watchlist 000222 NET-NCFC only
6117	212.179.95.5(CABLE-XPRMNT)	Watchlist 000220 IL-ISDNNET-990517 only
4967	209.92.40.32(DSL1-FASTNET)	SYN-FIN scan! only
4011	212.179.27.6(ADI-ASSOCIATION)	Watchlist 000220 IL-ISDNNET-990517 only
3950	212.179.79.2(CREOSCITEX)	Watchlist 000220 IL-ISDNNET-990517 only
3938	212.179.44.115(GIVAT-BRENER)	Watchlist 000220 IL-ISDNNET-990517 only
3897	63.195.56.20(SBCIS39515)	SYN-FIN scan! only
3860	130.89.229.48(UTNET)	SYN-FIN scan! only

Top ten talkers list shows a large amount of recon activity, which must be considered as premisses of an attack [DR08]. Top talkers also shows that IL-ISDNNET and NET-NCFC are quite active. Further analysis of the 2 whatchlist activity should be done. If we don't take into account the recon activity and the watchlist, here is the “hot top-ten" list of detects by IP sources :

#detects	Src-ip	Detect
1883	63.193.210.208(SBCIS990913-39)	WinGate 1080 Attempt only
1792	211.46.110.81(KRNIC-KR-23)	1789 SMTP Source Port traffic, 3 RPC attemps
1097	24.7.227.215(BB1-RDC2-TX-2@Home Network)	1906 SMTP Source Port traffic, 1 External RPC call
628	205.188.153.108(AOL-DTC)	Attempted Sun RPC high port access only
517	205.188.153.107(AOL-DTC)	Attempted Sun RPC high port access only
435	205.188.153.116(AOL-DTC)	Attempted Sun RPC high port access only
334	205.188.153.109(AOL-DTC)	Attempted Sun RPC high port access only
306	62.136.90.120(POL-CAG1)	Back Orifice only
291	63.46.46.143(NETBLK-UUNET97DU)	Back Orifice only
222	208.194.161.155(UU-208-194-160)	WinGate 1080 Attempt only
179	198.63.2.192(VRIO-198-063)	WinGate 1080 Attempt only
157	204.117.70.5(SPRINT-BLKB)	WinGate 1080 Attempt only
137	64.86.5.250(9NETAVE-1)	WinGate 1080 Attempt only
132	207.114.4.46(ABSNET-BLK1)	WinGate 1080 Attempt only

We already sees here a lot of SMTP traffic coming from KOREA. This can be seen as a real concern if you don't usually have mail exchange with korean. Please see [DR09]

Why not sorting detects by netblock id ? Here is the top ten NETBLOCK being detected :

#detects	netblock
2548	AOL-DTC, AOL, US
1883	SBCIS990913-39, ADSL BASIC-rback7-snfc21, US

1793	KRNIC, from a block assigned by the “Korea Network Information Center, KR”
1097	BB1-RDC2-TX-2, @Home Network, US
523	NETBLK-UUNET97DU, UUNet Technologies, US
354	POL-CAG1, CAG Block 1 Planet Online Limited, UK
302	FX-NET, FX Internet - One Trading Group Burebista 1, bl. D15, sc. 3 Bucuresti 3, RO
275	IXIR, Iksir Uluslararasi Elektronik Ticaret Bilgil endirme ve Haberlesme Hizmetleri A. S., TR
244	MOBIFON, MobiFon S.A. 3, Nerva Traian Street Complex M101, Sector 3 Bucharest, RO
241	UU-208-194-160, First Internet Alliance, US

As one may guess, dialup ISP are used by hackers. This gives them the ability to do some reconnaissance with one ISP and drives the attack with another one.

A strange source address where found : 2.2.2.2 in the snort log files. It seems that antispoofing setup of GIAC Enterprise's routers is not adequate, unless those addresses where spoofed inside the Snort perimeter ... :

```
09/26-05:40:00.709907  [**] NMAP TCP ping! [**] 2.2.2.2:80 ->
MY.NET.6.47:25
09/26-05:52:53.541646  [**] NMAP TCP ping! [**] 2.2.2.2:80 ->
MY.NET.253.42:25
```

Top listeners list

Here is the top ten list of listeners

#detects	Dest-ip	Detects
5808	MY.NET.6.7	mainly (5801) NET-Watchlist 000222 NET-NCFC, others are reconaissance
4814	MY.NET.211.146	mainly (4810) Watchlist 000220 IL-ISDNNET-990517, others are recon
3940	MY.NET.223.98	mainly (3938) Watchlist 000220 IL-ISDNNET-990517, others are recon
3918	MY.NET.206.90	mainly (3914) Watchlist 000220 IL-ISDNNET-990517, others are recon
1813	MY.NET.70.255	All are roadcast Ping to subnet 70
1640	MY.NET.203.142	mainly (1638) Watchlist 000220 IL-ISDNNET-990517, others are recon
1463	MY.NET.218.142	mainly (1459) Watchlist 000220 IL-ISDNNET-990517, others are recon
1371	MY.NET.214.170	mainly (1353) Watchlist 000220 IL-ISDNNET-990517, others are WINGATE 1080 Attempt
1302	MY.NET.100.230	mainly (1299) Watchlist 000222 NET-NCFC, others are reconaissance
952	MY.NET.202.22	mainly (950) Watchlist 000220 IL-ISDNNET-990517, others are reconaissance

It seems that GIAC enterprises are having troubles with NET-NCFC and IL-ISDNNET. They have defined a rule to watch for every traffic coming from those 2 netblocks. Unfortunately, precious data are hidden behind those 2 rules and can't be further investigated. GIAC should consider using another system to do traffic logging, leaving the IDS analyzing signatures.

If we remove everything thats fits under tha watchlists, we find :

#detects	Dest-ip	Ddetects
1813	MY.NET.70.255	All are roadcast Ping to subnet 70
561	MY.NET.101.192	468 SNMP public access, 93 SMB Name Wildcard
490	MY.NET.221.246	488 Attempted Sun RPC high port access, others are TCP SMTP Source Port traffic and reconnaissance
437	MY.NET.225.210	435 Attempted Sun RPC high port access, others are reconaissance
374	MY.NET.206.118	372 WinGate 1080 Attempt, others are reconaissance
366	MY.NET.217.214	365 Attempted Sun RPC high port access, other is reconaissance
323	MY.NET.206.222	299 Attempted Sun RPC high port access, 21 SUNRPC highport access!, others are reconnaissance
187	MY.NET.222.98	All are Attempted Sun RPC high port access
157	MY.NET.226.74	154 are Attempted Sun RPC high port access, others are reconnaissance
136	MY.NET.228.42	132 are Attempted Sun RPC high port access, others are reconnaissance

Here we can see that RPC is the hottest threat at "GIAC enterprises". Every sys admins should be notified of this and should check that their systems are well secured on that specific point.

There is something strange however with the folowing two detects. Those were exception in the listener list as not being from MY.NET :

```
11/22-11:24:06.406682  [**] connect to 515 from inside [**]
MY.NET.179.78:2274 -> 64.244.202.110:515
11/22-11:33:56.296324  [**] connect to 515 from inside [**]
MY.NET.179.78:2707 -> 64.244.202.66:515
```

Sysadmin of MY.NET.179.78 should be contacted in order to understand what's is going on there. Please see the detects explanation for more info.

Detects

Here is the 20 detects found sorted by number of occurences (detects are the result of suspicious activity triggering snort rules). The spp-portscan detects have been removed from the calculation but will be analysed below. Those signatures where found in the SnortA*.txt files :

Detect	#detects	#ip-src	#ip-dst
HAPPY 99 VIRUS	2	2	2
TINY FRAGMENTS - POSSIBLE HOSTILE ACTIVITY	7	5	6
EXTERNAL RPC CALL	13	8	3
SITE EXEC - POSSIBLE WU-FTPD EXPLOIT - GIAC000623	13	4	7
PROBABLE NMAP FINGERPRINT ATTEMPT	15	14	13

CONNECT TO 515 FROM INSIDE	56	2	3
SUNRPC HIGHPORT ACCESS!	60	13	12
NMAP TCP PING!	96	21	20
QUESO FINGERPRINT	142	29	58
SMB NAME WILDCARD	218	33	33
NULL SCAN!	283	204	196
SNMP PUBLIC ACCESS	468	23	1
BACK ORIFICE	1697	40	932
BROADCAST PING TO SUBNET 70	1813	216	1
ATTEMPTED SUN RPC HIGH PORT ACCESS	2542	20	33
TCP SMTP SOURCE PORT TRAFFIC	2893	4	2836
WINGATE 1080 ATTEMPT	4802	570	2655
WATCHLIST 000222 NET-NCFC	8166	45	26
WATCHLIST 000220 IL-ISDNNET-990517	30998	61	108
SYN-FIN SCAN!	56250	30	25751

The detects will be categorized into 4 sets : Reconnaissance, Attacks, Already compromised internal systems and Unkonwn. For every detect, you will find a short description, some link to more detailed info, a comment about the detect and some snort samples.

HAPPY 99 VIRUS

Categorization : Attack

Description: HAPPY99 is a Win32-based e-mail and newsgroup worm. Infected systems (W95 and W99) will automatically attach the HAPPY99 worm to every outgoing e-mail or USENET posting without the notice of the sender. Aliases names to this virus are I-Worm Happy and SKA.

Links: http://www.f-secure.com

Comments: This alert is the HAPPY99 signature. It seems that 2 external sites are infected (see data excerpt below) and are trying to span the virus. The best way to get rid of this kind of atacks if to have good antivirus software at the mail gateway and (or at least) at every workstation in the network [DR03]. One can think of sending en e-mail to those sites admins reporting the threat.

Sample:

```
10/05-03:59:51.460766  [**] Happy 99 Virus [**] 216.6.117.11:41827 ->
MY.NET.253.41:25
11/06-16:06:44.170359  [**] Happy 99 Virus [**] 209.94.224.13:2708 ->
MY.NET.6.35:25
```

TINY FRAGMENTS - POSSIBLE HOSTILE ACTIVITY

Categorization: Reconnaissance/Attack

Description: This kind of alert is indicating that some unnaturally fragmented ip packets where detected (crafted small fragments). One should never see that kind of higly fragmented packets running on the network. Those frags are usually used for Denial of Services attacks by killing the IP stack and is also a way to bypass bad packet filters.

Links: SANS/GIAC Track3.1 handouts

Comments: Real care must be given to this kind of alerts. If there is good firewall or packet filters on the way in, this shouldn't become a threat [DR04]. This activity is a good candidate for further analysis. All traffic would be logged in order to understand what is going under those tiny frags. The 6 destinations hosts should be examined to see if they were compromised.

Sample:

```
09/28-22:02:54.922273  [**] Tiny Fragments - Possible Hostile Activity
[**] 216.43.55.44 -> MY.NET.211.2
```

EXTERNAL RPC CALL

Categorization: Reconnaissance

Description: This alert is due to external access to port 111 where portmapper is runing (port 111). This service can be used to map RPC services running on the system (list of services offered and ports where they can be accessed). This kind of activity is usualy the prelude of an RPC attack. Great care should be given to this alert due to the fact that RPC is a sensitive service usually left open.

Links: none

Comments: One can see that system MY.NET.6.15 is having some success being targeted 9 times from 7 different locations. It seems that MY.NET.6.15 is running some interresting (from the hackers point of view) stuff. Sysadmin of MY.NET.6.15 should be notified of this kind of activity [DR05]. Further analysis shows that MY.NET.6.15 also had a connection to port 32771 from 211.46.110.81 which was one of the source ip found in this analysis. Nothing can be clearly said, but this system should be monitored. Please look at the "SUNRPC highport access" analysis.

Sample:

```
10/10-20:23:36.018641 [**] External RPC call [**] 200.191.80.206:931 ->
MY.NET.6.15:111
```

SITE EXEC - POSSIBLE WU-FTPD EXPLOIT - GIAC000623

Categorization: Attack

Description: SITE EXEC is a feature giving the ability to remote ftp users to execute arbitrary commands on the FTP server.SITE EXEC is usually fired with the QUOTE command on the client side. Some implementations of FTP server (WU-FTPD) are known to have vulnerabilities with SITE EXEC, giving the opportunity to become root on the FTP server.

Links: http://www.cert.org, CA-2000-13 and IN-2000-10,
ftp://ftp.auscert.org.au/pub/auscert/advisory/AA-2000.02

Comments: This kind of alert shows that from the 13 alerts, 9 are coming from 208.61.44.215 hitting 6 internal hosts. The time frame is from 10/01 06:17:23 to 07:46:19. The sysadmin of the 6 internal hosts should be notified as those system can be compromised. Those alerts can, of course, be false positive alerts.

Sample:

```
10/01-06:17:23.004770  [**] site exec -  - GIAC000623 [**]
208.61.44.215:3746 -> MY.NET.205.94:21
```

PROBABLE NMAP FINGERPRINT ATTEMPT

Categorization: Reconnaisance

Description: This alert is triggered by pakets containing an illegal combination of TCP flags : SFPU. This is usually referenced as an nmap fingerprinting attempt.

Links: http://www.insecure.org/nmap/nmap-fingerptnting-article.html

Comments: see [DR08]

Sample:

```
10/06-13:38:00.767581  [**] Probable NMAP fingerprint attempt [**]
128.194.79.228:195 -> MY.NET.206.50:80
```

CONNECT TO 515 FROM INSIDE

Categorization: already compromised internal systems ?

Description: Port 515 is used by lpr/lpd process for print spooling and delivery. No "public" snort rules has been identified, but one can gess that printing over the internet is not quite usual !. This kind of rule can detect "data evasion" to the internet and is either due to misconfigured system (is that possible ?) or already compromised internal system which is spreading info to the internet.

Links: none

Comments: You will find hereafter the traffic flows :

#occurrence	Src-ip	Dst-ip
6	MY.NET.101.142:1020	MY.NET.100.3:515
48	MY.NET.101.142:1022	MY.NET.100.3:515
1	MY.NET.179.78:2274	64.244.202.110:515
1	MY.NET.179.78:2707	64.244.202.66:515

It seems that most of the traffic is from inside to inside and can be considered as normal. 2 detects were coming from MY.NET.179.78 and going to 2 different destinations in the same NETBLOCK. Sysadmin of MY.NET.179.78 should be notified of this strange situation.

Sample:

```
11/22-11:33:56.296324  [**] connect to 515 from inside [**]
MY.NET.179.78:2707 -> 64.244.202.66:515
```

SUNRPC HIGHPORT ACCESS!

Categorization: Attack

Description: Sun RPC's are running on highports (32770 to 32900). RPC is a protocol suffering from a myriad of security-related problems and should be avoided on the Internet (RPC was not originally developped with security in mind, RPC services are usually complex, RPC services usually runs as root, ...). This detect can be triggered by an SYN-ACK sent on the port 32771 (which is one of the RPC services). This kind of detect must be considered as serious.

Links: none

Comments: On the 60 detects, 33 are coming from 216.10.12.30 port 2078 to MY.NET.202.242 and MY.NET.206.222. This is strange and should be verified. Sys admin of targeted MY.NET hosts should be contacted.

Sample:

```
11/11-11:08:56.576798  [**] SUNRPC highport access! [**] 211.46.110.81:690 -
> MY.NET.6.15:32771
```

NMAP TCP PING!

Categorization: Reconnaissance

Description: This kind of detect show packet with the ACK seq number equal to zero. This is theorically possible (can lead to false positive) but should be most likely considered as a nmap scan.

Links: http://www.insecure.org/nmap/nmap-fingerptnting-article.html

Comments: On the 96 detects, 42 was generated by the ip 92.102.197.234 from NETBLOCK to MY.NET.1.8:53 in less than a month. This seems strange to us. Further analysis should be done (is that network reconnaissance or load balancing tool ?).

Sample:

```
11/17-23:29:28.036252  [**] NMAP TCP ping! [**] 192.102.197.234:53 ->
MY.NET.1.8:53
```

QUESO FINGERPRINT

Categorization: Reconnaissance

Description: Queso is a operating system fingerprinting tool. It was developped before Fyodor nmap os detection feature and need a single open port to do the OS detection. The main signature of Queso is packets with the folowing bits set : Syn bit, reserverd bit 1, reserved bit 2. Queso detection is based on the OS reaction to illegal bit combination.

Links: http://www.apostols.org/projectz

Comments: This is a well known OS reconaisance tool that seems to be quite widely used. As Reconaissance is usually the premisses of attack, one can think to put those sources addresses into the red list to be monitored seriously (It is unlikely that all of those source addresses were spoofed because they need to receive info back).

Sample:

```
10/14-07:54:06.133387  [**] Queso fingerprint [**] 64.80.63.121:4480 ->
MY.NET.222.62:6346
```

SMB NAME WILDCARD

Categorization: Reconnaissance

Description: SMB name wildcard alerts is triggered by traffic trying to enumerate the NETBIOS table of a target system. This kind of enumeration can be done with a nbtstat -A ip-of-target-system. It can lead to false positive if the user is doing legitimate use of your system (remote user of GIAC enterprise). All other cases must be considered as premisses of attack.

Links: SANS/GIAC Track3.1 handouts

Comments: Don't let SMB going to and from the Internet.

Sample:

```
11/03-15:03:22.016400  [**] SMB Name Wildcard [**] 168.143.29.9:137 ->
MY.NET.60.17:137
```

NULL SCAN!

Categorization: reconnaissance

Description: Nmap has a feature called the NULL SCAN. Here is an abstract of nmap's man page :

"The idea is that closed ports are required to reply to your probe packet with an RST, while open ports must ignore the packets in question (see RFC 793 pp64). The FIN scan uses a bare (surprise) FIN packet as the probe, while the Xmas tree scan turnson the FIN, URG, and PUSH flags. The Null scan turns off all flags. Unfortunately Microsoft (likeusual) decided to completely ignore the standard and do things their own way. Thus this scan typewill not work against systems running Windows95/NT. On the positive side, this is a good way to distin-guish between the two platforms. If the scan finds open ports, you know the machine is not a Windowsbox. If a -sF,-sX,or -sN scan shows all ports closed, yet a SYN (-sS) scan shows ports beingopened, you are probably looking at a Windows box."

Links: http://www.insecure.org/nmap/nmap-fingerptnting-article.html

Comments: NULL scan is a good way to go trough bad firewall and should be considered if replys are going back.

Sample:

```
09/26-16:23:42.708891  [**] Null scan! [**] 24.27.230.65:6699 ->
MY.NET.219.74:1580
```

SNMP PUBLIC ACCESS

Categorization: Reconnaissance/Attack - Or what ?

Description: SNMP public access is one of the easiest way to obtain huge amount of info from a poorly secured SNMP agent. "public" community name (pasword) seems to be the default in many equipment. Obviously, this kind of rule matching for the word "public" into snmp UDP traffic is subject to a high false positive alarms level. Anyway this kind of alerts can indicate that some reconnaisance is running or this can indicate poorly configured systems.

Links: none

Comments: First, don't use "public" as you community string. NEVER do that [DR06]. Second, in this case,all traffic is internal (from and to MY.NET/16), so we fall into the

poorly configured systems category. Administrators of MY.NET.97.108, MY.NET.97.115,
MY.NET.97.130, MY.NET.97.159, MY.NET.97.171, MY.NET.97.178, MY.NET.97.185, MY.NET.97.189, MY.NET.97.192, MY.NET.97.204, MY.NET.97.208, MY.NET.97.215, MY.NET.97.219, MY.NET.98.106, MY.NET.98.109, MY.NET.98.122, MY.NET.98.123, MY.NET.98.132, MY.NET.98.141, MY.NET.98.160, MY.NET.98.174, MY.NET.98.191, MY.NET.98.197 should be notified as those systems are sending snmp traffic to MY.NET.101.192:161 using “public” as community string.

Sample:

```
10/10-16:15:49.253089  [**] SNMP public access [**] MY.NET.98.122:1066 ->
MY.NET.101.192:161
```

BACK ORIFICE

Categorization: Attack

Description: Back Orifice is the well known Win 9x hacking tool to date. BO is presented as a Win 9x remote administration tool and is very good in that job. But BO is also the dreamed tool for hackers as BO installs silently and is not obvious for (unaware) users to detect. BO add-on plug-ins automatically connects to specific IRC channels such as #BO_OWNED and announce the BO servers ip address there. It just looks like an "offering"

Links: http://www.cultdeadcow.com/tools

Comments: This kind of event indicate an attempt to connect to port 31337. This is a probe and does not indicate a compromise. This alert can be false positive as software program can listen at the same port.

Sample:

```
10/01-23:12:32.651586  [**] Back Orifice [**] 209.94.199.141:31338 ->
MY.NET.60.95:31337
```

BROADCAST PING TO SUBNET 70

Categorization: Attack

Description: Pinging a broadcast address have a strong effects known as amplification. Many hosts in the targeted network will respond to the requester (Windows machines won't). This results in a DoS for the source of the PING. This can also be a way to distinguish between Win machines and others. Source address are usually spoofed.

Links: www.powertech.no/smurf, registry of SARs

Comments: Do not allow directed broadcast to be forwarded by routers. Please see "no ip directed broadcast" command on CISCO routers.

Sample:

```
10/03-21:17:34.169827  [**] Broadcast Ping to subnet 70 [**]
193.230.129.169 -> MY.NET.70.255
```

ATTEMPTED SUN RPC HIGH PORT ACCESS

Categorization: Reconnaissance/Attack

Description: Please see Description of "SUNRPC HIGHPORT ACCESS!". This snort rule is less specifically looking only for traffic to destination port 32771. This kind of rule is subject to high false positive detects, as destination high ports are randomly selected.

Links: none

Comments: On the 2542 detects, 2534 are from hosts in 205.188.153/24 (AOL-DTC) range with source port 4000. This can be ICQ traffic and should be verified. The latter should be seriously considered :

```
10/23-19:39:00.848056  [**] Attempted Sun RPC high port access [**]
200.53.184.66:2078 -> MY.NET.1.8:32771
11/06-20:30:50.415282  [**] Attempted Sun RPC high port access [**]
63.83.225.106:2629 -> MY.NET.205.130:32771
11/06-20:30:52.117613  [**] Attempted Sun RPC high port access [**]
63.83.225.106:2629 -> MY.NET.205.130:32771
11/06-20:30:52.617789  [**] Attempted Sun RPC high port access [**]
63.83.225.106:2629 -> MY.NET.205.130:32771

10/04-05:49:29.920767  [**] Attempted Sun RPC high port access [**]
205.188.153.116:53 -> MY.NET.225.210:32771
10/25-10:31:15.731605  [**] Attempted Sun RPC high port access [**]
205.188.153.107:53 -> MY.NET.217.214:32771
11/11-13:58:54.275509  [**] Attempted Sun RPC high port access [**]
205.188.153.102:53 -> MY.NET.206.222:32771
10/13-00:44:45.101394  [**] Attempted Sun RPC high port access [**]
205.188.153.104:53 -> MY.NET.219.130:32771
```

Sample:

```
10/23-19:39:00.848056  [**] Attempted Sun RPC high port access [**]
200.53.184.66:2078 -> MY.NET.1.8:32771
```

TCP SMTP SOURCE PORT TRAFFIC

Categorization: Attack ?

Description: This rule seems to be triggered by traffic originating from port 25 of a host and going to port 25.

Links: none

Comments: 4 hosts are triggering this rule and going to a lot of systems in MY.NET. All of the source hosts are suspects as no MX records are linked to those addresses and no SMTP services are behind those addresses.

Sample:

```
10/23-13:10:31.789055  [**] TCP SMTP Source Port traffic [**]
24.7.227.215:25 -> MY.NET.2.10:25
```

WINGATE 1080 ATTEMPT

Categorization: Reconnaissance

Description: WinGate is a well known software package (primarily) used to share a single Internet connection by users of a LAN. WinGate concept is being a proxy for all "requesting" hosts to the Internet, hiding all the "requestors" behind the WinGate server's address. As, bu default, WinGate connections are not logged and WinGate will accept ANY incoming connection, WinGate is one of the perfect tool for "anonymization" of your activity. WinGate can be therefore used to launch internet based attack with a somewhat good conceal of the true source.

Links: http://www.cert.org, VN-98.03, http://www.wingate.net

Comments: none

Sample:

```
10/10-17:52:58.855634  [**] WinGate 1080 Attempt [**] 202.9.134.7:51572 ->
MY.NET.97.154:1080
```

WATCHLIST 000222 NET-NCFC

Categorization: Unknown

Description: WATCHLIST ruleset seems to be developped for active "red list" network monitoring. NET-NCFC is the netblock using the network 159.226/16 and is assigned to "The Computer Network Center Chinese Academy Of Science, Institute of Computing Technology, China".

Links: none

Comments: 8166 alerts has been logged. Amongst them 45 different sources address targetting 26 hosts in the MY.NET network. Targeted ports are mainly 25 (7842 hits), 113 (108 hits), 23, 443 and 21. There is 2 top of the list targets : MY.NET.100.230 (1299 hits) and MY.NET.6.7 (5801 hits), mainly on port 25. One can imagine that

activity from this network must be further more analysed based on some previously hostile activity. Nothing realistic can be said here because the snort rule "WATCHLIST 000222 NET-NCFC" prevent others rules to be triggered, hiding real suspicious activity detection from this network. This kind of "red list" monitoring should be done by another instance of IDS [DR07]

Sample:

```
09/30-04:35:43.594065  [**] Watchlist 000222 NET-NCFC [**]
159.226.45.3:2978 -> MY.NET.253.42:25
```

WATCHLIST 000220 IL-ISDNNET-990517

Categorization: Unknown

Description: WATCHLIST ruleset seems to be developped for active "red list" network monitoring. IL-ISDNNET-990517 is the netblock using the network 212.179/16 and is assigned to an Israelian Internet provider called Bezeq-international. As the netname suggest, users of IL-ISDNNET-990517 can be dialup users.

Links: none

Comments: 30998 alerts has been logged. Amongst them, 61 different sources address targetting 108 hosts in the MY.NET network. Targeted ports are mainly high ports. ET ALORS ???
Please see comments for "WATCHLIST 000222 NET-NCFC"

Sample:

```
10/10-12:14:55.619819  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.27.111:4767 -> MY.NET.201.238:6699
```

SYN-FIN SCAN!

Categorization: Reconnaissance

Description: This rule detects packets with the SYN and FIN bit sets. One can consider that this rule has no false positive trigger i(so it's easy to detect) and it is likely that some sources addresses are spoofed just to hide the real source of scanning. SYN-FIN scanning is used as an OS detection tool.

Links: www.whitehats.com/info/ids198

Comments: Port scanning is a noisy activity on the Internet. Please see [DR08]

Sample:

```
10/10-14:02:01.057481  [**] SYN-FIN scan! [**] 212.0.107.107:53 ->
MY.NET.4.31:53
```

spp-portcan detects

6400 portscan activity has benn detected. As already said, portscan is the premisses of attack [DR08] and should be considered seriously. The top scanner is KIONGGIDO YONGIN Office of Education, KOREA. Notifying admins of those netblock should be considered seriously. Another nteresting thing is the folowing MY.NET addresses issuing portscan :

#occurrence	source address
51	MY.NET.1.3
12	MY.NET.221.82
3	MY.NET.101.1
2	MY.NET.5.25
2	MY.NET.1.4

Sysadmins of those systems should be notified. Either those systems are already compromised or users behind those systems are :-).

Defensive recommendations

Here is the list of defensive recommendations that we suggest "GIAC Enterprises" should apply. Those recommendationis are driven by the common sense of network security and is a result of the data analysis done.

One shoud know that the analyst was not aware of any security policies already in place at your site neither of any sensitive systems that has not been seen in the snort datafiles given. The aim of this warning is to point out that there can be some bias between the defensive recommendations given here and the weight "GIAC Enterprises" should give to them.

[DR01] This document is NOT a security risk assesment neither gives you some "proof of concept" of the security policies adequation at your site. "GIAC enterprises" should run those specific analysis in order to have the full picture of their security posture.

[DR02] In order to have a good level of confidence with the technical security infrastructure deployed at "GIAC enterprise", there should be some basic setup available in order to maximize the system uptime : every systems should be powered trough an UPS system and monitored for any power loss. Every system should have plenty of disk space available and some scripts should be ready to download overflowding data to second level devices (TAPES, CDR , ...). At least, there should be some way to request manual intervention in case of disk space missing (some kind of "HIGH WATER" detection scripts). Last but not least, every systems should be time sync'ed.

[DR03] Having good, updated and running antivirus software on every systems should be considered seriously. Some procedures should also be prepared in order to respond to (rapid) spreading of unknown-for-the-moment viruses (remember the "I Love you" virus).

[DR04] Keep in mind that hacking is a highly intensive activity on in computer science. Hackers are always trying to find "the hole" and exploit them. By always having the latest [tested] software release installed on all of your systems will remove some well known vulnerabilities which are so easy to exploit.

[DR05] At install time, many services are running even if they are not needed (is this one of the meaning of "open systems" ? ;-)). Sysadmin should be aware of this and un-needed software should be stopped and even removed from the system. Many applications are having backdoors, well known default password, known vulnerabilities ans so on. So there is no reason to leave unused software running.

[DR06] SNMP is known to be a weak protocol at the security point of view. SNMP activity can easily lead to community string exposure (think of the automatic device discovery feature of your Network Managment Software trying to query snmp agent of newly detected device and using, of course, a list of known internal community). IDS rules should look not only for "public" attemps but also for queries with the community strings you are using internally and comming from unusual locations (i.e. the internet).

[DR07] An IDS is just an IDS and nothing else. A Firewall is just a Firewall and nothing else. Don't use those systems for other goals (adding software, adding functionality), and don't over trust those systems (being an IDS or a Firewall doesn't mean that they are unbreakable).

[DR08] Net reconnaissance should be considered as premisses of an attack. Ideally, every recon activity should be logged. The netblock owner of those networks should be contacted and if there is any presumption of attack the netblock should be added to the redlist in order to be monitored for a while.

[DR09] Know your network. Easy to say, hard to do! By knowing what is the "normal" traffic coming in and going out of your network, you will have a better sensitivity and understanding of what is wrong with some traffic. This can be done trough documentation, sniffing and network exposure analysis, just to name some.

[DR10] Security analysis is a “never ending” process. GIAC Enterprises should consider doing log analysis on a day by day basis. Having a database of previous detects and anomalies used for correlation of actual detects should also be considered.

[DR11] Be prepared to be hacked. All sysadmin and netadmins should be well trained about security risks and intrusion activity. Some aspects need to be emphasized, as they are usually underestimated :
- What posture to adopt when a system is suspected to be compromised,
- What are the tools available to repair after an attack,
- What, how and who to report after intrusion.

Conclusions

The fairly high amount of suspicious traffic logged already drives us to a very first conclusion : The Giac Enterprises network seems to have some success (at least from the hacker point of view :-). This underscore the need for a very strong Security Policy.

Many sysadmins of GIEC enterprises should be contacted (see split of detects). They should be urged to check their systems for :
1) evidence of intrusion (having tripwire running should help),
2) unnecessary exposure (think about RPC),
3) poorly configured applications (think of snmp public community name)
against the detects categorized as attacks. They should be sensitized to the high level of recon the GIAC network is suffering.

All the defensive recommendation should be applied and followed.

Assignment 3 : Analysis process

Introduction

I had a very first question before starting this assigmnent section : will I write some usefull tools which could be re-used from time to time or will I focus on the analysis for this assignment only. The time frame available for this assignment quickly drives me towards the second solution : give'em an analysis with some tools, not some analysis with great tools !

An other question was "will I try to find some usefull tools and use them ?". As I am doing this kind of analysis for the very first time, I prefered to use my knowledge of basic unix tools (sort, cut, sed, awk, ...) which I am confident with, than relying on tools that I don't really understand (yet ;-). This doens't means of course that there are no such usefull tools, and that, at some time, i will not use them.

The analysis was mainly done on a unix system. The report was build on the fly (vi) and compiled as a MSword document afterwards.

During the awk analysis, I've replaced all occurence of [**] by %%%% in the snort alert files (thanks to sed). This gives me the ability to use awk as pattern processor without headache. [If someone know how to define [**] as FS, please let us know].

The steps

Step 1

Downloading the sample snort data, dezipping the 3 files. Should I explain ? ;-)

Step 2

Looking what is inside the samples. It seems that with awk, sort, uniq and sed one will be able to analyse those text data file. As explained above, using sed all occurence of [**] in the alert files must be replaced by something that `awk` will understand. In this case, "%%%%" has been chosen.

```
sed -e "s/\[\*\*\]/%%%%/g" < datafile > new-datafile
```

Step 3

In order to get the time line, extraction of the date and time field (hours only) has been done, using some very simple awk script. Let's look at the most complicated one (:-) :

```
BEGIN  { FS="[ /\.:-]"; Y=2000;
         month["Jan"]=1; month["Feb"]=2; month["Mar"]=3; month["Apr"]=4;
         month["May"]=5; month["Jun"]=6; month["Jul"]=7; month["Aug"]=8;
         month["Sep"]=9; month["Oct"]=10; month["Nov"]=11;
month["Dec"]=12;
       }
/->/ { printf("%04d %02d %02d %02d\n", Y, month[$1], $2,$3);}
```

This script can be ran trough the "scan" files with the folowing shell script

```
awk -f scans.awk $SC/* | sort | uniq -c > $analysis/timeline_for_scans.txt
```

having predefined where are the data files ($SC) and where to put the results ($analysis). It will gives you a split hour by hour with a count of detects. Here is an abstract :

```
...
   1466     2000 09 27 05
     48     2000 09 27 06
     45     2000 09 27 07
      7     2000 09 27 08
      4     2000 09 27 09
     11     2000 09 27 11
     14     2000 09 27 12
      1     2000 09 27 13
     10     2000 09 27 14
      1     2000 09 27 15
      5     2000 09 27 16
      4     2000 09 27 17
     23     2000 09 27 18
     26     2000 09 27 19
      3     2000 09 27 20
...
```

Step 4

Same technique was used to extract the detects distribution in the alerts files. A somewhat more sophisticated `awk` program is extracting the alert identfication and doing some math on src and ip addresses :

```
BEGIN { FS="%%%%";
        IGNORECASE=1;
      }
```

```
/->/  {
        split( $3, tmp, "->");
        split( tmp[1], src, ":");
        split( tmp[2], dst, ":");
        alert = toupper( $2);
        /* printf(" %s  %s  %s \n", alert, src[1], dst[1]);  */
        countalerts[ alert] = countalerts[ alert] + 1;
        alertssrc[ alert, src[1]] = alertssrc[ alert, src[1]] +1;
        alertsdst[ alert, dst[1]] = alertsdst[ alert, dst[1]] +1;
      } ;

END   {

       for ( element in alertssrc )
       {
         split( element, tmp, "\034");
         countalertssrc[ tmp[1]] = countalertssrc[ tmp[1]] + 1;
       }

       for ( element in alertsdst )
       {
         split( element, tmp, "\034");
         countalertsdst[ tmp[1]] = countalertsdst[ tmp[1]] + 1;
       }

       printf("Breakdown of alerts\n");
       printf("Detect, #detects, #ip-src, #ip-dst\n");
       for ( detect in countalerts )
          { printf("%s,%d,%d,%d\n", detect, countalerts[ detect],
                      countalertssrc[ detect], countalertsdst[ detect]); }
      };
```

```
cat $SA/* | awk -f detects.awk | sort -n -t "," -k 2,3 >
analysis/detects.txt
```

Step 5

In order to make some correlation about the source network, I decided to "whois" all sources address found. This was done
with some grep, cut, sort and uniq giving a list of source ip's. Then some bash shell script where applied along with some awk.

```
for ip in `cat ips`
do
  echo "Looking for $ip within ARIN"
  result=`whois $ip@whois.arin.net | sed -e "s/  //g" | awk -f whois-
arin.awk
                                                          ip=$ip`
  echo $result | grep -q "NOTARIN"
  if [ $? = 0 ];  then
     echo " ... Quering the RIPE DB"
     result=`whois $ip@whois.ripe.net | sed -e "s/  //g" | awk -f whois-
                                                          ripe.awk
ip=$ip`
     echo $result | grep -q "NOTRIPE"
     if [ $? = 0 ];  then
```

```
        echo " ... Quering the APNIC DB"
        result=`whois $ip@whois.apnic.net | sed -e "s/  //g" | awk -f
whois-
                                                          apnic.awk
ip=$ip`
        echo $result | grep -q "NOAPNIC"
        if [ $? = 0 ];  then
          echo " ... sorry; not found anywhere"
          result="$ip not found in ARIN-RIPE-APNIC whois DB. Try by hand
!"
        fi
     fi
  fi
  echo $result >> out
done
```

Here is the whois-ripe-ripe.awk script :

```
BEGIN               { FS=":"; numdesc=0; netnum=""; netname=""; }
/No entries found/ { netname="NOTRIPE"; }
/inetnum/           { netnum=$2 }
/netname/           { netname=$2 }
/country/           { desc[ numdesc] = $2; numdesc = numdesc + 1; }
/descr/             { desc[ numdesc] = $2; numdesc = numdesc + 1; }
END                 { printf ("%s|%s|%s|%s %s %s %s\n", ip, netnum,
netname, desc[0], desc[1], desc[2], desc[3], desc[4]); }
```

This kind of tool gives us the folowing list :

```
...
168.191.250.64| 168.191.0.0 - 168.193.255.255| SPLK-DIAL| SPRN
168.191.91.142| 168.191.0.0 - 168.193.255.255| SPLK-DIAL| SPRN
169.132.154.25| 169.132.0.0 - 169.132.255.255| IDT2| IOS
169.232.73.204|169.232.0.0-169.232.255.255|UCLANET4| University of
California, Los Angeles
169.233.14.204|169.233.0.0-169.233.255.255|NSTDATA| University of
California, Santa Cruz
169.254.184.161| 169.254.0.0 - 169.254.255.255| LINKLOCAL|
172.130.97.123| 172.128.0.0 - 172.185.255.255| AOL-172BLK| AOL
172.134.3.235| 172.128.0.0 - 172.185.255.255| AOL-172BLK| AOL
172.141.91.45| 172.128.0.0 - 172.185.255.255| AOL-172BLK| AOL
...
```

Which is more than we need in order to to some correlation about the network triggering the detects.

P.S. I don't know at the time of writing if they are some better tools to do that. I will try to find one or write it.

Step 6

Breakdown explanation. Each kind of alerts is analyzed and described. The tools used here are mainly :

- "search on the net",
- read TRACK 3 handouts,
- look at some snort rulebase available on the net (http://www.whitehats.com)
- Have a look at some security books available.

This kind of reading gives me some good understanding of what was the alert for, what is the security risk here and give some comments about the alert in the contect of GIAC Enterprises (that's why I am paid. No ?).

Of course, a lot of "grep" on the data files were done to look at the specific alerts and trying to make some correlations. At some time, one of the data mining tool used look like this (get a list of source ip and # occurences that fired the Back Orifice alert) (they were many variations on this) :

```
grep -i orifice $SA/* | cut -d"%" -f 9 | cut -d "-" -f 1 | cut -d":" -f 1
| sort | uniq -c
```

Conclusion

It seems clear now that having some kind of database available should help the data mining process. In this case, the only real process done was counting trough the data files and finding some signatures. Having those signatures correlated to known intrusion signatures will help to reduce false positive and false negatives triggers, but moreover, will help the analyse to find suspicious traffic that nned to be documented and further analysed. This process was not done in this assignment due to a lack of time. It could have shown some hidden patterns that could lead to intrusion discovery.