

Weekly Tasks – Week 5

Rich Macfarlane 2013

Week	Date	Teaching	Attended
5	Feb 2013	Lab 7: Snort IDS Rule Development	
<p>Aim: The aim of these labs are to further investigate the Snort, network IDS, and methods for detection rule development, on both a Windows and a Linux platform</p> <p>Time to complete: 4 hours (Two supervised hours in lab, and two additional hours, unsupervised).</p> <p>Activities:</p> <ul style="list-style-type: none">• Complete Lab 7: Snort .pdf from WebCT or http://www.dcs.napier.ac.uk/~cs342/CSN11102/Lab7.pdf (Use Unit 2 – IDS for reference)Optional PartB: .pdf from WebCT or http://www.dcs.napier.ac.uk/~cs342/CSN11102/Lab7B.pdf (Use Unit 2 – IDS for reference)• The End Of Unit Tutorial Questions for the Authentication chapter (from the unit pdf).• Take some End Of Unit Online Tests for the Authentication chapter at: http://www.asecuritysite.com/security/tests/tests?sortBy=sfc04 <p>Learning activities: At the end of these activities, you should understand:</p> <ul style="list-style-type: none">• How to perform deep packet inspection of data packets, using Snort.• How develop Snort rules, by analyzing protocol traffic.• How the NMap network scanner is used to scan networks and systems, and how to detect scans using Snort.• How to use the Windows Superscan network scanner. <p>Reflective statements (end-of-exercise): How is it possible to match on data inside the data packet? Why is it important to tune IDS Sensors? What are the main uses of a network scanner, such as Nmap?</p> <p>References: Course Handbook - Unit 2 IDS Online: Snort User Manual, NMap User Manual.</p>			

Lab 7: Snort IDS - Detection Rule Development

Rich Macfarlane 2013

7.1 Details

Aim: To develop the use of the **Snort IDS** software, to create NIDS Sensors capable of detecting unwanted network traffic, as well as report on possible intrusions.

7.2 Overview

Employing Network IDS (NIDS) sensors around an organisations network, means traffic can be monitored for intrusions and security policy breaches. IDS sensors can be used to report on insider attacks, misuse of network resources, and intrusions from outside the organisations network perimeter.

IDS can be compared to a **burglar alarm** system, with sensors around an organisation's premises. Similar to an IDS, it can raise the alarm if intruders are detected, but cannot stop the attack from proceeding.



This lab gives further IDS development using the **Snort IDS**, including application level analysis, rule development, and network reconnaissance detection. Use the previous lab or online materials as reference for Snort configuration.



Snort documentation, including the Snort Users Manual can be found at:
<http://www.snort.org/docs>

7.3 Lab Setup

The majority of this lab can be run on a standalone machine. Some of the later sections need a second machine, so using the Napier Windows server 2003 Virtual Machine with the Snort IDS sensor running on it, and a second VM such as the Napier Ubuntu VM, is recommended (it can be started later, when needed, to save host resources)

The hosted virtualised lab architecture, using VM Workstation to run the Windows2003 Server VM, is shown below.

Run the Linux virtual image **UBUNTU**. Log into the Linux server as User name: **napier**, Password: **napier123**).

Within the virtual image UBUNTU, open a **Terminal Window** (Applications->Accessories->Terminal) and determine the servers IP Address using **ifconfig**.

7.4 Activities

7.2.1 Snort as Packet Sniffer

On **WINDOWS2003** change to the directory Snort is installed in, typically `c:\snort`

Check the interfaces available to Snort with:

```
C:\Snort> snort -W
```

Run the Snort IDS sensor as a packet sniffer to test it can inspect traffic, using:

```
C:\Snort> snort -dev -i 3
```

Test by creating ICMP traffic from **UBUNTU**.

7.2.2 Snort as an IDS Sensor – Configure Detection Rules

To run Snort as an IDS Sensor, a *Snort Detection Rules file* is used as input to the Snort software to specify the traffic to detect and report on, as shown below.

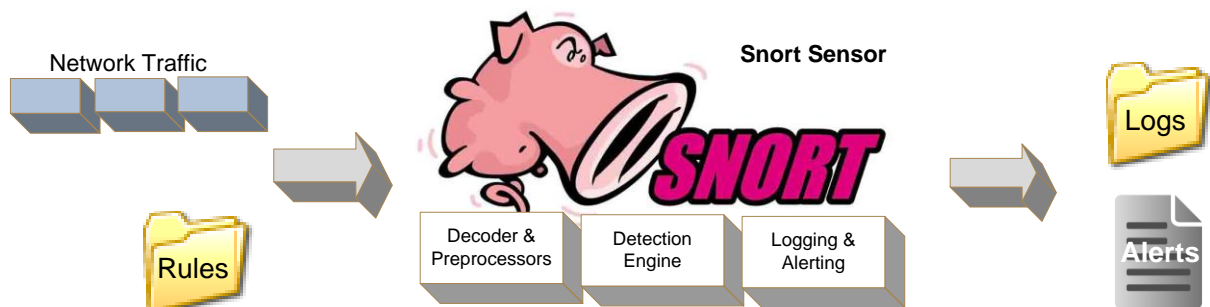


Figure 1 - Snort Sensor Architecture

Snort can be used as a signature-based IDS, with signatures defined in the rules. Signature-based IDS can compare signatures against each packets application protocol data (the packet payload). This is sometimes called *deep packet inspection*.

Firewalls tend to filter at lower levels as inspecting deep inside each packet can reduce throughput. As IDS sensors are inspecting copies of the packets off-line, there is no effect on throughput. This means IDS sensors can inspect packets much more thoroughly than most firewalls.

7.2.3 Create Snort Detection Rules for Facebook Traffic

On **WINDOWS2003** using Windows Explorer and a text editor, create a Snort Detection Rules file `c:\snort\rules\rules.txt` where our Snort rules will be created.

To detect Facebook use by employees, an IDS detection rule can be created which will detect the text "facebook.com" in the payload of HTTP packets being downloaded from a web server.

```
alert tcp any 80 -> any any ( \
    msg:"Facebook web traffic detected"; \
    content:"facebook.com"; sid:10000; )
```

Note: The \ character is used to split rules over multiple lines.

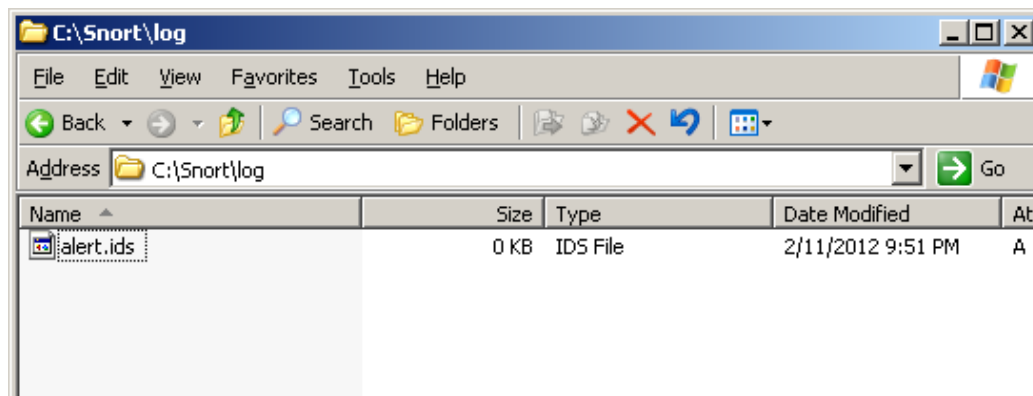
Test Detection Rules

From the `c:\snort` directory, run the Snort IDS sensor using the `rules.txt` file as input detection signatures, and logging alerts to the default Snort log folder, using:

```
snort -dev -i X -p -c rules\rules.txt -l log -K ascii -k none
```

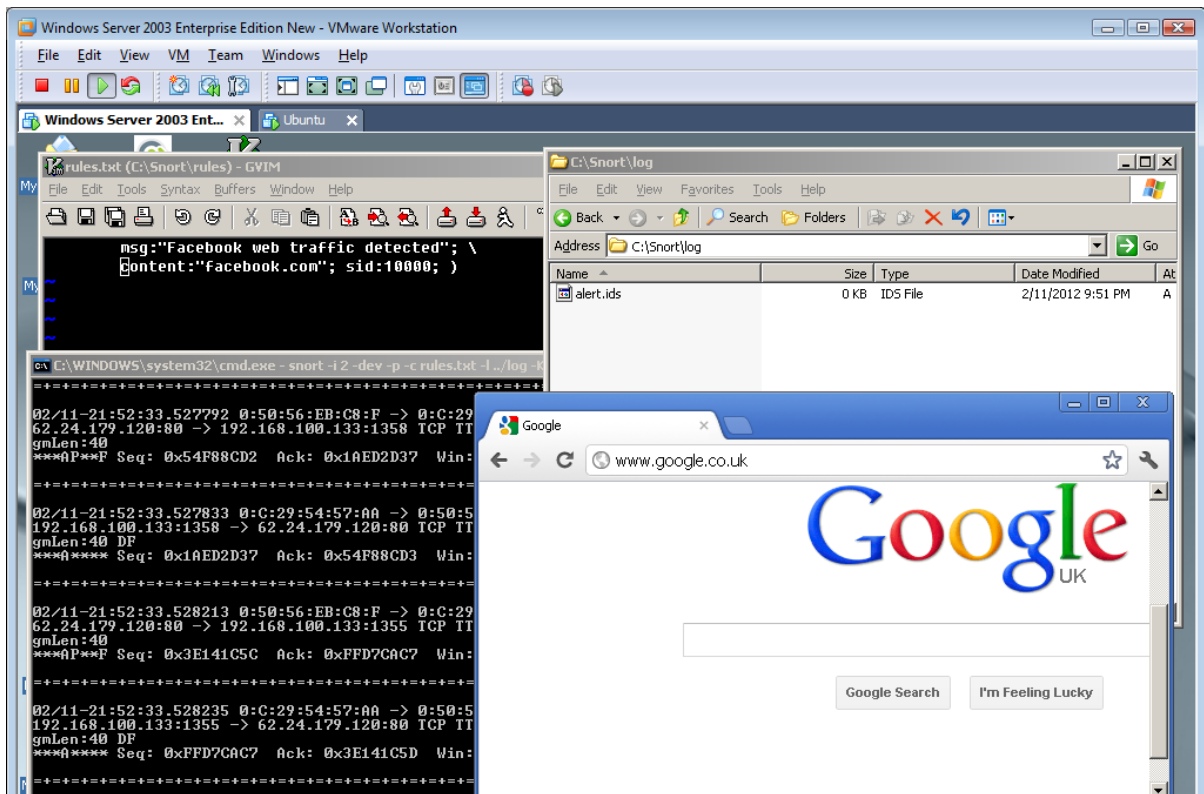
Where `X` is the interface to capture on.

Now monitor the output folder `c:\snort\log` folder with Windows Explorer. Right click the file panel and select **View>Details**, as shown below. The timestamp and size of the file can now be viewed, and used to check if a rule has caused an alert to be raised, and packet details to be logged.

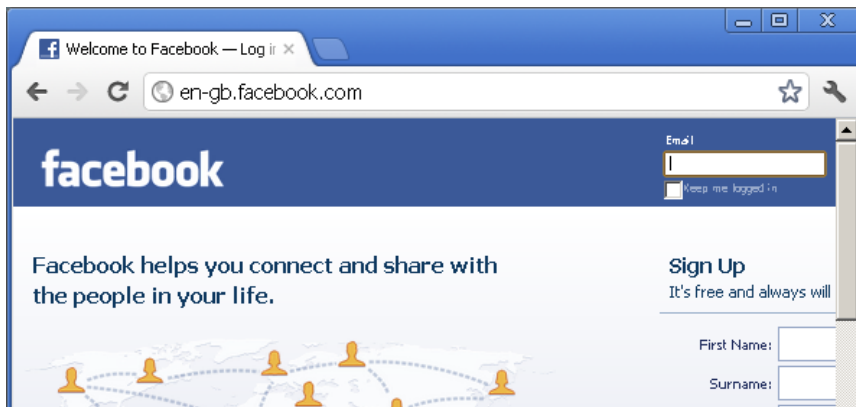


To test, run a web browser.

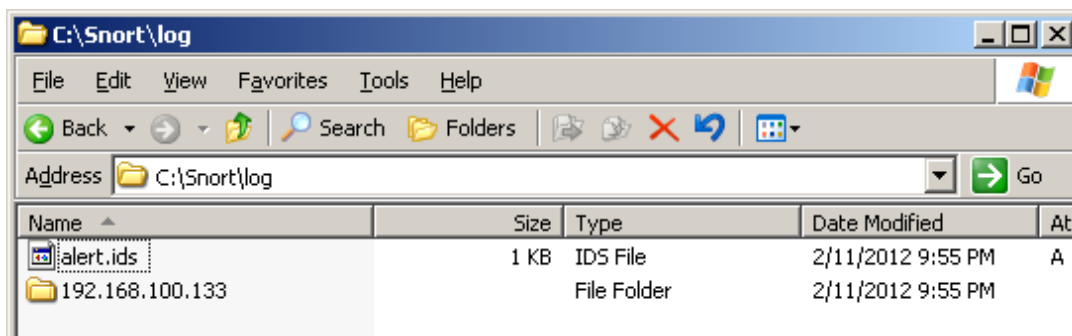
Resize/rearrange your windows, so you can see the output `alert.ids` file in explorer, the console window running `snort`, and the web browser window.



Use the web browser to navigate to the default Facebook login page, as shown below.



Check if an alert has been generated, by checking the log folder, as shown below. Stop Snort running with <CTRL+C>, and open the alert file in an editor.

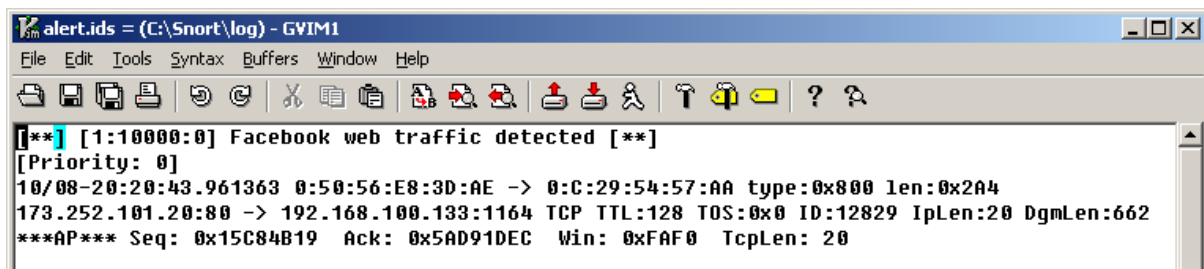


Q: Has an alert been generated for the facebook traffic?

YES/NO

Q: What is the IP Address of the facebook server, which the packet came from?

The alert file should look similar to the following:



```
alert.ids = (C:\Snort\log) - GVIM1
File Edit Tools Syntax Buffers Window Help
[1:10000:0] Facebook web traffic detected [**]
[Priority: 0]
10/08-20:20:43.961363 0:50:56:E8:3D:AE -> 0:C:29:54:57:AA type:0x800 len:0x2A4
173.252.101.20:80 -> 192.168.100.133:1164 TCP TTL:128 TOS:0x0 ID:12829 IpLen:20 DgmLen:662
***AP*** Seq: 0x15C84B19 Ack: 0x5AD91DEC Win: 0xFAF0 TcpLen: 20
```

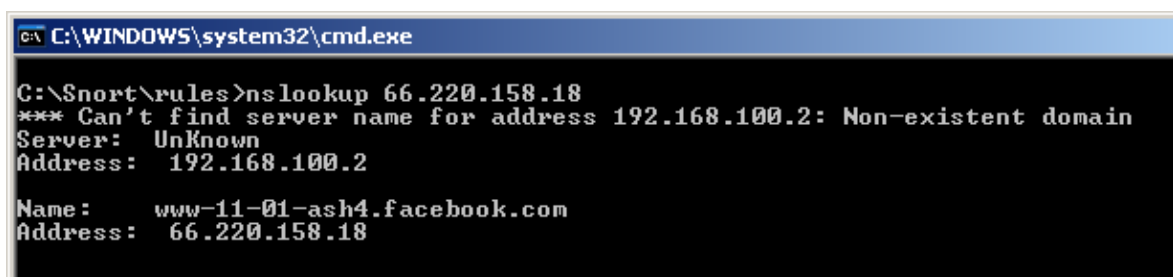
From the Windows command line, use **nslookup** (name server lookup tool) to check for DNS information on the server such as with the following:

```
C:\> nslookup serverIPAddress
```

Q: Does it confirm the server from the facebook.com domain?

YES/NO

The output should be similar to shown below:



```
C:\WINDOWS\system32\cmd.exe
C:\Snort\rules>nslookup 66.220.158.18
*** Can't find server name for address 192.168.100.2: Non-existent domain
Server: Unknown
Address: 192.168.100.2

Name: www-11-01-ash4.facebook.com
Address: 66.220.158.18
```

7.2.4 Create Detection Rules for Job Searching

Create another Snort detection rule to detect employees doing internet searches for the word “jobs” (add it to the rules.txt file). The alert should be “Job searching found in Web traffic”.

Note: Each snort rule must have a unique **SID**.

Test Detection Rules

Test the rule, by using a search engine to search for the string “Security jobs”.

Q: Does the Snort IDS Sensor detect the job search, and produce an alert?

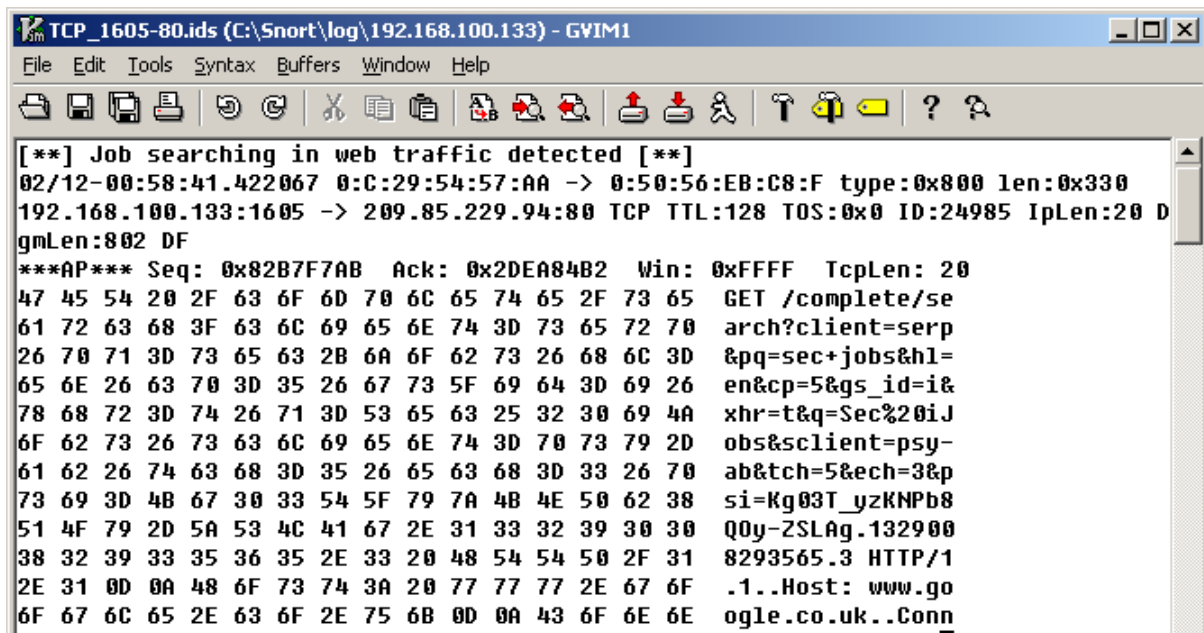
YES/NO

Q: What is the working Snort rule?

Q: Can you locate and view the associated packet logged by Snort, and can you identify the search string in the payload of the packet.

YES/NO

The packet details file should look similar to the following:



The screenshot shows a window titled 'TCP_1605-80.ids (C:\Snort\log\192.168.100.133) - GVIM1'. The window contains a Snort log entry for a detected rule. The log entry is as follows:

```
[**] Job searching in web traffic detected [**]
02/12-00:58:41.422067 0:C:29:54:57:AA -> 0:50:56:EB:C8:F type:0x800 len:0x330
192.168.100.133:1605 -> 209.85.229.94:80 TCP TTL:128 TOS:0x0 ID:24985 IpLen:20 D
gmLen:802 DF
***AP*** Seq: 0x82B7F7AB Ack: 0x2DEA84B2 Win: 0xFFFF TcpLen: 20
47 45 54 20 2F 63 6F 6D 70 6C 65 74 65 2F 73 65 GET /complete/se
61 72 63 68 3F 63 6C 69 65 6E 74 3D 73 65 72 70 arch?client=serp
26 70 71 3D 73 65 63 2B 6A 6F 62 73 26 68 6C 3D &pq=sec+jobs&hl=
65 6E 26 63 70 3D 35 26 67 73 5F 69 64 3D 69 26 en&cp=5&gs_id=i&
78 68 72 3D 74 26 71 3D 53 65 63 25 32 30 69 4A xhr=t&q=Sec%20iJ
6F 62 73 26 73 63 6C 69 65 6E 74 3D 70 73 79 2D obs&sclient=psy-
61 62 26 74 63 68 3D 35 26 65 63 68 3D 33 26 70 ab&tch=5&ech=3&p
73 69 3D 4B 67 30 33 54 5F 79 7A 4B 4E 50 62 38 si=Kg03T_yzKNPb8
51 4F 79 2D 5A 53 4C 41 67 2E 31 33 32 39 30 30 Q0y-ZSLAg.132900
38 32 39 33 35 36 35 2E 33 20 48 54 54 50 2F 31 8293565.3 HTTP/1
2E 31 0D 0A 48 6F 73 74 3A 20 77 77 77 2E 67 6F .1..Host: www.go
6F 67 6C 65 2E 63 6F 2E 75 6B 0D 0A 43 6F 6E 6E ogle.co.uk..Conn
```

7.2.5 Developing Snort Detection Rules - Methods

As well as developing your own detection rules, peer reviewed Snort detection rules can be downloaded and used from the Internet, such as from the Snort web site. These can then be tailored quickly to specific needs.

To develop your own Snort rules, it is recommended that the vulnerability be researched or the intrusion traffic generated in a controlled environment while Snort or another packet sniffer capture the network traffic. The traffic should then be analysed, along with researching any protocols involved, and a Snort rule(s) created based in any triggering conditions (Roesch, 1999).

7.2.6 Develop a Snort Rule to Detect Ping Traffic

Ping is a utility, which is typically used to test connectivity. To capture the traffic, run Wireshark, and listen on an interface which you can send packets though. (see previous Wireshark lab for details)

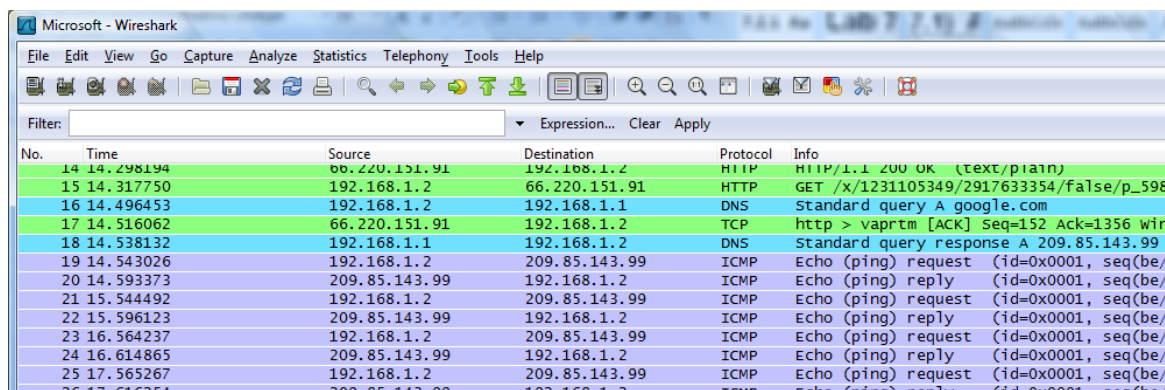
From the local command line, open a new command windows, and Ping a neighbouring machine, an internet server, or another VM, as shown below.

```
C:\Snort\bin>ping google.com

Pinging google.com [209.85.143.99] with 32 bytes of data:
Reply from 209.85.143.99: bytes=32 time=51ms TTL=52
Reply from 209.85.143.99: bytes=32 time=52ms TTL=52
Reply from 209.85.143.99: bytes=32 time=51ms TTL=52
Reply from 209.85.143.99: bytes=32 time=52ms TTL=52

Ping statistics for 209.85.143.99:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 51ms, Maximum = 52ms, Average = 51ms
```

Stop the wireshark capture, and scroll up to the Ping packets, as shown.



Select some Ping packets and review the details.

Q: What is the protocol of the packets generated by the ping tool?

Q: Which Wireshark display filter can be used to filter out these packets?

Q: What is the payload (data) within the packet?

Q: Is this the same for all the Ping packets? (research online)

YES/NO

Create Snort Detection Rule

Create a new Snort rule (add it to the rules.txt file) to detect Windows Ping. The alert should be "Windows Ping Detected – Possible Reconnaissance"

Test the rule

First delete everything in the log folder c:\snort\log with Windows Explorer.

Then run Snort, while monitoring the output log directory as before.

Generate traffic using ping from the command line, while monitoring the log file for alerts.

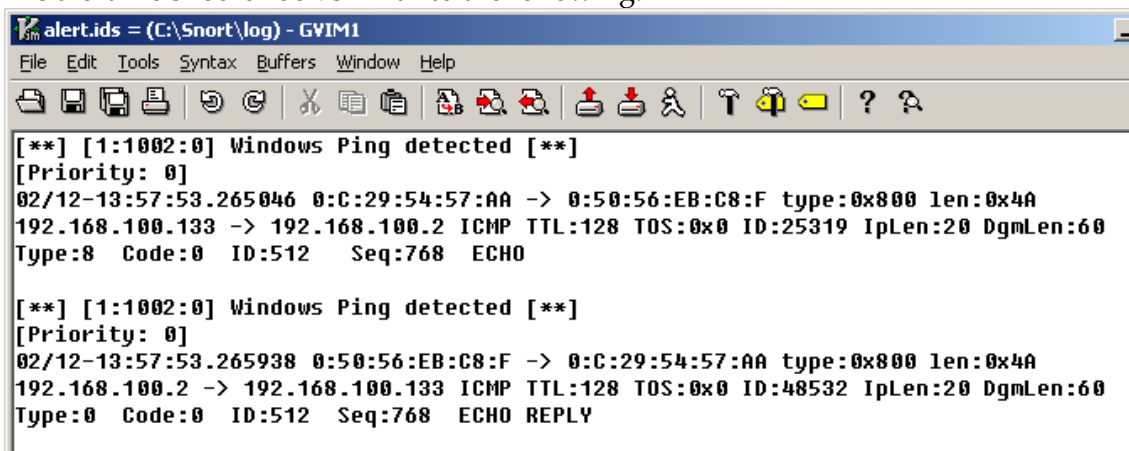
Q: What is the working Snort rule?

Q: How might this signature differ for Ping traffic from a Linux OS?

Q: How many alerts did the ping generate?

Q: Why this many alerts?

The alert file should look similar to the following:



```
alert.ids = (C:\Snort\log) - GVIM1
File Edit Tools Syntax Buffers Window Help
[**] [1:1002:0] Windows Ping detected [**]
[Priority: 0]
02/12-13:57:53.265046 0:C:29:54:57:AA -> 0:50:56:EB:C8:F type:0x800 len:0x4A
192.168.100.133 -> 192.168.100.2 ICMP TTL:128 TOS:0x0 ID:25319 IpLen:20 DgmLen:60
Type:8 Code:0 ID:512 Seq:768 ECHO

[**] [1:1002:0] Windows Ping detected [**]
[Priority: 0]
02/12-13:57:53.265938 0:50:56:EB:C8:F -> 0:C:29:54:57:AA type:0x800 len:0x4A
192.168.100.2 -> 192.168.100.133 ICMP TTL:128 TOS:0x0 ID:48532 IpLen:20 DgmLen:60
Type:0 Code:0 ID:512 Seq:768 ECHO REPLY
```

Change the Snort ICMP detection rule to only alert on Pings which are outgoing from your host system. The alert should be “Windows Ping Detected - Outgoing”

Test the changed rule.

Q: How many alerts did the ping generate?

Q: What is the working Snort rule?

Add a new Snort ICMP detection rule to only alert on Pings which are generated within the local network network, but are destined for an external IP Address. (Hint: the CIDR /24 and the negation operator ! might help)

The alert should be “Windows Ping Detected – Inside to Outside Net”

Test the rule **DOES NOT** fire for Pings on the local network first. Ping a single packet to a device on the local network, such as shown below.

```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\Administrator>ping -n 1 192.168.100.2
Pinging 192.168.100.2 with 32 bytes of data:
Reply from 192.168.100.2: bytes=32 time<1ms TTL=128
Ping statistics for 192.168.100.2:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\Documents and Settings\Administrator>_
```

Q: Which rule fired, and how many alerts were generated?

Now test the rule for Pings from the local network to an external network. Ping a single packet to a server on the internet, such as **google.com**.

Q: What is the working Snort rule?

The alert file should look similar to the following:

```
alert.ids = (C:\Snort\log) - GVIM1
File Edit Tools Syntax Buffers Window Help
[Icons]

[**] [1:1002:0] Windows Ping Outgoing [**]
[Priority: 0]
02/12-14:15:55.708398 0:C:29:54:57:AA -> 0:50:56:EB:C8:F type:0x800 len:0x4A
192.168.100.133 -> 192.168.100.2 ICMP TTL:128 TOS:0x0 ID:25374 IpLen:20 DgmLen:60
Type:8 Code:0 ID:512 Seq:6400 ECHO

[**] [1:1004:0] Ping detected - to outside net [**]
[Priority: 0]
02/12-14:17:14.082890 0:C:29:54:57:AA -> 0:50:56:EB:C8:F type:0x800 len:0x4A
192.168.100.133 -> 209.85.229.105 ICMP TTL:128 TOS:0x0 ID:25379 IpLen:20 DgmLen:60
Type:8 Code:0 ID:512 Seq:6656 ECHO

12,0-1
```

7.2.7 Developing Snort Detection Rules – Network Scanning

Typical network **Reconnaissance** involves Network Scanning. **Host Scanning** where hosts are located on a target network, and **Port Scanning**, where systems are scanned to find services running on a remote machine. IDS such as Snort can be configured to detect this type of behaviour.

Open a command window, and using the **netstat** command, determine your systems connected ports.

Then using **netstat -a -p TCP**, determine the all your machines TCP listening ports (servers running on the local machine).

Note: **-n** can be used to check port numbers.

Q: List some of the well known TCP-based services running?

7.2.8 The NMap Network Scanner

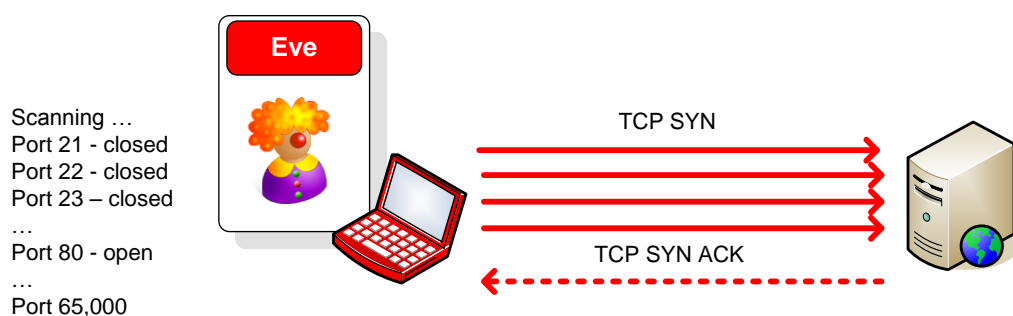
The Nmap (NetworkMAPper) network scanner can perform a wide variety of network scans, and can be used for network exploration, network inventory, and network security auditing and assessments.

A command line nmap executable, or a GUI version Zenmap can be used.



The **nmap** users manual is available at:
<http://nmap.org/book/man.html>

A typical scan would be a **Port Scan** which is used to determine the network services which are running on a specific target machine.



This is performed by sending packets to ports, on the target machine, and checking which services respond. A good example, and the default Nmap scan, is a TCP SYN scan, or Half-open scan, where TCP SYN packets are crafted by Nmap, and sent to the target, and the SYN ACK response packets show that TCP services are running on the target.

Note: DO NOT port scan any machines out with the local subnet. (Most ISPs prohibit port scanning and maintain the right to disconnect customers who perform such activities)



If **nmap** is not on your system, the latest version can be downloaded from:
<http://nmap.org/download.html>

A basic nmap port scan (TCP SYN scan) against a single system should look something like the below:

```
C:\>nmap 192.168.1.1

Starting Nmap 5.00 ( http://nmap.org ) at 2011-02-10 23:51 GMT Standard Time
Stats: 0:01:50 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 94.90% done; ETC: 23:53 (0:00:05 remaining)
Interesting ports on 192.168.1.1:
Not shown: 997 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
23/tcp    open  telnet
80/tcp    open  http
MAC Address: 00:16:E3:ED:55:9A (Askey Computer)

Nmap done: 1 IP address (1 host up) scanned in 139.07 seconds
```

Note: From the Windows version of nmap, a scan must be performed from another machine, as it cannot scan the local system.

Scan your machine, by running the nmap scanner from the host machine, or from another VM (such as The Napier UBUNTU Linux VM):

nmap WindowsPC_IPAddress

Q: List some services and their ports, Nmap reported?

On the Windows machine, determine the TCP services running, and their port numbers:

netstat -a -p TCP -n

Q: Do the services match what Nmap reported?

YES/NO

The output should look similar to the below (based on Windows2003 server VM as target machine):

```
napier@ubuntu: ~  
File Edit View Terminal Help  
Starting Nmap 5.00 ( http://nmap.org ) at 2012-02-11 18:06 PST  
Interesting ports on 192.168.100.133:  
Not shown: 975 closed ports  
PORT      STATE SERVICE  
7/tcp     open  echo  
9/tcp     open  discard  
13/tcp    open  daytime  
17/tcp    open  qotd  
19/tcp    open  chargen  
21/tcp    open  ftp  
23/tcp    open  telnet  
25/tcp    open  smtp  
42/tcp    open  nameserver
```

```
C:\>netstat -a -p TCP -n  
Active Connections  
Proto Local Address Foreign Address State  
TCP 0.0.0.0:7 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:9 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:13 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:17 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:19 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:21 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:23 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:25 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:42 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:53 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:80 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:135 0.0.0.0:0 LISTENING
```

Q: Did the Nmap scan report any UDP services running?
YES/NO

Q: Why might this be?

The default Nmap scan is a TCP SYN Scan. Scan your machine again with Nmap, this time with a UDP scan: (the scan may need sudo from a Linux OS)

```
sudo nmap -sU WindowsPC_IPAddress
```

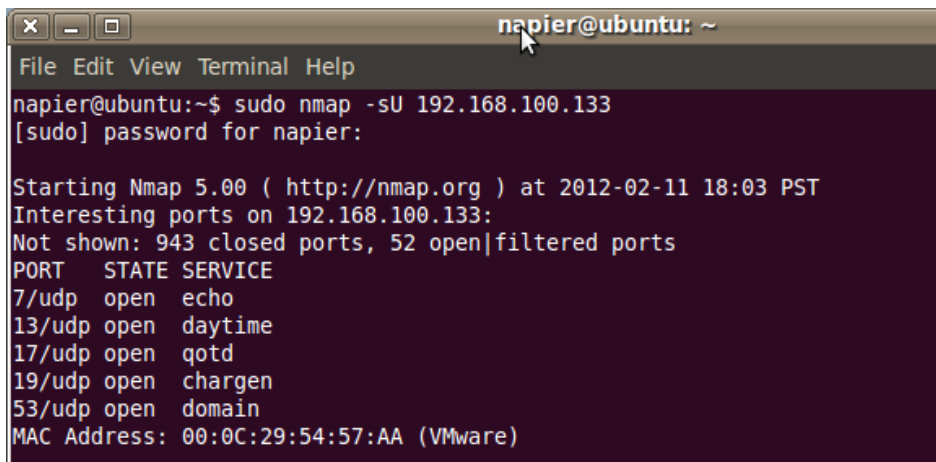
Q: List some services and their ports, Nmap reported?

On the Windows machine, determine the TCP services running, and their port numbers:

```
netstat -a -p TCP -n
```

Q: Do the services match what Nmap reported?
YES/NO

The output should look similar to the below (based on Windows2003 server VM as the target machine):



```
napier@ubuntu: ~  
File Edit View Terminal Help  
napier@ubuntu:~$ sudo nmap -sU 192.168.100.133  
[sudo] password for napier:  
  
Starting Nmap 5.00 ( http://nmap.org ) at 2012-02-11 18:03 PST  
Interesting ports on 192.168.100.133:  
Not shown: 943 closed ports, 52 open|filtered ports  
PORT      STATE SERVICE  
7/udp     open  echo  
13/udp    open  daytime  
17/udp    open  qotd  
19/udp    open  chargen  
53/udp    open  domain  
MAC Address: 00:0C:29:54:57:AA (VMware)
```

7.2.9 Create Snort Detection Rule to Detect a SYN Scan Manually

Create a new Snort rule (add it to the rules.txt file) to detect a SYN Scan. The alert should be “Windows Ping Detected – Possible Reconnaissance”

As before, capture the Nmap port scan traffic, with Wireshark, and analyse for a signature to detect.

Create a Snort rule which detects the incoming SYN flag from the scanner. (See the Course Handbook - Chapter 2 or the Snort Manual online on how to detect TCP flags)

Test the rule

Again delete everything in the log folder c:\snort\log with Windows Explorer.

Then run Snort, while monitoring the output log directory as before.

Generate traffic to test the rule using the Nmap TCP SYN Scan, while monitoring the log file for alerts.

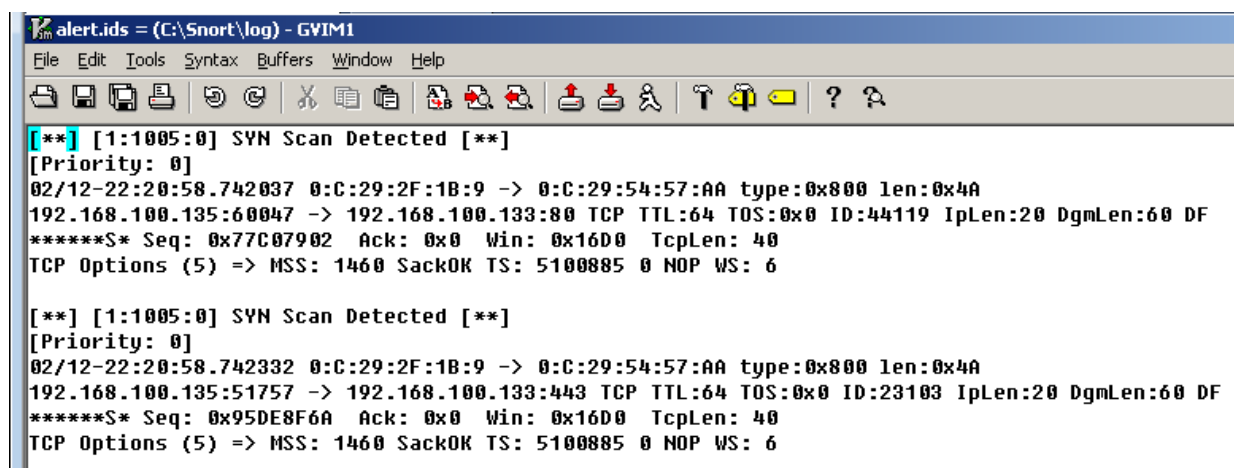
Q: What is the working Snort rule?

Q: How many alerts did the test scan generate?

Q: How many alerts per port scan would be ideal?

Q: Would this rule generate false positives? Why?

The Snort alerts should be similar to those shown below:



```
alert.ids = (C:\Snort\log) - GVIM1
File Edit Tools Syntax Buffers Window Help

[**] [1:1005:0] SYN Scan Detected [**]
[Priority: 0]
02/12-22:20:58.742037 0:C:29:2F:1B:9 -> 0:C:29:54:57:AA type:0x800 len:0x4A
192.168.100.135:60047 -> 192.168.100.133:80 TCP TTL:64 TOS:0x0 ID:44119 IpLen:20 DgmLen:60 DF
*****S* Seq: 0x77C07902 Ack: 0x0 Win: 0x1600 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 5100885 0 NOP WS: 6

[**] [1:1005:0] SYN Scan Detected [**]
[Priority: 0]
02/12-22:20:58.742332 0:C:29:2F:1B:9 -> 0:C:29:54:57:AA type:0x800 len:0x4A
192.168.100.135:51757 -> 192.168.100.133:443 TCP TTL:64 TOS:0x0 ID:23103 IpLen:20 DgmLen:60 DF
*****S* Seq: 0x95DE8F6A Ack: 0x0 Win: 0x1600 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 5100885 0 NOP WS: 6
```

7.2.10 Detecting Network Scans using Snort IDS Pre-Processor

Rules cannot detect scans easily, as a scan uses many packets and these can mixed in with many other packets. Snort has a pre-processor module to help, which groups packets together into TCP streams (The Snort Manual or the IDS unit notes has more detail on this).

Add the Snort port scan pre-processor and the correct parameters, which allows a port scan to be detected. (pre-processors have to be the first rules in the rules file)

Test the Preprocessor

Again delete everything in the log folder c:\snort\log with Windows Explorer.

Then run Snort, while monitoring the output log directory as before.

Generate scan traffic to test the rule using the Nmap TCP SYN Scan, while monitoring the log file for alerts.

Q: Which Pre-processor, and with which parameters, was successful?

Q: Did Snort detect the port scan?

YES/NO

Other tools can be used to perform port scans such as the extremely useful network packet crafting tool **netcat**. Netcat is often called the swiss army knife of network tools as it can be used for many purposes.

From **UBUNTU** port scan **WINDOWS2003** for ports 1 to 100 with:

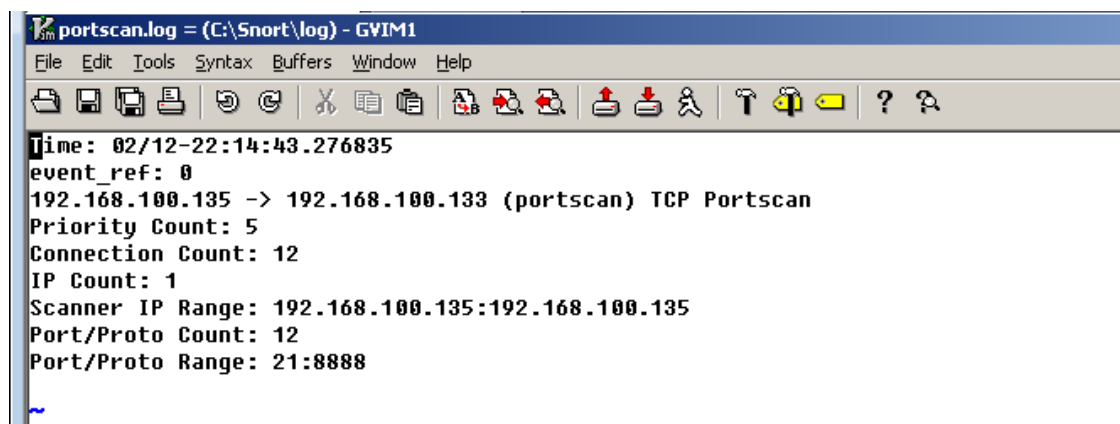
```
napier@ubuntu:~$ nc -v -z 192.168.100.133 1-100
```

Q: Which ports was netcat able to connect to?

Q: Did Snort detect the port scan?

YES/NO

The Snort network scanning detection output should look similar to the below:



7.2.11 IDS Evasion

Try to evade the IDS detection by using stealthy scans, and at the same time tune the IDS sensor to detect the scans.

This can be done by varying the parameters of nmap (-T parameter), while tuning the Snort IDS scan detection pre-processor (Sense level).

Q: Can you evade the Snort detection?

YES/NO

Try only scanning small ranges of ports using netcat, such as 21-25

Q: Did Snort detect the smaller port scan?

YES/NO

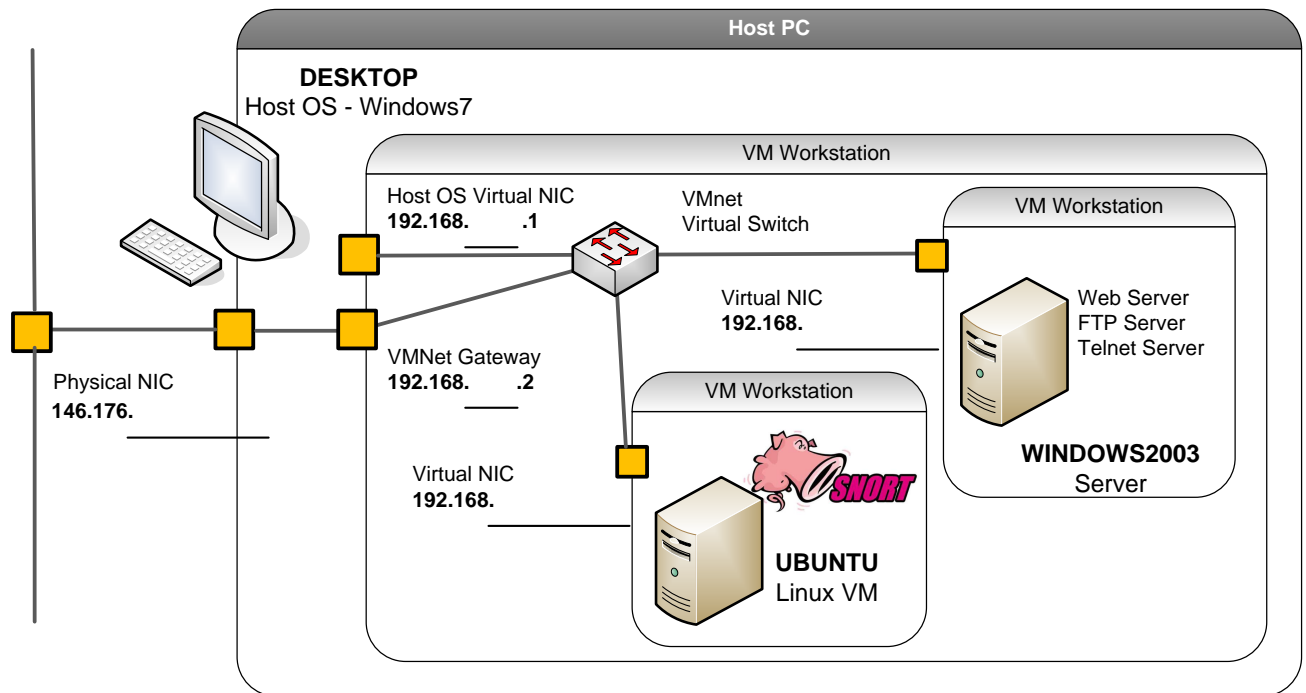
Try scanning the same small ranges of ports using netcat with Snort still running.

Q: Did Snort detect the second scan?

YES/NO

7.2.12 Snort on Linux

We can also run Snort on the Linux box **UBUNTU**, and use the **WINDOWS2003** VM as the attacker/security tester machine, as shown below.



7.2.13 Snort as Packet Sniffer

On **UBUNTU** check the interfaces available to Snort with **ifconfig**.

Run the Snort IDS sensor as a packet sniffer to test it can inspect traffic, against the appropriate interface (eth1 in the example below) using the following command. (Use CTRL+C to stop the Snort Sensor running):

```
napier@ubuntu:~$ sudo snort -dev -i eth6
```

Snort should run, and you should see the Not Using PCAP_FRAMES msg as shown below

```

napier@ubuntu:~$ sudo snort -dev -i eth6
Running in packet dump mode

    === Initializing Snort ===
Initializing Output Plugins!
Initializing Network Interface eth6
Decoding Ethernet on interface eth6

    === Initialization Complete ===

,,_      -*> Snort! <*-
o"_)~    Version 2.8.5.2 (Build 121)
' '      By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
        Copyright (C) 1998-2009 Sourcefire, Inc., et al.
        Using PCRE version: 7.8 2008-09-05

Not Using PCAP FRAMES

```

```
8.10.179.164:80 -> 192.168.100.133:1236 TCP TTL:128 TOS:0x0 ID:18565 IpLen:20 DgmLen:40
***A*** Seq: 0x45D279D2 Ack: 0x973C9DD1 Win: 0xFAEF TcpLen: 20

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

10/08-15:41:46.010696 0:C:29:54:57:AA -> 0:50:56:E8:3D:AE type:0x800 len:0x3C
192.168.100.133:1227 -> 8.10.179.160:80 TCP TTL:128 TOS:0x0 ID:9384 IpLen:20 DgmLen:41 DF
***A*** Seq: 0x210D281D Ack: 0x4635862F Win: 0xFFFF TcpLen: 20
00

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

10/08-15:41:46.011402 0:50:56:E8:3D:AE -> 0:C:29:54:57:AA type:0x800 len:0x3C
8.10.179.160:80 -> 192.168.100.133:1227 TCP TTL:128 TOS:0x0 ID:18566 IpLen:20 DgmLen:40
***A*** Seq: 0x4635862F Ack: 0x210D281E Win: 0xFAEF TcpLen: 20

==+=+~+=+~+=+~+=+~+=+~+=+~+=+~+=+~+=+~+=+~+=+~+=+~+=+~+=+~+=+~+=
```

Q:	Received Packets Total?
Q:	IPv4 Packets Total?
Q:	ICMP Packets Total?

To run Snort as an IDS Sensor, a *Snort Detection Rules file* is used as input to the Snort software to specify the traffic to detect and report on.

```
napier@ubuntu:~$ cd ~
napier@ubuntu:~$ mkdir snort
napier@ubuntu:~$ cd snort
napier@ubuntu:~$ mkdir rules
napier@ubuntu:~$ mkdir log
```

```
napier@ubuntu:~$ vi rules/snort.rules
```

```
napier@ubuntu:~$ gedit rules/snort.rules &
```

Add the signatures from the WINDOWS2003 server Snort sensor rules file, for Windows and Linux ping detection, such as the following:

```
napier@ubuntu: ~/snort
File Edit View Terminal Help
# Snort rules
#
alert icmp any any -> any any (msg:"Windows Ping detected"; content:"abcdef"; sid:1002)
alert icmp any any -> any any (msg:"Linux Ping detected"; content:"12345"; sid:1003)
```

From UBUNTU, in the snort dir, run the snort sensor (against the appropriate interface):

```
sudo snort -i eth0 -dev -p -c rules/snort.rules -l log -K ascii -k none
```

To monitor the alerts being generated by the Snort IDS Sensor open a second terminal window on the Linux VM. The output file can be checked for any lines being appended to it using the **tail** command, as shown below.

```
sudo tail -f log/alert
```

To test the detection rules, ping the Linux VM UBUNTU from WINDOWS2003.

Q: Did snort detect the ping traffic?

YES/NO

Q: Which rule fired?

From UBUNTU, open another terminal window, and ping the WINDOWS2003.

Q: Did snort detect the ping traffic?

YES/NO

Q: Which rule fired?

The output from the tail command should look something like the following.

```
napier@ubuntu:~/snort/log$ tail -f alert
10/08-16:04:54.602306 0:C:29:3E:8A:FE -> 0:C:29:54:57:AA type:0x800 len:0x62
192.168.100.129 -> 192.168.100.133 ICMP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:84 D
F
Type:8 Code:0 ID:57099 Seq:3 ECHO

[**] [1:1003:0] Linux Ping detected [**]
[Priority: 0]
10/08-16:04:54.603204 0:C:29:54:57:AA -> 0:C:29:3E:8A:FE type:0x800 len:0x62
192.168.100.133 -> 192.168.100.129 ICMP TTL:128 TOS:0x0 ID:9674 IpLen:20 DgmLen:
84 DF
Type:0 Code:0 ID:57099 Seq:3 ECHO REPLY
```

7.2.15 The SuperScan Network Scanner

The SuperScan network scanner is a well known Windows only GUI based scanner.

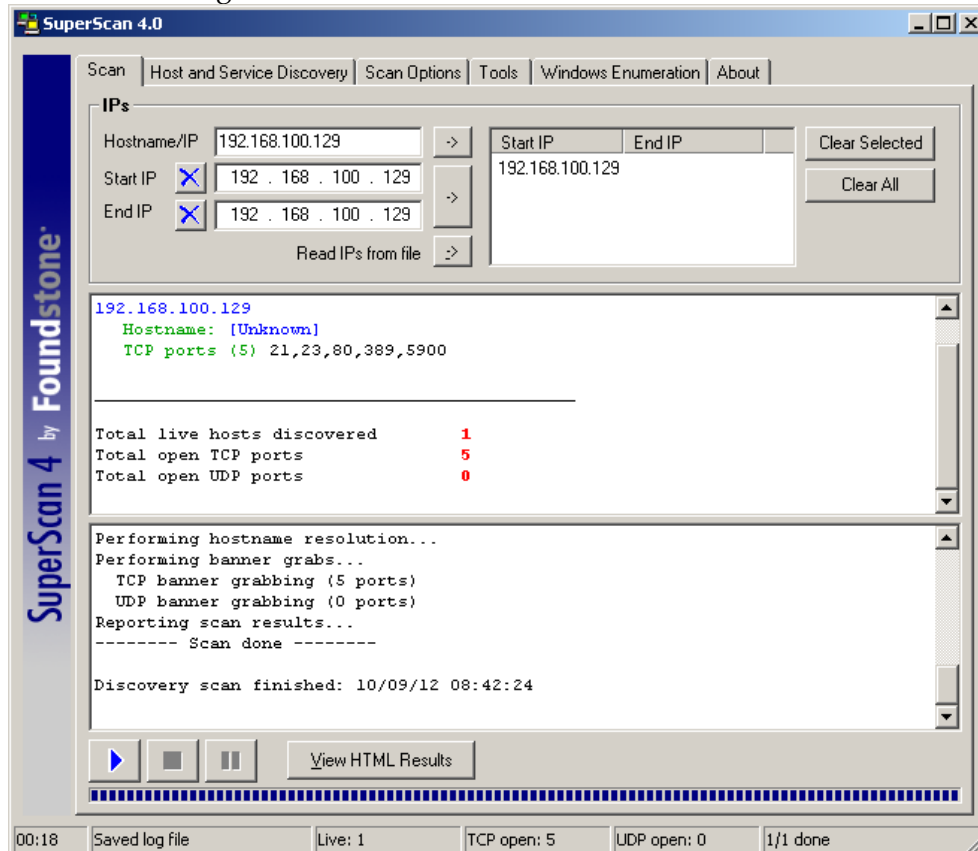
On WINDOWS2003, download the scanner from:



Superscan can be downloaded from:

<http://www.mcafee.com/uk/downloads/free-tools/superscan.aspx>

Unpack the zip file to the C:\ directory and run the executable superscan.exe. It should look like the following:



7.2.16 Detecting Network Scans using Snort IDS Pre-Processor

On UBUNTU, edit the detection rules file, and add the pre-processor rule for network scanning detection from the WINDOWS2003 server Snort sensor rules file:

Test the Preprocessor

Delete everything in the log folder using:

```
napier@ubuntu:~/snort/log$ sudo rm -r *
```

Then run Snort, while monitoring the output log directory as before.

Generate scan traffic to test the rule using Nmap, while monitoring the log file for alerts

with:

```
sudo tail -f alert
```

And open another terminal window and monitor the scanning log file with:

```
sudo tail -f portscan.log
```

From WINDOWS2003, run a default host discovery/port scan against UBUNTU, using the nmap network scanner.

Q: Did Snort detect the port scan?

YES/NO

The alert should be similar to the following:

```
napier@ubuntu:~/snort/log$ sudo tail -f portscan.log
[sudo] password for napier:
^CSorry, try again.
[sudo] password for napier:
sudo: 1 incorrect password attempt
napier@ubuntu:~/snort/log$
napier@ubuntu:~/snort/log$ tail -f alert
[Priority: 0]
10/08-16:04:54.603204 0:C:29:54:57:AA -> 0:C:29:3E:8A:FE type:0x800 len:0x62
192.168.100.133 -> 192.168.100.129 ICMP TTL:128 TOS:0x0 ID:9674 IpLen:20 DgmLen:
84 DF
Type:0 Code:0 ID:57099 Seq:3 ECHO REPLY

[**] [122:7:0] (portscan) TCP Filtered Portsweep [**]
[Priority: 3]
10/08-16:18:24.660949 4D:41:43:44:41:44 -> 4D:41:43:44:41:44 type:0x800 len:0xB1
192.168.100.133 -> 199.59.148.89 PROTO:255 TTL:0 TOS:0x0 ID:10344 IpLen:20 DgmLe
n:163 DF
```