



UNIVERSITEIT VAN PRETORIA  
UNIVERSITY OF PRETORIA  
YUNIBESITHI YA PRETORIA  
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

# **A Digital Forensic Readiness Approach for Ransomware Forensics**

**By**

**Avinash Singh**

Submitted in fulfilment in accordance with the requirements for the degree of

**MASTER OF SCIENCE (COMPUTER SCIENCE)**

in the  
**Faculty of Engineering, Built-Environment and Information Technology**

at the  
**UNIVERSITY OF PRETORIA**

**SUPERVISOR:**  
**Prof. H.S. Venter**

**CO-SUPERVISOR:**  
**Dr A.R. Ikuesan**

**November 2019**

## Declaration

I, Avinash Singh, hereby declare that this dissertation, “*A Digital Forensic Readiness Approach for Ransomware Forensics*” is submitted in accordance with the requirements for the Master of Science in Computer Science at the University of Pretoria, is my own original work and has not previously been submitted to any other institution of higher learning. All sources cited or quoted in this research paper are indicated and acknowledged with a comprehensive list of references. One international journal and two international conferences were published from this dissertation.

November 2019

“The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards.”

— Gene Spafford

“The more you know, the more you realize you know nothing.”

— Socrates

“Programming can be fun, so can cryptography; however they should not be combined.”

— Kreitzberg and Shneiderman

## Acknowledgements

To have achieved this milestone in my life, I would like to express my sincere gratitude to the following people:

- My revered Gods (Saraswathi, Hanuman and Shiva), who provided me with the strength, knowledge and perseverance to complete this research;
- Dr. Adeyemi R. Ikuesan and Prof. Hein S. Venter, my supervisors, for their invaluable advice, supervision and inspiring motivation during trying times throughout this research. For this I am eternally grateful;
- Language Editor, Issabel Classen for a great language editing experience;
- The staff in Department of Computer Science, for guiding me through my academic career, especially Mr Stallmann, Prof. Pillay, Ms Barror;
- The members from my research group DigiForS, I thank you all – Stacey, Kofi, Werner, Pierre, Ivans and Victor for all the input and encouragement that each of you has given me;
- Without the support of my family this would have never been possible, and I am blessed to have such loving and caring people in my life. Meena (Mom), Mothie (Dad), Asmita (Sister) and Princess (Doggie Sister) thank you all for everything you have done, you have no idea how important each of you are in my life;
- My colleagues – George, Pula and Frederick for brainstorming and sharing ideas;
- A special thanks to the ASN research group and their CCI program for providing me with some financial assistance.
- Last, but not the least, I would like to thank the Universal Forces and anybody whom I have left out mentioning.

## Abstract

Computers play a vital role in the automation of tedious tasks in our everyday lives. With the adoption of the advances in technology, there is a significant increase in the exploitation of security vulnerabilities, particularly in Windows computing environments. These exploitations are mostly carried out by malicious software (malware). Ransomware, a variant of malware which encrypts user files and retains the decryption key for ransom. Ransomware has shown its dominance over the years wreaking havoc to many organizations and users. This global digital epidemic is continuously on the rise with no signs of being eradicated. The current method of mitigation and propagation of malware and its variants, such as anti-viruses, have proven ineffective against most ransomware attacks. Theoretically, Ransomware retains footprints of the attack process in the Windows Registry as well as volatile memory of the infected machine. With the adoption of Digital Forensic Readiness (DFR) processes organizations can better prepare for these types of attacks. DFR provides mechanisms for pro-active collection of digital artifacts. These artifacts play a vital role when a digital investigation is conducted where these artifacts may not be available post-incident. The availability of such artifacts can be attributed to the anti-forensic properties of the ransomware itself cleaning up all the evidence before it can be investigated. Ransomware investigation often to a lengthy process because security researchers need to disassemble and reverse engineer the ransomware in order to find a inherit flaw in the malware. In some cases, the ransomware is not available post-incident which makes it more difficult. Therefore, study proposed a framework with the integration of DFR mechanisms as a process to mitigate ransomware attacks whilst maximizing Potential Digital Evidence (PDE) collection. The proposed framework was evaluated in compliance with the ISO/IEC 27043 standard as well as expert review using two prototype tools. These prototype tools realize the framework by providing a proof of concept implementation of such a framework within an organization. The evaluation revealed that the proposed framework has the potential to harness system information prior to, and during a ransomware attack. This information can then be used to help forensic investigators to potentially decrypt the encrypted machine, as well as providing automated analysis of the ransomware relieving the burden of complicated analysis. The implementation of the proposed framework can potentially be a major breakthrough in mitigating this global digital endemic that has plagued various organizations.

## Declaration from Language Editor

### DECLARATION

I herewith declare that I,  
**Isabel M Claassen** (APSTrans (SATI)),  
full-time freelance translator, editor and language consultant  
of  
1367 Lawson Avenue, Waverley, Pretoria  
(cell 082 701 7922)  
and  
accredited member (No. 1000583) of the South African Translators' Institute (SATI)  
completed the language editing\* of the dissertation entitled

### **A Digital Forensic Readiness Approach for Ransomware Forensics**

submitted for the degree  
**Master of Science - Computer Science**  
in the  
Faculty of Engineering, Built-Environment and Information Technology  
University of Pretoria  
by  
**Avinash Singh**

E-mail: [asingh@cs.up.ac.za](mailto:asingh@cs.up.ac.za)

Date completed: 03-12-2019

***\*Please note that no responsibility can be taken for the veracity of statements or arguments in the document concerned or for changes made subsequent to the completion of language editing. Also remember that content editing is not part of a language editor's task and is in fact unethical.***

## List of abbreviations

PC	Personal Computer
MPCU	Model of PC Utilisation
GUI	Graphical User Interface
CAT	Context-Aware Trigger
NIST	National Institute of Standards and Technology
PE	Portable Executable
OS	Operating System
DFI	Digital Forensic Investigator
DFR	Digital Forensic Readiness
POC	Proof of Concept
DFRWS	Digital Forensic Research Workshop
PKI	Public Key Infrastructure
RSA	Rivest Shamir Adleman
SSL	Secure Socket Layer
MD	Medical Doctor
SHA	Secure Hash Algorithm
RAID	Redundant Array of Inexpensive Disks
IP	Internet Protocol
DDOS	Distributed Denial of Service
VOIP	Voice over Internet Protocol
PUP	Potentially Unwanted Product
XSS	Cross-Site Scripting
DNS	Domain Name Service
DDNS	Dynamic Domain Name System
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
NSA	National Security Agency
SMB	Server Message Block
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
MBR	Master Boot Record
AES	Advanced Encryption Standard
VM	Virtual Machine
MFA	Multi-Factor Authentication
RFF	Ransomware Readiness Framework
EM	Entropy Monitoring
CFTT	Computer Forensic Tool Testing
MVC	Model View Controller
REST	REpresentational State Transfer
TOTP	Time-base One-Time Password
AM	API Monitoring
RM	Registry Monitoring
VPS	Virtual Private Server
SOC	Security Operations Center

# Table of Contents

## PART I 1

<b>1.</b>	<b>CHAPTER 1: INTRODUCTION.....</b>	<b>2</b>
1.1	Introduction.....	2
1.2	Problem statement.....	4
1.3	Limitations of the research .....	5
1.4	Goals and objectives .....	6
1.5	Motivation.....	7
1.6	Methodology.....	7
1.7	Layout.....	8

## PART II ..... 12

<b>2.</b>	<b>CHAPTER 2: DIGITAL FORENSIC SCIENCE .....</b>	<b>13</b>
2.1	Introduction.....	13
2.2	Forensic science.....	13
2.2.1	Digital forensic science.....	14
2.2.2	Cyber forensic services .....	14
2.2.2.1	Confidentiality .....	15
2.2.2.2	Integrity.....	16
2.2.2.3	Availability.....	16
2.2.2.4	Authentication.....	17
2.2.2.5	Authorization.....	17
2.2.2.6	Non-repudiation .....	17
2.3	Digital forensic investigation .....	17
2.3.1	Digital investigation lifecycle .....	18
2.3.1.1	Planning.....	18
2.3.1.2	Acquisition .....	19
2.3.1.3	Preservation.....	19
2.3.1.4	Analysis .....	19
2.3.1.5	Reporting and dissemination .....	20
2.3.2	Digital investigation tools .....	20
2.4	Digital forensic readiness.....	21
2.4.1	Digital forensic readiness processes .....	22
2.5	Conclusion .....	25
<b>3.</b>	<b>CHAPTER 3: MALWARE FORENSICS.....</b>	<b>26</b>
3.1	Introduction.....	26



<b>3.2</b>	<b>An overview of malware .....</b>	<b>29</b>
3.2.1	Types of malware.....	29
3.2.2	Method of propagation.....	32
3.2.3	Adaptive techniques used by malware .....	33
<b>3.3</b>	<b>Malware analysis.....</b>	<b>34</b>
3.3.1	Reverse engineering .....	34
3.3.2	Controlled environment .....	35
3.3.3	Signatures .....	36
3.3.4	Static analysis.....	36
3.3.5	Dynamic analysis.....	36
3.3.6	Exploitation techniques .....	37
3.3.7	Obfuscation .....	37
3.3.8	Encryption methods .....	37
3.3.9	Communication protocols .....	38
3.3.10	Attribution.....	38
3.3.11	Categorisation.....	38
3.3.12	Memory analysis.....	39
<b>3.4</b>	<b>Ransomware.....</b>	<b>39</b>
<b>3.5</b>	<b>Conclusion .....</b>	<b>43</b>

**PART III..... 44**

**4. CHAPTER 4: RANSOMWARE READINESS FRAMEWORK ..... 45**

<b>4.1</b>	<b>Introduction.....</b>	<b>45</b>
<b>4.2</b>	<b>Ransomware Readiness Framework (RRF).....</b>	<b>45</b>
4.2.1	<b>Identification.....</b>	<b>46</b>
4.2.1.1	Network.....	46
4.2.1.2	Computing devices .....	48
4.2.1.3	Operating Systems .....	48
4.2.1.4	Evidence sources .....	48
4.2.1.4.1	Memory.....	48
4.2.1.4.2	Registry.....	49
4.2.1.4.3	Storage media .....	50
<b>4.3</b>	<b>Collection.....</b>	<b>50</b>
4.3.1	<b>Dynamic information .....</b>	<b>52</b>
4.3.1.1	Potential evidence collection from memory.....	52
4.3.1.2	Process memory.....	53
4.3.1.3	Registry.....	54
4.3.1.4	Network.....	54
4.3.2	<b>Static information .....</b>	<b>55</b>
<b>4.4</b>	<b>Secure storage .....</b>	<b>55</b>
4.4.1	<b>Security.....</b>	<b>56</b>

4.4.1.1	Two-factor authentication.....	56
4.4.1.2	Sandboxing.....	56
4.4.1.3	Access control .....	57
4.4.2	<b>Data .....</b>	<b>57</b>
4.5	<b>Conclusion .....</b>	<b>57</b>
<b>5.</b>	<b>CHAPTER 5: WINDOWS REGISTRY AND RAM COLLECTOR (W2RC)</b> .....	<b>60</b>
5.1	<b>Introduction .....</b>	<b>60</b>
5.2	<b>Windows Registry and RAM Collector (W2RC).....</b>	<b>60</b>
5.2.1	DLL monitoring .....	63
5.2.2	API call monitoring .....	65
5.2.3	Windows Registry monitoring.....	66
5.2.4	Entropy monitoring .....	67
5.2.5	Context-Aware Trigger .....	67
5.2.6	Investigative procedure .....	68
5.3	<b>W2RC system requirements specification .....</b>	<b>68</b>
5.3.1	Secure Collection Core Requirements (SC-CR) .....	69
5.3.2	Secure Collection Optional Requirements (SC-OR) .....	69
5.4	<b>Architectural design .....</b>	<b>70</b>
5.5	<b>W2RC system implementation .....</b>	<b>72</b>
5.6	<b>Conclusion .....</b>	<b>75</b>
<b>6.</b>	<b>CHAPTER 6: WINDOWS REGISTRY AND RAM READINESS STORAGE (W3RS)</b> .....	<b>76</b>
6.1	<b>Introduction .....</b>	<b>76</b>
6.2	<b>Windows Registry and RAM Readiness Storage (W3RS) .....</b>	<b>76</b>
6.2.1	Data ingestion process .....	77
6.2.2	Forensic soundness assurance process .....	79
6.2.3	PDE storage process.....	80
6.2.4	Forensic soundness verification process.....	81
6.3	<b>W3RS system requirements specification .....</b>	<b>84</b>
6.3.1	Secure Storage Core Requirements (SS-CR) .....	84
6.3.2	Secure Storage Optional Requirements (SS-OR).....	84
6.4	<b>W3RS system implementation .....</b>	<b>84</b>
6.5	<b>Conclusion .....</b>	<b>91</b>
	<b>PART IV .....</b>	<b>92</b>

<b>7.</b>	<b>CHAPTER 7: RESULTS AND INTERPRETATION OF THE PROTOTYPE SYSTEM.....</b>	<b>93</b>
7.1	Introduction.....	93
7.2	Results obtained from the testing phase .....	93
7.2.1	DLL monitoring results .....	94
7.2.2	API monitoring results .....	95
7.2.3	Registry monitoring results.....	96
7.2.4	Entropy monitoring results.....	98
7.2.5	Context-Aware Trigger results .....	101
7.3	Conclusion .....	103
<b>8.</b>	<b>CHAPTER 8: REAL-WORLD CASE STUDIES .....</b>	<b>104</b>
8.1	Introduction.....	104
8.2	Real-world case studies .....	104
8.2.1	Case study 1: WannaCry.....	104
8.2.1.1	WannaCry scenario definition .....	105
8.2.1.2	WannaCry detection and results.....	105
8.2.1.3	WannaCry discussion .....	106
8.2.2	Case study 2: Dharma .....	107
8.2.2.1	Dharma scenario definition .....	107
8.2.2.2	Dharma detection and results .....	108
8.2.2.3	Dharma discussion .....	108
8.2.3	Case study 3: RobbinHood.....	109
8.2.3.1	RobbinHood scenario definition .....	109
8.2.3.2	RobbinHood detection and results.....	110
8.2.3.3	RobbinHood discussion .....	110
8.3	Conclusion .....	111
<b>9.</b>	<b>CHAPTER 9: CRITICAL EVALUATION .....</b>	<b>112</b>
9.1	Introduction.....	112
9.2	Software verification and validation process.....	112
9.2.1	Secure collection validation .....	113
9.2.1.1	Secure Collection Core Test Assertions (SC-CA).....	114
9.2.1.2	Secure Collection Test Cases (SC-TC) .....	114
9.2.1.3	Secure Collection Compliance Matrix (SCCM) .....	115
9.2.2	Secure storage validation .....	117
9.2.2.1	Secure Storage Core Test Assertions (SS-CA) .....	117
9.2.2.2	Secure Storage Test Cases (SC-CA) .....	117
9.2.2.3	Secure Storage Compliance Matrix (SSCM).....	117
9.3	Expert review process .....	119
9.3.1	Respondent identification.....	120
9.3.2	Measurement item development.....	121
9.3.3	Data analysis and presentation.....	122

9.3.4	<b>Results of the model of PC utilisation .....</b>	<b>123</b>
9.3.4.1	MPCU complexity .....	123
9.3.4.2	MPCU job fitness .....	124
9.3.4.3	MPCU long-term consequence.....	125
9.4	<b>Mapping of the proposition to a Digital Forensic Standard .....</b>	<b>127</b>
9.4.1	<b>Mapping of the proposed framework to ISO/IEC 27043:2015 ...</b>	<b>127</b>
9.5	<b>Related literature.....</b>	<b>128</b>
9.5.1	<b>Digital Forensic Readiness.....</b>	<b>130</b>
9.5.2	<b>Ransomware investigation .....</b>	<b>131</b>
9.5.3	<b>Summary of the findings from related literature .....</b>	<b>131</b>
9.6	<b>Conclusion .....</b>	<b>132</b>
<b>PART V .....</b>		<b>134</b>
10.	<b>CHAPTER 10: CONCLUSION .....</b>	<b>135</b>
10.1	<b>Summary of chapters.....</b>	<b>135</b>
10.2	<b>Addressing the problem statement .....</b>	<b>136</b>
10.3	<b>Contributions made by the current research .....</b>	<b>137</b>
10.4	<b>Limitations of this research.....</b>	<b>137</b>
10.5	<b>Future work .....</b>	<b>138</b>
10.6	<b>Final words .....</b>	<b>139</b>
<b>PART VI .....</b>		<b>140</b>
<b>Appendix A .....</b>		<b>141</b>
<b>Appendix B .....</b>		<b>142</b>
A.	<b>Setting up the environment.....</b>	<b>142</b>
B.	<b>Setting up Virtual Box.....</b>	<b>142</b>
C.	<b>Setting up Cuckoo Framework.....</b>	<b>145</b>
D.	<b>Setting up Storage (W3RS).....</b>	<b>148</b>
E.	<b>Setting up Collection (W2RC).....</b>	<b>150</b>
<b>Appendix C .....</b>		<b>153</b>
<b>Bibliography .....</b>		<b>159</b>

## List of Figures

Figure 1-1. The layout of the dissertation .....	11
Figure 2-1. Investigation lifecycle .....	18
Figure 2-2. Digital forensic investigation process model.....	22
Figure 2-3. Readiness processes groups.....	23
Figure 2-4. DFR planning processes group.....	24
Figure 2-5. DFR implementation process group.....	25
Figure 2-6. DFR assessment process group .....	25
Figure 3-1. Microsoft advanced threat analytics infographic [65].....	27
Figure 3-2. A decade's statistics of malware trends [67] .....	27
Figure 3-3. Signature-based detection .....	28
Figure 3-4. Behaviour-based detection .....	29
Figure 3-5. Ransomware statistics compiled by 24BY7 Security.....	40
Figure 4-1. A high-level overview of RRF.....	45
Figure 4-2. Part A - Overview of identification phase .....	46
Figure 4-3. Client-server architecture.....	47
Figure 4-4. Peer-to-Peer architecture.....	47
Figure 4-5. Common network topologies.....	47
Figure 4-6. Memory structure.....	49
Figure 4-7. Part B – Overview of the collection phase.....	51
Figure 4-8. Part C - Overview of Secure Storage Phase .....	56
Figure 4-9. Second layer high-level view of RRF .....	59
Figure 5-1. RRF mapped to the prototype system.....	60
Figure 5-2. Model for ransomware forensics .....	61
Figure 5-3. Architectural design of the proposed system.....	71
Figure 5-4. Lifecycle of W2RC .....	73
Figure 5-5. W2RC GUI view.....	74
Figure 5-6. W2RC new process detected.....	74
Figure 5-7. W2RC displaying analysed sample with CAT value .....	74
Figure 6-1. High-level process model of W3RS.....	77
Figure 6-2. W3RS data ingestion process.....	78
Figure 6-3. W3RS forensic soundness assurance process .....	79
Figure 6-4. W3RS PDE storage process.....	80
Figure 6-5. W3RS forensic soundness verification process.....	81
Figure 6-6. W3RS detailed process model.....	83
Figure 6-7. High-level lifecycle of W3RS .....	86

Figure 6-8. W3RS user password standards.....	87
Figure 6-9. W3RS user permissions assignment .....	88
Figure 6-10. W3RS adding 2FA .....	88
Figure 6-11. W3RS user interface.....	89
Figure 6-12. W3RS redacted view of stored PDE .....	89
Figure 6-13. PDE sample showing process memory.....	90
Figure 6-14. PDE sample showing network activity.....	90
Figure 9-1. NIST validation cycle .....	113
Figure 9-2. MPCU complexity graph .....	124
Figure 9-3. MPCU job fitness graph.....	125
Figure 9-4. MPCU Long-term Consequence .....	126
Figure 9-5. Mapping of the Ransomware Readiness Framework to the ISO/IEC 27043:2015 International Standard.....	128
Figure 0-1. PDE snippet showing detected signature.....	153
Figure 0-2. PDE snippet showing delay operations.....	153
Figure 0-3. PDE snippet showing PE sections and entropy.....	154
Figure 0-4. PDE snippet showing process memory.....	154
Figure 0-5. PDE snippet showing buffer information location .....	155
Figure 0-6. PDE snippet showing network activity.....	155
Figure 0-7. PDE snippet showing loaded DLLs .....	156
Figure 0-8. PDE snippet showing cryptographic key information.....	156
Figure 0-9. PDE snippet showing API calls .....	157
Figure 0-10. PDE snippet showing various signatures .....	158

## List of Tables

Table 2-1. A taxonomy of forensic science.....	13
Table 2.2. Comparison of digital forensic tools.....	20
Table 3.1. A summary of trending ransomware.....	41
Table 5-1. DLL cryptographic commonalities for ransomware.....	64
Table 7-1. DLL monitoring results .....	94
Table 7-2. API monitoring results .....	95
Table 7-3. Windows Registry monitoring results .....	97
Table 7-4. Authentic vs non-authentic encryption using entropy analysis.....	99
Table 7-5. Entropy monitoring results .....	100
Table 7-6. Experimental results from well-known applications and ransomware .....	101
Table 7-7. A Summary of thresholding values.....	102
Table 8-1. WannaCry sample information.....	105
Table 8-2. WannaCry case study.....	106
Table 8-3. Dharma sample information .....	108
Table 8-4. Dharma case study .....	108
Table 8-5. RobbinHood sample information .....	110
Table 8-6. RobbinHood case study.....	110
Table 9-1. W2RC SCCM.....	115
Table 9-2. W3RS SSCM.....	118
Table 9-3. Measurement instrument for tool evaluation.....	121
Table 9-4. Response statistics of MPCU complexity .....	123
Table 9-5. Response statistics of MPCU job fitness.....	125
Table 9-6. Response statistics of MPCU long-term consequence.....	126
Table 9-7. Summary of related literature findings (2019/11/28).....	132
Table 10-1. Limitations of this research .....	137

# PART I

## INTRODUCTION



# 1. CHAPTER 1: INTRODUCTION

## 1.1 Introduction

With the advancements in technology and easy access to the internet, more users are moving towards the digital world. Smart devices such as smartphones, tablets, laptops and desktop computers are bringing more interconnectedness to everyday human life. By using these devices, more people are creating and sharing information, thus opening the landscape to security attacks [1]. Hackers exploit the vastness of the internet by exploiting unpatched vulnerabilities and inexperienced and over-trustworthy users. While technology is enhancing life and making the internet more accessible, many people are still not familiar with the risks introduced by these technologies. For example, mobile devices have become an integral part of our life, and many people cannot go without them. People store most of their essential information on these devices, such as passwords, emails, sensitive data, and even social media information. With all this information kept in one place, attackers are gifted a central point and a huge incentive to attack and extract private information about individuals – even up to the extent of being able to impersonate them (pretending to be another person) [2].

Malware (Malicious Software) is the common term used for a piece of program code that is used to cause harm [3]. Attackers use malware to exploit vulnerabilities that exist on electronic devices. These vulnerabilities come from incorrect configurations, security flaws in applications and improper or lousy use of code in applications. Cybersecurity is a field that combines the digital world with security to protect systems against outside threats. Effective cybersecurity reduces the risk of cybercrime by finding flaws and vulnerabilities in the cyberspace and releasing patches to prevent hackers from exploiting these vulnerabilities. Research is still ongoing in this area where new challenges and problems surface every day, thereby widening the gap and attack vectors of malicious activities [4]–[6].

Ransomware is one of the most dangerous forms of malware that encrypts user data on a system, leaving many users and organisations crippled by its effects. This form of malware can spread rapidly over a network to render the system inaccessible and the end-user helpless. However, there are two types of ransomware, namely Locker and Crypto. Locker ransomware aims to make an operating system inaccessible until the ransom has been paid. Crypto ransomware is the most common form of ransomware today. It uses strong encryption to encrypt files on a system and then withholds the decryption keys of these files for a ransom. Such ransom is usually demanded in untraceable currency, with cryptocurrency such as Bitcoin being the most prevalent. Ransomware commonly occurs in the Windows Operating System (OS) due to the exploitation of unpatched vulnerabilities and the system's large user base [7]. Based on recently reported events [8], this trend is gradually seeing a drift towards Android devices, particularly mobile phones.

This drift is further prompted by the ubiquitous nature, sensitive contents and attached importance of mobile devices, which compels an individual to urgently pay such ransom. Till date, Anti-Virus (AV) software and tools that are used to detect malicious software have proven to be ineffective to detect new variants of ransomware [9]. This ineffectiveness can be attributed to the limitations of the signature-based detection approach, as well as the mitigation techniques employed by sophisticated ransomware [4]. However, the point of attack and methods employed in the exploitation caused by this ransomware malware can potentially be uncovered using Digital Forensics and investigation processes.

Digital forensics involves the recovery and investigation of data acquired from digital devices related to computer crime [10][11]. Encrypted devices pose a significant challenge to digital forensics due to the difficulty of retrieving potential evidential information for litigation [12]. In digital forensics, the use of a cryptographic mechanism such as BitLocker as well as advanced encryption standards to protect the system/information poses a significant problem for an investigator. If a drive has been encrypted, an investigator would need the decryption keys to investigate the drive. Most of the time, however, the decryption keys are unknown, and an investigator would have to use a brute-force approach to decrypt the drive and perform an investigation. The Windows Operating System (OS), being the most widely used OS [7] [13] is a central target for attackers who exploit the vulnerabilities of each version of the OS. Therefore, to investigate a ransomware attack, it is often difficult for a Digital Forensic Investigator (DFI) to recover the system from the attack as well as to find any potential digital evidence that can be used in a court of law. However, upon investigation, the method of exploitation can be found by dissecting the ransomware executable on a lower level, which involves tracing the execution of the program and monitoring the changes in the behaviour of each instance [14] [15].

Since the use of ransomware is so widespread, it is almost impossible to trace the source. However, it is possible to pinpoint the country in which it was first reported. Research has been done to trace the payment endpoints by tracking cryptocurrency addresses such as Bitcoin wallets. So far, this is the only method that provides some information on where the ransom money is flowing to [16] [17]. The process of a traditional investigation would involve an incident to occur and be reported so that an investigation can be triggered. Unfortunately this is a manual process that can be delayed by several unforeseen factors [18]. The data that is needed by an investigator might also not have been collected, which further delays the investigation and renders the entire process slightly inefficient.

The remainder of this chapter is structured as follows: First, the problems addressed by this research are identified and defined, followed by a discussion of the limitations

of the research, the research goals and objectives, the motivation for this research and lastly the layout and structure of the dissertation.

## **1.2 Problem statement**

Cybercriminals are frightfully active in the digital world. Most attackers gain access through leaked or unchanged administrator passwords and even through social engineering or spear-phishing attacks. Social engineering is one of the common tactics used by attackers to exploit the ignorance of a user – the latter is tricked into divulging sensitive information such as bank account details and passwords. Malware on the other hand adopt a more advanced technical approach by using malicious code to perform criminal activities. More dangerous than ever is ransomware that relies on the widespread distribution of the malware. This distribution can happen as a result of an array of causes and effects, such as infected Microsoft Office files, unpatched vulnerable systems, unsolicited emails, and poor user education.

Most ransomware leaves traces/footprints on the machine, particularly in the Windows Registry, which could provide an evidentiary source for mitigation and litigation of such attacks. Furthermore, this potential evidence source can be incorporated into the forensic process required to trace the propagation path and method of the ransomware. This could help to classify ransomware to find probable behavioural consistencies (also referred to as the behavioural signature). However, the current investigation process for ransomware forensics (the forensic domain saddled with investigating ransomware incidents) has primarily neglected to leverage the potential of the Windows Registry in combination with volatile memory. In addition, the process of corroborating evidence from the Windows Registry using data from the Random-Access Memory (RAM) has been widely overlooked [19]–[21]. This lapse can be attributed to the unavailability of RAM information upon investigation, and the complexity of extracting evidential information from the Windows Registry and RAM [19] [22] [23].

To address some of these challenges faced during investigation and detection, this dissertation breaks down the main research problem into subproblems, where a more robust solution can be used for each of the challenges. Together, they can then provide a complete solution to the main research problem. The subproblems are further distinguished as follows:

- There is no framework, model or standard for collecting potential digital evidence (PDE) for ransomware forensics. Most investigators do not have a standard process to follow when investigating ransomware. Attempts to uncover any evidence may potentially inadvertently destroy the evidence. This can be due to negligence, for example, putting the computer off instead of capturing the volatile memory, or working on the system, which affects the

memory without a prior memory dump, can invalidate the potential digital evidence, as it will legally be seen as tampering with evidence.

- The use of Digital Forensic Readiness (DFR) in malware investigation, specifically *ransomware forensics*, has been overlooked [24]–[26] in many organisations and has therefore, induced a higher cost of incident response. Furthermore, incident response elicits excessive waiting times for analysing and uncovering any corroborating evidence.
- Ransomware continues to plague the internet by rendering systems unusable through the encryption of user/company data for ransom. Consequently, it causes business downtime, delayed operations and (eventually) considerable costs to be incurred by the user/organisation.
- To the best of the author’s knowledge, there are no automated processes for the detection and investigation of ransomware. Static analysis of malicious samples has been ineffective in detecting newer variants and types of ransomware attacks. Given that automation has significantly improved the way we live and perform tasks, neglecting the effectiveness of using automation in a ransomware investigation indeed constitutes a research gap in the digital forensic community.

Each of the problems listed above can be converted into short research questions. Therefore, the current research attempted to answer the following questions.

- Q1)** To what degree can a framework/model be created to aid digital forensic investigators to perform a ransomware investigation?
- Q2)** What potential digital evidence can be collected from a ransomware attack using Digital Forensic Readiness?
- Q3)** Can such a framework reduce costs and improve incident response?
- Q4)** To what degree can ransomware be detected before it causes permanent damage?
- Q5)** Is there a way to automate the digital forensic process for investigating ransomware?

Some aspects of the above research questions go beyond the scope of this dissertation. These limitations are presented in the next section.

### **1.3 Limitations of the research**

The following are the restrictions and limitations of this research, as the aspects listed have not been included in the scope of this research.

- The impact of human behaviour has not been fully incorporated into this study. This means that any user-temperament or user-imposed restrictions to the prototype tools were not considered in their development. Due to negligence and poor user education, a significant issue that affects the security of a system is the human factor. Man is often the weakest part of any

system, which is why this research attempts to automate this process as much as possible.

- Virtual environments used in this study were assumed to be safe and secure; any current/future exploits or vulnerabilities exposed by these environments were not considered.
- The prototype tools developed relied on correct setup, and configurations with the results obtained extracted from ideal environments. It was assumed that every service and process are running as intended; otherwise, an unexpected conclusion might be reached.
- Industry standards were used for securing and extracting information, and this study did not attempt to improve or reinvent these standards.

With these limitations laid out, the goals and objectives of this research are presented in the next section.

#### **1.4 Goals and objectives**

The main goal of this research was to create a system that can detect, prevent and potentially recover from ransomware attacks. This system has to help users and organisations to gather critical information that can be used to quickly and cost-effectively conduct forensic investigations. Forensic investigators can use the system to trace and investigate the detected anomalies, based on the information that has been collected from the time when the incident was detected. In addition, the research in hand has the following objectives:

- Review the best practices and techniques and improve the current state of research by improving the collection, preservation, and investigation of PDE.
- Ensure the forensic soundness of the PDE collected from volatile memory and Windows registry – from the point of collection to the point of investigation. This is extremely important for the authenticity of the PDE collected.
- Take advantage of the massive data repositories that exist in the Windows computing environment to leverage more PDE, and to collate and corroborate the collected information.
- Design a conceptual model to address the limitations of existing literature, solve the primary goal of this research, and reduce the costs of litigation and investigation.
- Evaluate and apply real-world, use-case scenarios to verify the viability and usefulness of the prototype.

The next section outlines the motivation as to why this research was conducted and why it is relevant.

## 1.5 Motivation

Ransomware is on the rise, with trends predicting that the problem will not go away anytime soon [27]–[29]. Therefore, novel and effective ways need to be researched on how to mitigate ransomware attacks, as well as how to potentially trace the origin, prevent attacks and gather information when an attack has been detected. AV programs are continually becoming more sophisticated, but unfortunately, they cannot keep up with the rate of exploits and threats. Having an effective AV program running on a system does not mean that the system is secure and not susceptible to exploits and attacks. The rate at which ransomware spreads is faster than the rate at which AV programs can develop updates to prevent the exploit; thus, ransomware can do enough damage to cause havoc across organisations. Incorporating DFR processes within an organisation better prepares the organisation for such attacks, as is gathering information and potentially preventing the attacks from occurring, while improving incident response times. In order to properly implement these processes within an organisation, an appropriate research methodology needs to be formulated.

The research methodology presented in the next section explains how this research was planned and carried out.

## 1.6 Methodology

The research methodology entails the method of scientific investigation that is used to solve the problem on hand in such a way that the study can be reproduced and achieve the same result. The current research adopted a systematic approach to solve the research problem and the following scientific methods were used:

- (i) Conducting a literature review by exploring various related and current literature – extending from malware to ransomware – as well as the background of digital forensics and digital forensic investigations.
- (ii) Discovering existing standards that have or could have been used to further structure and create a standardised viable solution. Reviewing the digital forensic readiness process models stemming from the ISO/IEC 27043 [30] and how they can be applied.
- (iii) Conceptualising a model to verify and illustrate the functioning of the proposed prototype system. Looking into potential digital evidence collection and preservation through digital forensic soundness in order for PDE to be admissible in a court of law.
- (iv) Developing the prototype systems while abiding by software requirements, engineering and specifications through best practices. Finding the correct programming language to build a scalable and updateable system that can be used by organisations to demonstrate the proof of concept (POC).

- (v) Performing critical evaluation by analysing and interpreting results to further improve or correlate the workings of the system in terms of digital evidence collection, preservation and secure storage. Evaluating the usefulness of the prototype system and performing software verification, validation, as well as expert reviews. Evaluating the proposed framework by mapping it to the ISO/IEC 27043 international standard.

The above methods were incorporated to find a viable solution to the problems that this research identified. The next section discusses the dissertation structure and what each chapter will entail. It also provides a visual representation of the chosen structure.

## **1.7 Layout**

The layout of this dissertation as depicted in Figure 1-1 shows that it consists of six parts and ten chapters. An overview is given of each part and chapter, after which it is further discussed:

### **PART I: Introduction**

- **Chapter 1 – Introduction**

This first chapter of the dissertation presents the introduction, problem background, limitations, goal, motivation, methodology and layout. Chapter 1 constitutes the entire Part I and lays out the scene and scope of the research.

### **PART II: Background**

- **Chapter 2 – Cybersecurity**

This chapter focuses on what cybersecurity entails, and what the fundamental challenges are that people and organisations face in the current digital age. Chapter 2 presents the challenges and shortcomings of the cyber domain as well as the difficulties thereof.

- **Chapter 3 – Digital Forensics**

The background to digital forensics is presented in this chapter, together with related literature and existing methods and frameworks that were developed to counter criminal activities. Chapter 3 also discusses the concept of digital forensic readiness, how it plays a vital role in incident response, and how an entity can easily prepare before a cyberattack can even occur.

### **PART III: Model and Prototype**

- **Chapter 4 – Ransomware Readiness Framework**

Chapter 4 presents the research framework – Ransomware Readiness Framework (RRF) – which comprises a number of subcomponents and systems that are further explained in Chapters 5 and 6. This chapter gives a high-level

overview of the research and indicates how the forensic readiness processes are incorporated into the design.

- **Chapter 5 – Windows Registry and RAM Collector (W2RC)**

This is the core component of the research monitoring the RAM and Registry. Together, Windows Registry and RAM collector (W2RC) play the vital role of monitoring all the processes by using a scientifically derived formula called Context Aware Trigger (CAT) to determine if a process is malicious.

- **Chapter 6 – Windows Registry and RAM Readiness Storage (W3RS)**

Due to digital forensic readiness requirements, potential digital evidence is collected on the fly and needs to be stored in a safe and secure manner. The system W3RS (Windows Registry and RAM Readiness Storage) securely stores the collected information by utilising access control with multifactor authentication so that no unauthorised parties can access sensitive information.

## **PART IV: Evaluation**

- **Chapter 7 – Results and Interpretation of the Prototype System**

In order to show the impact of the developed prototypes, several real-world case studies were selected and simulated in this chapter. The developed prototype tools were used to see if they are able to detect and potentially prevent a ransomware attack. The collected information was analysed to see if potential digital evidence is admissible and can hold up in a court of law.

- **Chapter 8 – Real-World Case Studies**

The prototype tools were tested in this chapter by using various malicious and benign samples. The prototype was also tested and used within organisations with surveys and reporting. Chapter 8 groups the result in various ranks or categories so that threats can be detected more easily, and automated incident response can occur.

- **Chapter 9 – Critical Evaluation**

This chapter evaluates the proposed framework based on expert reviews and benchmarking. Expert reviews were conducted to determine the usefulness of the tools, whereas benchmarking was used to validate the usefulness of the proposed framework by mapping it to an international standard.

## **PART V: Conclusion**

- **Chapter 10 – Conclusion and Future Work**

This chapter wraps up the dissertation and discusses what further research outputs can be achieved by extending on the current research. It concludes that



the research questions were answered, the developed tools are viable, and they meet the requirements that were set.

## **PART VI: Appendices**

The different appendices contain the screenshots and additional information that was not included in the core of the dissertation. They serve as a reference point for extensive reporting and raw result postings.

- **Appendix A**

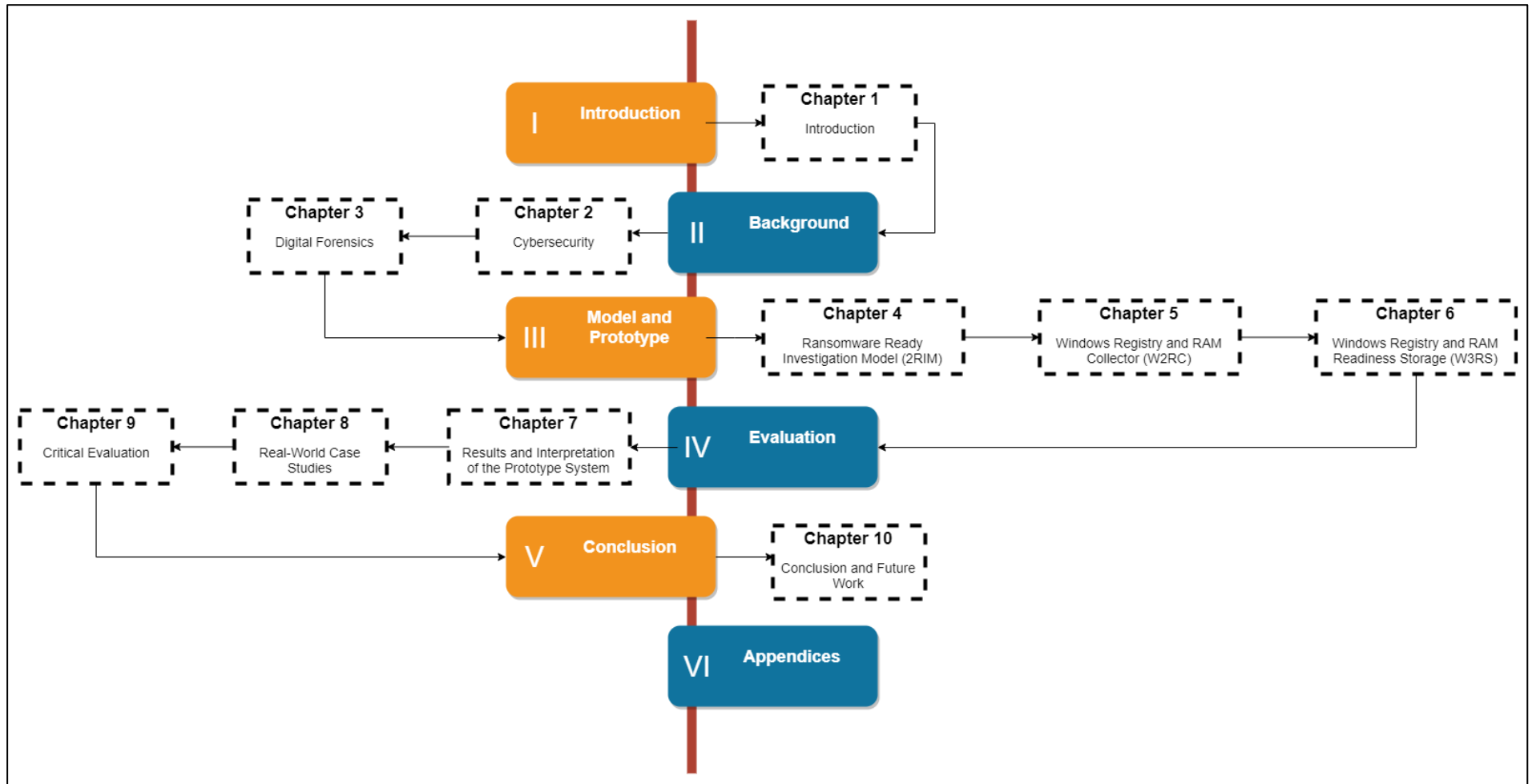
A list of publications originating from this dissertation is presented in this appendix.

- **Appendix B**

This appendix provides the installation and user guide for the prototype tools (W2RC and W3RS).

- **Appendix C**

Sample PDE snippets are presented in this appendix showing key findings and information that was collected showing the usefulness of DFR by providing an investigator with a data repository of information.



**Figure 1-1.** The layout of the dissertation

# PART II

## BACKGROUND

## 2. CHAPTER 2: DIGITAL FORENSIC SCIENCE

### 2.1 Introduction

This chapter presents a broad overview of forensic science and digital forensics, cyber forensics services, digital forensic investigations, digital forensic investigations, and digital forensic readiness. The next section provides some background information on forensics.

### 2.2 Forensic science

Forensic science is known as the application of science to law enforcement to aid the processing and investigations of criminal offences [31][32]. There are several fields within forensic science, which include anthropology, chemistry, DNA analysis, pathology and digital forensics [33]. A brief taxonomy of each field within forensic science is further discussed in Table 2-1. Forensic science investigators are specially trained professionals and follow strict procedure and protocol when collecting, preserving, analysing and storing physical evidence [30] [33]. This is to ensure that the physical artefacts are not contaminated, and the integrity of the artefacts are safeguarded and maintained. In order for the evidence to hold in a court of law, the entire process needs to be documented as well have a well-documented chain of custody (list of persons in contact with the evidence). In this dissertation, however, only the digital aspect of forensic science is considered. The next subsection discusses digital forensic sciences.

**Table 2-1.** A taxonomy of forensic science

<b>Field of forensic science</b>	<b>Description</b>	<b>Examples</b>
Anthropology	The application of anatomical science to forensics.	Identification of deceased humans based on their remains at a crime scene.
Chemistry	The application of chemistry to forensics.	Identifying illicit drugs and chemicals at a crime scene.
DNA Analysis	The process of determining DNA characteristics based on certain individuals.	Identifying an individual from the blood sample obtained from a crime scene.
Pathology	The application of medical science to forensics.	Identifying the cause of the death of a person.
Entomology	The application of studying insects to aid forensics.	Identifying and examining insects around a human remains to predict the time and location of death.

Field of forensic science	Description	Examples
Digital Forensics	The application of scientific methods and techniques in order to recover data from electronic media.	Identifying and analysing evidentiary sources to prove cyber-crime activity.

### 2.2.1 Digital forensic science

Digital forensics is a field within the scope of forensic science that focusses on the scientific investigation of cases associated with any digital media or digital devices. The Digital Forensic Research Workshop (DFRWS) in 2001, defined digital forensics as “The use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations.” [34]. From this definition, it can be asserted that there are two approaches to digital forensics; proactive and reactive. Proactive digital forensics involves the application of digital forensics before the occurrence of digital crime [35]. Potential digital evidence is collected on the fly and detects when a digital crime is being committed and triggers an investigation which is synonymous to digital forensic readiness [30]. Reactive digital forensics entails the application of digital forensics after a digital crime is committed, post-incident, which is the current traditional investigative process [32]. Reacting to a crime that has already occurred encompasses the complexity of using digital forensics to collect and gather evidence post-mortem. Digital forensics also has standardised processes and methodologies that have been tested and peer-reviewed and accepted [30]. The purpose of these processes is to ensure the same result can be obtained each time the process is repeated, thus, making it scientific. To ensure the integrity of the collected information from the processes employed, cyber forensic services needs to be followed. These cyber forensics services are presented in the proceeding subsections.

### 2.2.2 Cyber forensic services

Information security focuses on securing both digital assets within a computing system and non-digital assets, which could contain information [36]. These digital assets are the most sensitive information that needs to be secured to prevent unauthorised access. Therefore, best practices must be used, ranging from access control to encryption. Information security is built around three objectives; Confidentiality, Integrity and Availability, which commonly referred to as the CIA triad [37]. The traditional CIA triad can further be enhanced by adding more objectives to it, consequently, encapsulating cyber forensic services. Such service includes

authentication, availability and non-repudiation. Taken together, the cybersecurity services can, therefore, be defined as CIAAN, which now stands for Confidentiality, Integrity, Availability, Authentication, Authorization, and Non-repudiation. These are further discussed.

### 2.2.2.1 Confidentiality

Dealing with sensitive information with personal identifiers like identity numbers, home addresses, credit card information and cell numbers need to be kept secure and confidential. This triad focusses on the confidentiality of information [38]. Confidentiality of information can be kept by using best practices when dealing with data. Having access control will preserve confidentiality to a certain extent. However, it can still be exposed to untrustworthy users. To fully ensure confidentiality, encryption is generally used as only the person holding the decryption key has the ability to gain access to the information. There are two commonly found types of encryption, mainly symmetric and asymmetric encryption [39]. The main difference between them is that symmetric encryption only makes use of one key, meaning that one key is used for encryption, and the same key is used for decryption [39].

Asymmetric encryption makes use of two keys one key to decrypt and the other to encrypt [40] [41]. When data is being transported from a sender to a receiver, an infrastructure is needed to ensure secure key generation and distribution. This infrastructure is called PKI (Public Key Infrastructure) [42]. The Rivest–Shamir–Adleman (RSA) encryption is the most common form of encryption used today for transporting data over the internet [43]. RSA allows for secure key exchange and uses clever cryptography to ensure confidentiality. When a message is being sent is encrypted using the receivers public key. Thus, only the receivers private key can decrypt the encrypted message, ensuring non-repudiation (non-deniability of receipt). Ransomware abuses this triad by using confidentiality against a user. This is done by encrypting user files and holding these files decryption as a ransom [44]. In the developed tools, potential digital evidence is confidentiality kept using secure channels like SSL for transport, access control, and two-factor authentication. While having encryption securing digital evidence, it preserves confidentiality but causes overhead in the investigation. This is because the evidence or acquired device would have to be decrypted first in order to perform any analysis, making it a lengthy process [45] [46].

In digital forensics, confidentiality is maintained through restricted access and non-disclosure agreements. Due to sensitive information being collected and analysed by investigators, confidentiality is key in order not to defame a person or criminal and to protect user privacy. The next subsection discusses what integrity is and how it can be maintained.

### **2.2.2.2 Integrity**

The integrity of digital data is a measure of the authenticity and originality of the data [37]. Integrity is used to determine if the data was modified or tampered in any way. Cryptographic hash functions are used as a measure of integrity; these hash functions are mathematical computations that are performed on the data to generate a fixed number of characters referred to as a hash. With a small change in the data, the hash of the data can significantly change. There are different types of hashing algorithms, each being an improvement to the other.

The main hashing algorithms that are used for integrity checks are MD5 (Message Digest version 5) and SHA-1 (Secure Hashing Algorithm version 1) [47]. These hashing algorithms are used for their speed in calculating a hash. Hash functions are used for a one-way operation, meaning that data can only go one way, resulting in an irreversible process. In other words, a hash cannot be converted back to the original data. However, MD5 and SHA-1 have a few vulnerabilities that allow them to be cracked (obtain the real unhashed data). This is because hashes length is not significantly longer, with MD5 only having 128 bits (16 bytes) and SHA-1 having 160 bits (20 bytes). This means it is easier to perform a brute force attack as what can be seen from password cracking.

Although, password cracking is generally short length strings, hashing an entire file and attempting to get the original file back is near impossible with current computation limitations. Since integrity of the information is at question and not the security of the information these cryptographic algorithms perform their roles as an integrity verifier. In digital forensics the integrity of potential digital evidence is always questioned in a court of law this is to ensure that evidence was not tampered with and that the evidence came from a credible source. Data needs to be available in order to ensure its integrity, therefore the availability of data is one of the vital aspects of cyber forensic services.

### **2.2.2.3 Availability**

The availability of data is ensured by the maintenance of the machine on which the data is stored on [37]. Data is available on-demand at any time giving a user having 24/7 access to the data on the machine. In disaster recovery, for example, when the host machine fails to access to the data is therefore lost and thus making the data unavailable. Backups can be used to prevent the loss of availability of data when there are dire situations. Redundancy mechanisms like RAID (Redundant Array of Inexpensive Disks) is used to ensure data is always available if a disk drive fails the data is still available on another drive as opposed to the data only being available on the failed disk [48].

Communication channels also have a role in the availability of the data and must be functional at all times. DDoS (Distributed Denial of Services) is one attack that in

great lengths attempts to bring down the availability of any rendered services [49]. This is one of the biggest attacks that many companies have to face and avert in order to keep their services running as well as their reputation. In digital forensics, the availability of information is what investigators use to determine or uncover any incriminating evidence. Therefore, potential digital evidence (PDE) must be available to an investigator at all times in order to perform timeous investigations.

#### **2.2.2.4 Authentication**

Authentication is performed by a user supplying a user identity as well as a password or key phrase in order to prevent unprecedented access. Having secure channels and secure validation mechanisms in place help preserve the secure transfer of the authentication information. Once a user is authenticated, they have to be authorized to perform certain operations as discussed in the next subsection.

#### **2.2.2.5 Authorization**

Once a user is authenticated, their access roles are then looked up and based on the access roles a user is then allowed to perform certain operations. In digital forensics only, authorized parties may access PDE due to the sensitivity of the information as well as the integrity of the information. PDE that is collected and stored is only available to authorized users with 2-factor authentication, this further ensures that the data confidentiality, integrity, availability (CIA) is preserved. Once a user accesses the PDE, all processes are logged, ensuring non-repudiation as discussed next.

#### **2.2.2.6 Non-repudiation**

The goal of non-repudiation is that a user cannot deny being a part off or receiving a transaction [50] [51]. In digital forensics, a user cannot deny having accessed PDE and thus obeying the law as well as a chain of custody further safeguarding the PDE's integrity. Therefore, in order to achieve non-repudiation authentication and integrity cannot be violated. The use of 2-factor authentication eliminates the user from claiming that their password was hacked due to the uniqueness of the second phase of authentication [52]. It is near impossible for an attacker to gain access to a system with 2-factor authentication without any knowledge derived from the user. Thus, non-repudiation is achieved, and a digital signature is logged as well as the IP for traceability purposes. The next section focusses on digital forensic investigations and how cases are processed and conducted whilst following strict procedures.

### **2.3 Digital forensic investigation**

Digital forensic investigation (for convenience sake referred to as digital investigation) is the process of uncovering incriminating evidence that can be



admissible and used in a court of law [33]. Digital investigation is a subset within digital forensics that focuses on the process performed to carry out an investigation as well as to discover evidence that can prove that a crime was committed. Several digital forensic investigation processes were followed by various authors, with each presenting their own framework or model [53] [54]. A study by Salamat et al. [55] who mapped the various frameworks together to get a bigger picture, integrated these frameworks to get a more robust investigation lifecycle. This lifecycle is discussed in more detail.

### 2.3.1 Digital investigation lifecycle

From the mapping done by Salamat et al. [55], it can be concluded that the digital investigation lifecycle consists of five main phases that are to be conducted to maintain standardisation. An overview of the investigation lifecycle is presented in Figure 2-1. The different phases are discussed in the proceeding subsections.

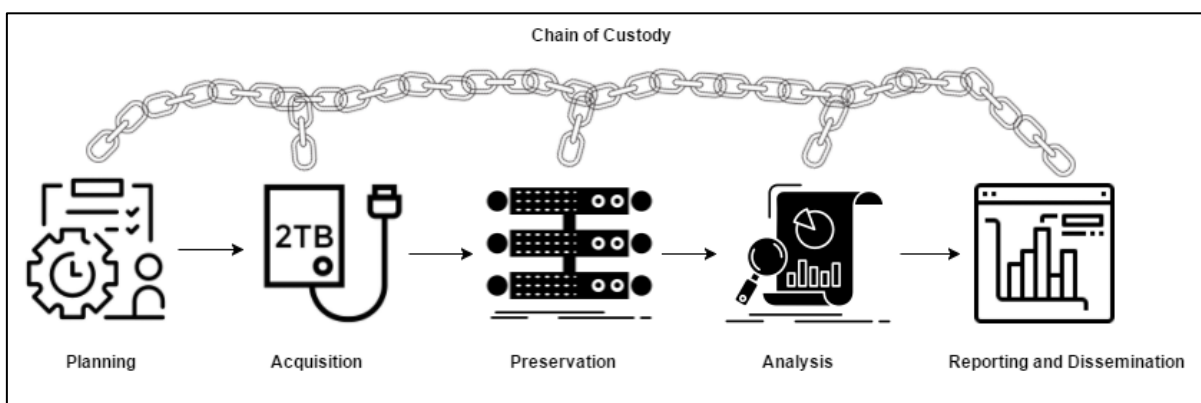


Figure 2-1. Investigation lifecycle

#### 2.3.1.1 Planning

The planning phase of the investigation lifecycle involves identifying and locating computing devices that need to be acquired [56] [57]. These devices must be linked to probable cause, and a warrant is needed to seize the required devices. An investigator would need to study the case at hand and know what crime was committed in order to better collect information from potential sources. For example, an investigator would need to identify the case as being either civil or criminal, and whether the environment where potential digital evidence may be found is controlled or uncontrolled. From there, the investigator would need to prepare how to deal with the case and identify the computing devices to acquire for forensic imaging. The investigator would also need to identify any tools or hardware needed to extract the data for analysis.

### **2.3.1.2 Acquisition**

Only items specified in a warrant are allowed to be collected in a legal manner. The collection of items (in this case, information) needs to be admissible in a court of law [30]. Therefore, the integrity and originality of the data must be proved. During acquisition, a chain of custody needs to be maintained, a log must be kept of all the processes performed, as well as the timestamps of each item [54]. Seizing a digital device can only be allowed if it does not affect day-to-day business operations [30]. An image of the device is normally obtained through imaging software. Before the imaging device is connected, a hardware write-blocker is used to prevent any information being written to the original device and thus violating its integrity. Any evidence found thereafter is consequently inadmissible. The original data is never touched and should never be altered or modified in any way (this serves as an integrity check [37]). When an investigator arrives on the scene, a hash is obtained of the original data. Then a copy is made, and the hashes of the copy and original are compared to ensure that integrity of the data has been preserved.

### **2.3.1.3 Preservation**

The copy of the original is subsequently securely stored and transported in an enclosure, making sure that the data does not get corrupted or incurs any physical damage that may invalidate the collected data [58]. The enclosure is intended to protect the data from extreme weather such as direct sunlight and electromagnetic fields, which may damage the storage device [59]. It is also important to maintain the chain of custody when the data is transported. The data is next transported to a digital forensic lab where analysis of the stored data can be performed.

### **2.3.1.4 Analysis**

Traditionally, data analysis would be done on a copy of the original copy. In other words, the copy obtained and preserved from the above-mentioned phases is copied again and care is taken to ensure that the integrity is the same between the two copies. One copy will now serve as the original. This is done because the original device may need to be used again by the target and to allow business operations to resume as normal. This image copy is safely stored and write-protected, and it is not used for any analysis due to it being equivalent to the original. The second copy is now used for analysis, but it is also mounted as read-only to ensure further analysis is unbiased and the integrity of the data remains intact. The analysis is performed by mapping all data obtained, for example log files, timelines, event logs, and event reconstruction. Several tools that can be used to perform analysis are further explained in Section 2.3.2. After the analysis has been completed, the final reporting phase is reached.

### 2.3.1.5 Reporting and dissemination

This phase serves as the conclusion of the investigation where all the findings from the analysis phase are documented in detail, forming a report. This report needs to be thorough and describe all processes explored and analysed. The report can serve as evidence and be admissible in a court of law, depending on the nature of the case and the processes followed. The author of the report may also need to testify in court. After a case was closed, the data and reports need to be safely stored or securely destroyed, depending on the sensitivity of the case. This process still needs to maintain the chain of custody and only the report and evidence may be destroyed. Thus, the original unprocessed data still needs to be kept, in the event that the case is re-opened. This phase also serves as remediation for the other phases on how to improve the investigation lifecycle by being better prepared. Tools were created to assist with the investigation lifecycle, thereby reducing the burden for forensic investigators to perform analysis and ensure evidence admissibility.

### 2.3.2 Digital investigation tools

Several tools that exist in the digital forensic community help investigators with their day-to-day jobs and reduce the complexity and dynamics of investigations. Some of the main tools that are used in the investigation lifecycle are listed below.

**Table 2.2.** Comparison of digital forensic tools

<b>Tool</b>	<b>Description</b>	<b>Licence</b>	<b>Lifecycle phase</b>
FTK Imager	Dumps and previews recoverable data from a disk of any format. FTK Imager can also acquire live memory dumps and paging files on 32bit and 64bit systems.	Free	Acquisition
Encase	Provides the ability to image disks and perform analysis and evidence identification with the disk image. Provides an investigator with a guided process of investigation lifecycle.	Yes	Acquisition, Preservation, Analysis and Reporting
Dumplt	Capture volatile memory in a live Windows computing environment.	Free	Acquisition
DD	A simple command-line tool that is usually found in Unix systems, enabling the investigator to disk dump an entire disk into an image. This tool does not automate anything or verify the image integrity, making it a manual process.	Open Source	Acquisition
FTK	Analyses disk images in a distributed manner and speeds up the processing of data.	Yes	Preservation, Analysis and Reporting

<b>Tool</b>	<b>Description</b>	<b>Licence</b>	<b>Lifecycle phase</b>
Autopsy	Enables parallel processing with Open Source plugin functionality, making it extensible and effective for non-technical investigators.	Open Source	Preservation, Analysis and Reporting
Volatility	The leading Open Source memory analysis platform that provides powerful packages to analyse volatile content in memory from a memory dump.	Open Source	Acquisition

The investigation lifecycle is a reactive approach to digital forensics. It is performed post-incident when more unknowns are at play and potential digital evidence may be lost due to the volatility of memory and the nature of the digital artefacts. In order to address this limitation, the Digital Forensic Readiness (DFR) approach, a proactive approach of digital forensics, has been developed [60] [61]. The DFR and its corresponding processes are further discussed in Section 2.4.

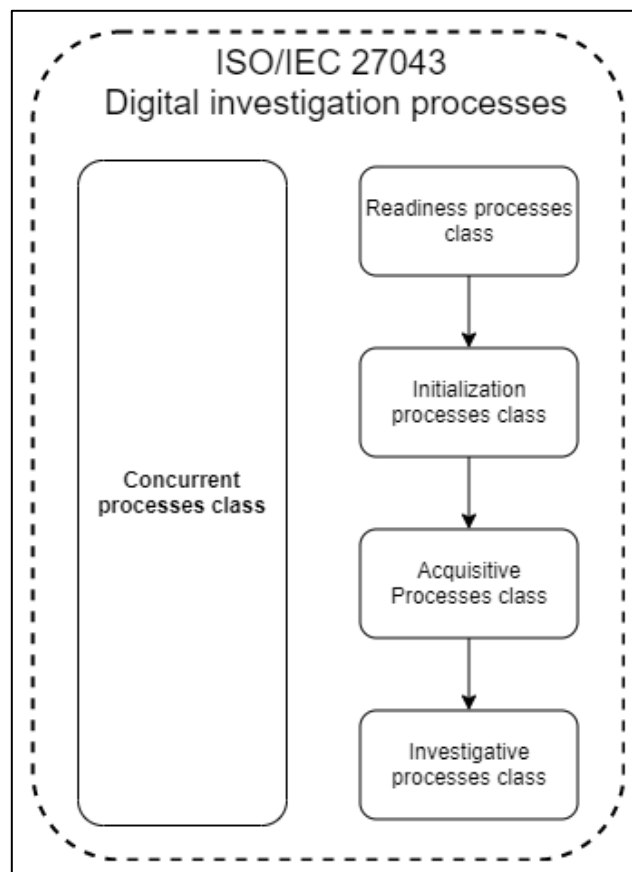
## **2.4 Digital forensic readiness**

Digital Forensic Readiness (DFR) is the ability of an organisation to maximise evidence collection whilst minimising costs (as defined by Tan [60]). DFR has two main objectives. The first is to maximise the ability of an environment to collect credible digital evidence. The second is to minimise the cost of conducting an investigation in an incident-response scenario. This means that to achieve DFR, potential digital evidence collection needs to occur prior to an incident. DFR is a proactive approach to digital forensics that is more robust and cost effective. The implementation of a DFR in any organisation requires an in-depth understanding of business operations and may differ from company to company. Therefore, Rowlingson [61] proposed a ten-step process of implementing DFR. These processes help organisations better identify and prepare for evidence collection. However, the processes are not standardised, and to this effect, some organisations cannot follow this implementation. The ISO/IEC 27043 [30] provides a more robust guideline about digital investigation processes as well as readiness processes.

A high-level overview is depicted in Figure 2-2 that shows the readiness, initialisation, acquisitive and investigative processes whilst having concurrent processes [30]. The initialisation process deals with the procedure followed by first responders, including the planning and preparation phases of the investigative lifecycle. Acquisitive processes provide criteria on how potential digital evidence is identified, acquired, transported and stored. Investigative processes deal with the conducting of forensic analysis, the reporting and presentation of a case, as well as the dissemination thereof. The concurrent processes are the operations that occur

side by side with the other process classes, which include documentation, authorisation, preserving the chain of custody and digital evidence preservation. The readiness processes are the main class, where the other processes can be simplified by gathering potential digital evidence prior to the incident, as well as pre-analysis, thus making the other processes easier to achieve [35] [44].

The next section discusses the ISO 27043 readiness processes in more detail.



**Figure 2-2.** Digital forensic investigation process model

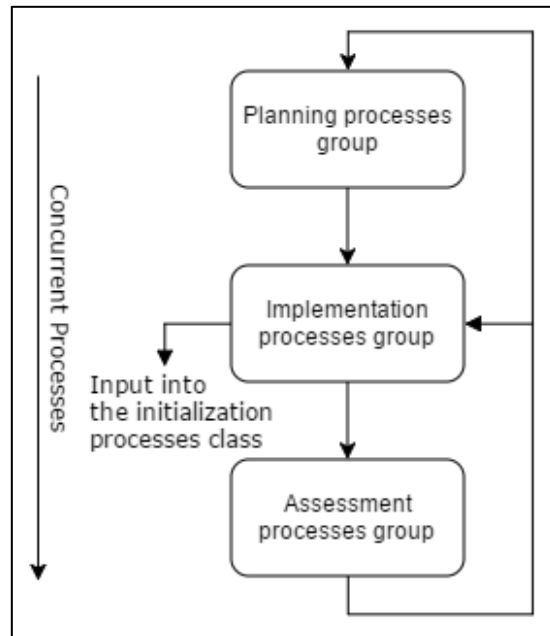
### 2.4.1 Digital forensic readiness processes

In order to assist forensic investigators and reduce the costs of an investigation, an organisation would need to implement some readiness processes. These processes have four main aims [30]:

1. Maximising the potential use of digital evidence
2. Minimising the cost of digital investigations
3. Minimising interference with business processes
4. Preserving or improving the current level of information security systems

In order to achieve these, three readiness processes groups were created, planned, implemented, and assessed, as is illustrated in Figure 2-3. The planning processes group consists of planning activities for scenario definition, identification of potential

digital evidence sources, pre-incident collection and storage, pre-incident analysis and defining the system architecture. The implementation processes group involves implementing all the planned activities. The assessment processes determine the effectiveness of the implementation and indicate if any adjustments are needed.



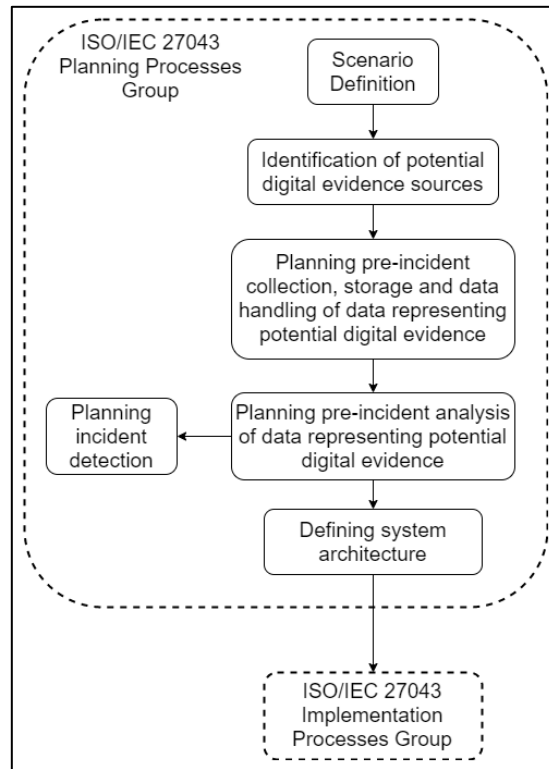
**Figure 2-3.** Readiness processes groups

The planning processes group consists of six stages, as shown in Figure 2-4. The scenario definition encapsulates all the probable scenarios for the examination of digital evidence. This defines the area and details of the aspects examined by the planning phase. The next step is the identification of potential digital evidence sources. This step looks into identifying probable sources that may have evidentiary information that could aid an investigation. It focuses on the larger picture by identifying generic sources like network or activity logs, which may have evidentiary value.

Planning pre-incident collection, storage and handling of data that may constitute potential digital evidence, forms the third phase, during which planning is done on how potential digital evidence should be collected. There also needs to be a criteria on how this collected potential evidence should be stored, represented and preserved, such that it can be used in a court of law, while ensuring forensic soundness processes.

Planning pre-incident analysis of data that represents potential digital evidence revolves around how data collected prior to the incident is analysed and what artefacts or patterns are being looked for in the data. Based on this analysis, the procedures for incident detection are planned as well as the steps that should be taken when an incident is detected. The last phase involves defining the appropriate

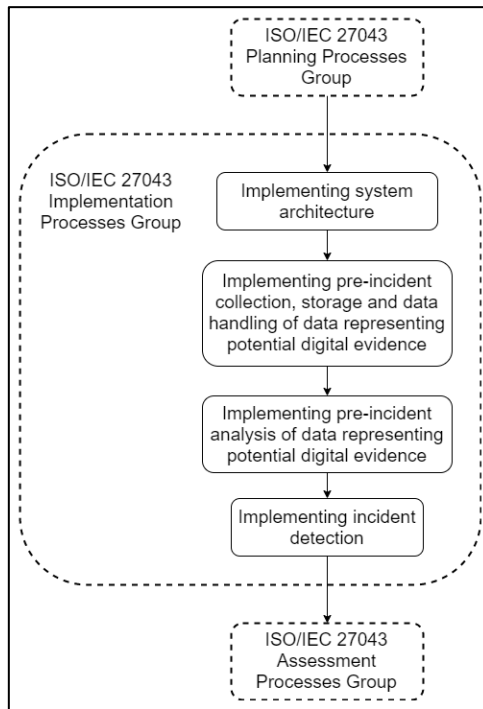
system architecture. In this system-planning phase, the best architecture is chosen to meet all the requirements of the previous phases. The phase focuses on the specific technologies and methods that are to be used in the next process group, namely the implementation process group.



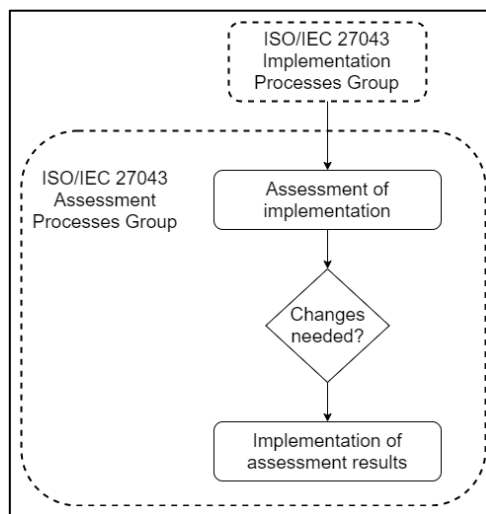
**Figure 2-4.** DFR planning processes group

The implementation processes group comprises four stages, as is shown in Figure 2-5. This group focuses on implementing the planning processes and starts off by implementing the system architecture that was defined in the planning process group. This first stage revolves around the technologies and topologies that will be used to realise the planned system. In the second stage – the pre-incident data collection, storage and handling stage – the implementation of the actual collection of data is performed. Storing the collected information can be seen as a buffer that will be used to implement the third stage – pre-incident analysis – which will then determine if an incident actually occurred and therefore trigger the rest of the investigation lifecycle.

The assessment processes group entails assessing whether more information needs to be collected and analysed. Figure 2-6 depicts the two stages in the assessment process group. They mainly serve as verification for the implementation group to determine whether they need to implement more changes in order that the proposed framework can be more refined and robust.



**Figure 2-5.** DFR implementation process group



**Figure 2-6.** DFR assessment process group

## 2.5 Conclusion

In this chapter, digital forensics was discussed in detail, with the focus on digital investigation and digital forensic readiness. In summary, it not only provides the background to this research, but also paints the picture of why digital forensic readiness can help the forensic community when conducting investigations.

Chapter 3 discusses malware forensics and what it entails. A brief overview of malware and malware analysis is presented in this chapter.



### 3. CHAPTER 3: MALWARE FORENSICS

#### 3.1 Introduction

This chapter provides an overview of malware, types of malware, types of propagation methods, adaptive techniques and ransomware. To better understand the scope of this research, Chapter 3 provides the necessary background to fully understand the proposition presented and modelled by this dissertation.

Cybersecurity involves the protection of computers, networks and data from unauthorised access [5], as well as securing systems from exploitation and exfiltration by internal and external parties [5]. Cybersecurity is one of the major research areas in the current digital age, owing to the ongoing interconnectedness of the internet [5] [62]. Since the internet is so vast, it is almost impossible to ensure that every one of its users is protected against threats [1] [28]. These threats mostly appear in the form of malware that exploits vulnerabilities within a system or resulting from user behaviour. Malicious software, also known as malware, is a piece of software code that is written with the intention to cause harm [5] [63].

According to the report by Symantec in 2017 [64], one in every 131 emails contains malware. The report further states that malware is one of the biggest promoters of system infection and that ransomware is mostly distributed through malicious emails [64]. According to the CSIS-McAfee Report, the potential cost of cybercrime to the global economy could be as high as US \$500 billion (see Figure 3-1) [65]. As a median value shown in Figure 3-1, an attacker stays in a network undetected for approximately 146 days. Furthermore, approximately 43% of cyberattacks are targeted at small businesses [66]. Users continue to have insecure passwords with statistics showing that at least 63% of all passwords in an organisation are compromised [65].

Recent statistics from AV-Test have indicated an exponentially growing trend of malware-based cybercrime over the time span of one decade, as depicted in Figure 3-2 [67]. This further affirms that malware can be expected to rise in years to come. In 2018, AV-Test reported a total of 856.62 million malware samples, in comparison to May 2019 when an increase of 48.29 million samples was observed in under six months. The rapid growth of malware can generally be attributed to the increase in computing devices, lack of software updates, unpatched systems and email infections [5] [64].



Figure 3-1. Microsoft advanced threat analytics infographic [65]

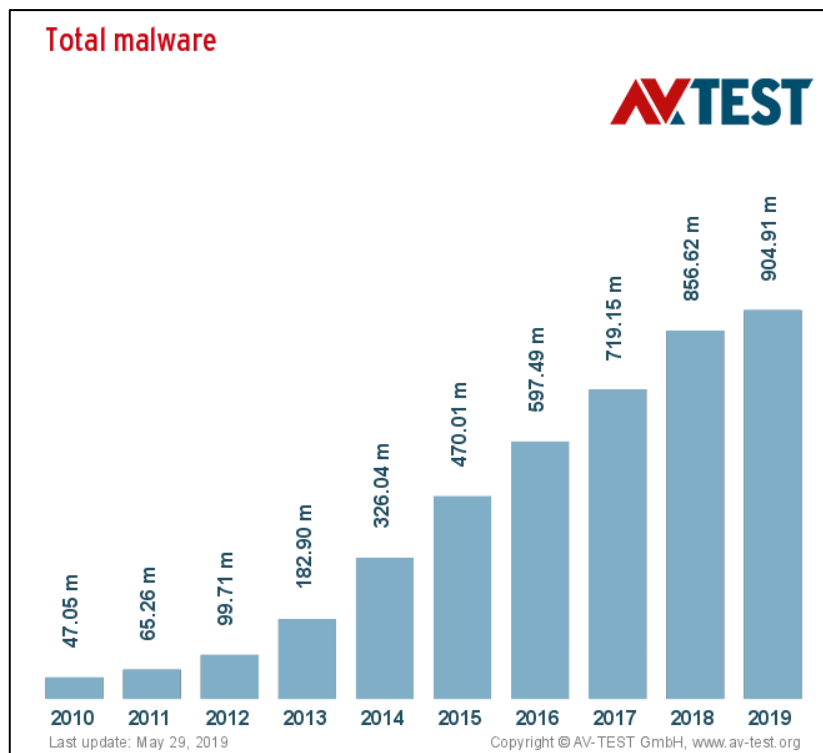
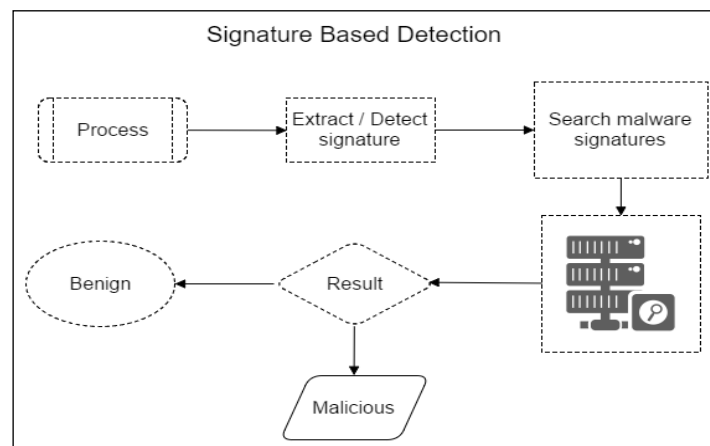


Figure 3-2. A decade's statistics of malware trends [67]

Fortunately, anti-virus software has improved significantly over the years, providing quick updates and defensive measures to prevent malicious activity [68]. However, despite having these measures in place, organisations are still vulnerable to new attacks and have proven to be ineffective against ransom-based type of malware [69]. This is attributed, in part, to the limitations of the traditional signature-based detection methods that simply extract the signature of an executable and then compare it to known malicious signatures [70]. Figure 3-3 next depicts the lifecycle of the malware detection process for traditional anti-virus software in a typical operating system (OS).



**Figure 3-3.** Signature-based detection

In an OS, a process is an execution of a program or executable code that performs a specific task. Signature-based detection is normally performed by taking the process or program and extracting patterns in terms of the byte sequences and actions performed. Malware has significantly progressed in recent years and now has adaptation techniques in place to avoid detection [17] [44]. Ransomware has many variants, with each being an improvement on the preceding variant, thus enhancing the effectiveness of subsequent variants. Due to the unpredictability of program execution, anti-virus software is rendered ineffective in detecting newer variants of malware.

The advancement of artificial intelligence has given rise to more models of detection using machine-learning techniques, for instance neural networks and decision tree classifiers [71]. Other models also looked at using the idea of honey files (files that are likely to be attacked) as a means for malware detection [72]. Event-based approaches looked at patterns and sequences that occurred to help detect malware activity [73]. These models and techniques recorded significant successes in detecting variants of malware. However, novel variants of malware tend to evade detection, while the existing techniques could be biased or limited towards a given variant [69]. Behavioural-based detection methods are more limited in capabilities as they induce higher overhead costs and cause degraded performance, yet added security. A typical process of behavioural detection can be seen in Figure 3-4.

Several factors are considered when analysing the behaviour of an executable – from the type of operations or sequences the executable follows to the patterns that are preloaded in AV databases.

The next section provides an overview of malware and presents the necessary background to understand how malware works and how their activities differ from one another.

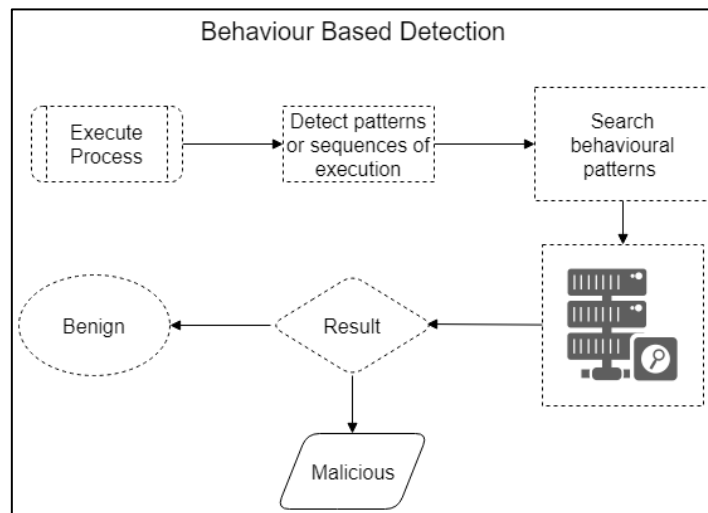


Figure 3-4. Behaviour-based detection

## 3.2 An overview of malware

Given the enormous volume of existing malware samples, each type of malware can be further classified, based on its characteristics. These characteristics help to determine the severity level of a specific malware. Some of the most common types and most recent malware types are further discussed in the proceeding subsection.

### 3.2.1 Types of malware

The types of malware commonly encountered include viruses, trojans, spyware, worms, adware, botnets, rootkits, and more recently, ransomware. A brief description and example of each malicious type are given below.

- **Virus** – Similar to the common definition of the English word, a virus is a type of malware that infects specific files within a computer system. These files can be program files, data or personal files. Viruses can spread themselves throughout the system, causing more havoc. Viruses is a misnomer as they are not necessarily intended to cause harm, but rather to make a user feel that their system is compromised. When a computer is infected with a virus, a range of strange effects may become evident, for example decreased performance, frequent pop-ups, crashes, browser homepage changes and the installation of unknown programs. The effects of a virus can sometimes be reversed easily [74].

One of the most virulent computer viruses was the ILOVEYOU virus [75]. This malware would infect the system and send itself to all the contacts in the computer's mailing list, followed by overwriting files in the system. This could lead to a complete compromise to the extent that a computer could become unbootable.

- **Worms** – Designed to quickly spread over a local network or the internet, a worm would infect a computer and spread to the next computer on the network, essentially infecting all the computers on the network. Morris worm was the first computer worm that was distributed over the internet [76]. This worm worked by exploiting Unix Sendmail and other vulnerabilities, thus slowing down the system. Furthermore, once a system had been infected, it could get infected again and further degrade performance, similar to the effect of a fork bomb (a process continuously replicating itself) [77].
- **Trojans** – This is a kind of malware that seems legitimate or is a part of the legitimate software that has been tampered with. The main purpose of a trojan is to gain backdoor access to the system [78]. With this backdoor, maximum information is extracted and used to plan an attack or breach into systems. A user is often unaware of these backdoors and oblivious to what is happening in the background of the system. This is because attackers run stealthy tools in the background. The different types of trojans, namely backdoor trojans, download trojans, remote access trojans and distributed denial of service (DDOS) trojans are each designed with a specific task. ANIMAL, believed to be the first example of a trojan, dates back to early 1975 [79]. This trojan was portrayed as a game to guess what animal the user was cognitively considering by asking a set of 20 questions. Inside the ANIMAL game was an additional software program called PERVADE, which scanned through all directories in a system replicating the program. Although there was no malicious intent in the replication, it introduced the fundamental idea of how a trojan works by creating backdoors into systems.
- **Spyware** – The sole purpose of spyware is to spy on the user. It runs in the background, tracking everything the user does on the computer – from web browsing to activity on the machine. Spyware provides an easy way for attackers to get hold of the user's passwords and banking information [78]. Some attackers use spyware to lay hands on highly classified information from high-profile targets like high government dignitaries, high-profile lawyers, and government security agents. A good example of spyware software is Pegasus, created by the NSO Group. Once installed, it had root and remote control of any mobile device [80]. A more recent case seen in May 2019 was the case of spyware that exploited the WhatsApp application; a commonly used messaging application. The exploit was executed by sending specially crafted Secure Real-Time Transport Control Protocol (SRTCP) packets through the application's Voice Over Internet Protocol

(VOIP) causing a buffer overflow. This gave Pegasus the ability to install and take over a mobile device without user notification [81].

- **Adware** – This malware is spread through online ads by downloading free games and software. It is not particularly malicious but creates backdoors or vulnerabilities in systems that other malware can exploit [74]. Adware mostly uses catchphrases to trick users into downloading and installing the software. This phenomenon is also known as Potentially Unwanted Programs (PUP). Frequently, other forms of malware like Trojans are used in conjunction with adware to compromise systems. Attacks usually originate from web browsers when users visit malicious sites, where the attacker tricks the user into downloading and installing malicious software. One of the types of adware most available today is 1ClickDownloader [82]. It claims to download any file with the fastest speed, but actually changes configurations in the system that cause problems and degrade performance.
- **Botnets** – These are a group of infected computers on a network that an attacker has control of. The computers usually work together towards a common goal that the attacker wishes to achieve. One such application of botnets is to cause a DDOS attack on a system to take it offline or to breach the system to steal some confidential information [25] [83]. An example of how a botnet can be infected occurred with Mirai malware. This malware scanned the network and infected all Linux-based machines to be used as part of a botnet. Mirai went even further by targeting all IoT (Internet of Things) devices with an ARC processor [84]. This made it easy for the attacker who controlled the botnet to target websites and launch DDOS attacks.
- **Rootkit** – This is the most dangerous and advanced form of malware that gives the attacker full administrator access to the system [74]. Rootkit generally masks its existence in a system so that the attacker can gain maximum time within the network or computer. It goes without saying that since rootkits are masked, they are more difficult to detect, and traditional anti-virus software would not be able to detect them easily. Different levels of infection can occur, namely user mode, kernel mode, and hypervisor mode [85]. The more common level is user mode, due to the ease of implementation and simplicity of infection. Kernel mode is a bit more difficult to achieve as it involves infection at a low level, but it is often extremely difficult to detect. Hypervisor mode is the most damaging, as this is an infection on the firmware level. It tricks the kernel and creates the illusion that it is interacting with the hardware directly and not with some malicious firmware, thus further compromising virtual machines [85].
- **Browser hijacking** – This is malicious software that infects your web browser and gives the attacker access to your online accounts and history [86]. The hijacking attack, which usually attempts to inject the browser with JavaScript, is also known

as Cross-Site Scripting (XSS). Once the browser has been infected through a malicious script, the attacker can gain access to the cookies in the browser. This allows the attacker to use these cookies that contain the user's session keys to websites that were logged into. In this way, the attacker gains access to logged in websites. The browser "Ask Toolbar" is a good example of browser hijacking as it changes the default toolbar and default homepage [87].

- **Ransomware** – Being a form of malware that affects a huge number of systems, mostly in organisations and institutions, ransomware encrypts the user files and holds the decryption key as a ransom for huge amounts of untraceable money, like cryptocurrencies [16]. This ransom money is usually paid through bitcoin because the payment is done anonymously, and it is very difficult to trace where the funds are being withdrawn from. The first known ransomware attack that occurred in 1989 used the so-called AIDS trojan and targeted the healthcare industry. From then onwards, the healthcare industry remained the top target of ransomware reports [17] [88]. This is due to the sensitive information and systems that are needed on demand. Since it is vital for the systems to be online, the only suitable solution is to pay the ransom and decrypt the systems so that operations can resume as normal [16]. A more detailed explanation of ransomware is discussed in Section 3.4.

The next subsection discusses the way in which malware can replicate itself within a network in order to fulfil its purpose and cause the biggest amount of harm.

### 3.2.2 Method of propagation

Different malware uses specific methods of propagation or replication to perform or cause maximum damage as intended. Some of the most common methods of propagation include social engineering, wired/wireless networks, file sharing, virtualised systems, and email. A brief description of each method of propagation follows next.

- **Social engineering** – In this method, attackers exploit human trust and behaviour. They manipulate people to perform a certain activity or to provide important information by using deception and the reputation of trusted friends/partners. This can be as easy as asking someone to download and install malicious software under the pretence that it could speed up the computer [74].
- **Wired/Wireless networks** – Integrated networks are the most vulnerable target if vulnerabilities exist on the network and a malicious payload manages to enter the network. This essentially means that all computers on that network could be infected. This is typically how a worm spreads and causes the most damage whilst using polymorphic functionality to change itself to remain undetected [89].

- **File sharing** – Peer-to-peer distributed file sharing is an easy way to store items on a public or private network. It also provides easier and cheaper file storage as opposed to using a central storage facility. However, it introduces security concerns, because if one computer is infected and uses file sharing, there is the potential that other computers on the network can also get infected [74]. A chain reaction is caused when someone accesses the infected file. For example, an attacker could attach malware to a newly screened movie or piece of music, and when people download and run it, the malware is allowed to run in the background. Another method of file sharing involves the traditional copying of files from one disk to another through an electronic medium [90], for example, using a flash disk or an external drive.
- **Virtualised systems** – Nowadays, many hosting providers use virtualisation in order to reduce hardware costs. If security is not a priority, the malware could infect one virtual system on the server and cause all virtual systems running on that server to be infected [88]. The reason for this is that virtual systems run on the same underlying hardware leveraged by the malware.
- **Email** – Most attacks originate from email spam and malicious emails that entice users to click on links or to download and install applications that open backdoors into systems. Attackers take advantage of inexperienced users by tricking them to do something that can benefit the attacker [74] [88] [91].

Each of the identified methods has a peculiar evasion technique that enhances its effectiveness as a malware. The various types of adaptive techniques used by malware are discussed further in the next section. These techniques are used to evade detection by anti-virus and other malware detection tools.

### 3.2.3 Adaptive techniques used by malware

Over the years, malware started to get more advanced by adapting and counteracting security mechanisms that prevent them from propagating. These techniques adopted by malware render it hard to detect, as each technique brings in a new aspect to consider. Some of the common techniques used are polymorphism, metamorphism, obfuscation, DDNS, and fast flux.

- **Polymorphism** – This technique employs a modification mechanism to avoid signature-based detection. The malware simply changes itself without completely changing the code or changing its execution structure [92] [93]. However, some parts of the malware remain the same, making it easier to be identified by using adaptive detection algorithms [94].
- **Metamorphism** – This technique completely rewrites the malware so that it is extremely difficult to be identified by anti-malware software. With each



propagation, the malware is changed (undergoes metamorphosis) that further adds to its unique behaviour and makes it almost impossible for anti-malware software to identify it [94] [95].

- **Obfuscation** – By using archive files such as .zip, .rar, .tar or .cab, the malware pretends to be an archive. This method encrypts the core (malicious) code so that it cannot be detected through an anti-virus. For example, base64 encoding is commonly used to sneak malware into the system using HTTP/HTTPS channels [94].
- **DDNS** – The Dynamic Domain Name Service (DDNS) is used where domain name resolutions (converting domain names to IP addresses) are performed dynamically in real-time. Compromised IP addresses can easily be moved anywhere, as domain caching is limited to short periods of time. This leaves a small gap for attackers to send through malicious payloads without easily being detected, owing to the small window created when domains are moved [93].
- **Fast Flux** – This technique is mostly used to control large networks through DNS. A botnet can use DNS records to hide malicious websites and phishing attacks by swapping IP addresses in and out at high frequencies [96].

It is not enough to know the basics of malware propagation methods. In order to understand how ransomware works and how it can be analysed, one first needs to understand how, from a security perspective, malware can be analysed. The next section presents several malware analysis techniques.

### 3.3 Malware analysis

Malware analysis revolves around breaking down and analysing malicious software. Several methods can be used to analyse malware and are further discussed in the subsections that follow.

#### 3.3.1 Reverse engineering

Reverse engineering involves breaking down an executable on a low level in order to analyse it in an attempt to reconstruct or determine its construction and composition [97]–[99]. The process of understanding a program or executable is not as easy as it seems, especially when nothing is known about the executable. This process of reverse engineering is a manual discovery process where the dissection of an executable is done on a low level by converting byte code into assembly code to better understand what logic was used in the program. After disassembling the code, the next phase of reverse engineering is to attach a debugger to the executable while it is being executed in order to step into the executable and determine what part of the code is executing.

In order to determine countermeasures and possible defence and recovery mechanisms, security researchers often make use of reverse engineering to understand better how malware works and what makes it so unique. Whilst this technique is beneficial to identify criminals, it poses a threat to enterprise and paid services. It makes it easier to crack paid programs by bypassing the authentication process or restricted features. Malware analysis is a field in which continued research is done [4] [14] [100]. It uses a huge amount of reverse engineering to uncover weaknesses within the malware sample and determine the plausibility of its effects to be reversed. To improve security and prevent the temperment of benign paid software from being reverse engineered, obfuscation techniques and encryption mechanisms are used. However, reverse engineering also introduced a double-sided sword effect, where attackers now use these techniques to prevent malware samples from being reverse engineered [63] [74]. This causes a huge challenge to a security researcher to find weak points in the malware.

Some of the fundamental concepts/approaches to reverse engineering and analysing malicious samples are further discussed in the proceeding subsections.

### **3.3.2 Controlled environment**

In order to reverse engineer a malicious executable, the environment in which the executable is running needs to be controlled [101]. This adds a layer of security to prevent the malware from infecting the host machine and doing something that cannot be undone. The use of sandboxes is a necessary precaution when a malware sample is analysed. Often, the first step in reverse engineering a malware sample is to outright run the executable and to observe what is happening and what the target or outcome is.

More advanced malware like ransomware needs to be run in a sandbox, due to the encryption of files and the irreversible nature of the effects of the ransomware. Many online sandboxing services such as VirusTotal [102] and Hybrid-Analysis [103] work out of the box by just uploading a malware sample. However, the samples submitted are publicly available and require an active internet connection, which is not optimal for larger files. A localised sandbox environment is more robust and performs better because it can be manually controlled – as opposed to the cloud versions. The popular open-source localised sandbox framework called Cuckoo [104] has built-in support for signature detection, both for static and dynamic analysis, which helps with malware analysis as it is easily integrated with popular tools.

Signatures and static and dynamic analysis are discussed in greater detail in the next subsection.

### **3.3.3 Signatures**

An important aspect of understanding an executable is its signature [105]. There are many ways in which a executable's signature can be defined, but the main aspect of a signature is that it is some sort of pattern (string matching) that can categorically or uniquely identify the executable. This pattern can be a static signature, for example taking the hash (generally MD5 or SHA-1) of the executable that is normally used as an integrity measure but can uniquely identify the executable [106].

A behavioural signature is a more advanced form of pattern matching that involves matching certain distinguishable characteristics of an executable and looking at the behaviour of the executable to determine what category or family the malicious sample belongs too. Taking a worm for example, some of its key characteristics would be replication over the network using a certain structure or protocol. Performing in-depth analysis of the executable can take place in two ways – either by means of static analysis or dynamic analysis. A brief overview of the types of analysis follows.

### **3.3.4 Static analysis**

Performing static analysis revolves around disassembling the byte code of an executable and translating it into an assembly language to further determine the logic and patterns that can be extracted [107]. Static analysis focuses on information that can be extracted from the executable without running it. Information that could be extracted includes searching for strings, calculating the entropy of the executable, examining the file signature, and determining the PE (Portable Executable) headers as well as any encryption [24]. Although static analysis works well for a quick overview of what the executable entails, it is still subject to evasion techniques like obfuscation and encryption, thus making the static analysis more limited. With the advances of malware samples, a static analysis approach is not enough to determine key characteristics of an executable. Therefore, the runtime execution needs to be analysed to determine the behaviour of the executable.

### **3.3.5 Dynamic analysis**

Analysing the runtime behaviour of an executable can be a trial-and-error tedious process. One would need to determine the nature of the executable by analysing several factors, including (but not limited to) network communication, I/O operations, memory, system operations as well as functional call analysis. It is important to know the nature of the executable, and therefore it is suggested that when performing dynamic analysis, a controlled environment should be used [101]. This is a precautionary measure to prevent the executable from doing something that cannot be undone. Analysing the memory can determine any key operations and potential passwords or decryption keys. More advanced malware can detect when it is being executed in a virtual environment, thus making dynamic analysis a bit more

cumbersome [108]. Determining the method by which a malicious executable is transferred or replicated can help to improve the results from analysis.

The next subsection briefly discusses exploitation techniques used by malware.

### 3.3.6 Exploitation techniques

Some malware uses existing exploitation techniques or even unknown/unaddressed vulnerabilities, also known as zero-day vulnerabilities [109], to get executed. One popular exploit developed by the NSA but later leaked by the Shadow Brokers hacker group is called Eternal Blue [110] [111]. This technique exploited Microsoft's SMB protocol, where the server mishandled specially crafted packets that allowed arbitrary code execution [110]. This meant that with this exploit, the attackers could deliver and execute their malware with ease. Thus, by determining that an executable uses SMB, it could mean that it is attempting to propagate over the network and support the assumption that it could be a worm.

### 3.3.7 Obfuscation

To protect and hide the operations of an executable in an attempt to make it more difficult to reverse engineer or detect, obfuscation techniques are employed [74]. These techniques help protect the integrity of enterprise applications and paid services. However, attackers may also adopt these techniques to prevent their malware from being detected. Some of the techniques they use are defined below:

- **Rename obfuscation** – renaming variable names to make the aim of the variable less obvious by using different schemes, for example single-letter alphabets.
- **String encryption** – hiding some signatures and hard coded strings by using encryption instead of plain text.
- **Binary linking** – converting parts of the code into separate libraries and linking the object code to the executable, which makes the file size smaller as well as more difficult to reverse engineer.
- **Anti-debugging** – detecting when a debugger is attached to the process to prevent the executable from being altered through debug stepping (going through the execution of code line by line), which is one of the vital processes used for reverse engineering.

Another major way of obfuscation is by encrypting parts of the code. The next subsection discusses what encryption methods are used and how they can help to determine the nature of the executable.

### 3.3.8 Encryption methods

Executables use several types of encryption: symmetric key encryption, asymmetric key encryption and even key-pair public-private key encryption [69] [112]. Most executables use pre-existing encryption schemes in order to have a form of

standardisation, as well as to ensure that nothing abnormal will occur during encryption or decryption. Malicious executables use well-known standards of encryption, often as a combination of two or more encryption standards. This is to ensure a robust encryption process that will make it difficult – if not impossible – to reverse engineer. By determining that an executable performs encryption, it can further be isolated as to the specific encryption schemes that are being used. For example, if RC4 encryption standard is used, we know that the executable might be contacting a command-and-control server, as RC4 encryption is generally used for its simplicity and speed [4].

The next subsection discusses the communication protocols in which a malware executable can be transferred or dropped.

### **3.3.9 Communication protocols**

Executables often have some form of communication with a server, using client-server architecture with the standard TCP/UDP control protocols. By establishing what external servers, the executable is contacting, it is easier to determine what information the executable is collecting and potentially to find out whether or not the executable is from malicious origin. Further analysis can determine what data is being sent and potentially trick the executable by altering the data it. For example, in the case of ransomware, if the SMB protocol is being used, we can make the assumption that the protocol is used to spread or get arbitrary code execution using an EternalBlue exploit [113]. In digital forensics, it is often a complicated problem to trace the origins of an attack as well as to find any incriminating evidence; therefore, criminal attribution poses a major challenge.

### **3.3.10 Attribution**

Determining the origin of an executable is usually not difficult for benign executables. This is because an executable often carries metadata with it, providing some background of the executable as well as the author and company it belongs to. However, tracing the origin of a malicious executable can be a difficult process [114]. Malicious executables do not have this information set and usually carry the details of an anonymous hacker group that is known just by its name [63]. The aim of such a hacker is to offer competition and simply to grab attention. Some hacker groups even embed their signature within the executable to take ownership of the attack. Unfortunately, this signature is not enough to identify a culprit and does not carry enough weight to incriminate someone in a court of law. Using some of the factors mentioned in the previous subsections, executables can be grouped into certain categories.

### **3.3.11 Categorisation**

By categorising executables, reverse engineering becomes a bit easier because a more specific process can be followed to analyse the executable [115]. For example,

if after performing some static analysis it is determined that the executable is malicious and resembles ransomware, it would reduce the scope for dynamic analysis and make it more specific to certain aspects that would be related to ransomware. This helps with the grouping of malware and with identifying the families the variants belong to.

### **3.3.12 Memory analysis**

Analysing memory can prove beneficial with passwords extraction, loaded libraries and buffer data [116]. A deep understating of memory management and memory mapping is needed to perform memory analysis, since it is random access. This is due to the complexity of the address space and the ability to link/look up the place where data is stored in memory. Tools like Volatility [117] that help with memory forensics can also help an investigator to analyse memory and extract dynamically loaded libraries and file locks. Buffers can also be inspected for potential passwords and data flow, and they are useful for editing settings within memory for the executable to change how it behaves. In a ransomware scenario, it is possible that the decryption keys are available within memory for a limited period of time, thus allowing potential recovery from a ransomware attack.

The next section offers an in-depth discussion of ransomware, proving some statistics of ransomware as well as a comparison of various ransomware.

## **3.4 Ransomware**

Ransomware is a form of malware that affects vast numbers of systems, mostly in organisations and institutions. As indicated earlier, this form of malware encrypts the user files and then withholds the decryption key as a ransom for huge amounts of untraceable money, usually paid through Bitcoin [88] [118]. One of the fastest and widespread propagation of malware is through ransomware. Ransomware uses a combination of different types of malware, for instance a worm that replicates and transfers itself over a network and that can be attached to a trojan or adware to enter the system.

As highlighted in Figure 3-5, global ransomware attacks increased by 36% in 2017, with more than 100 more variants used by hackers. A total of 34% of entities globally were willing to pay the ransom and about 64% of such entities involved American companies [64]. The FBI estimates that 4000+ ransomware attacks have occurred globally every day since 2016 [119]. The amount of ransom demanded per attack has increased to an average of \$1077, which is an increase of 266% [64]. Ransomware uses scare tactics to trick people into paying by using threatening messages and setting a time limit to pay before the ransom increases. Ransomware can appear in different shapes and sizes, some being more harmful than others;

however, all have the same goal. Common types of ransomware include cryptomalware, lockers, scareware, RaaS, and leakware.

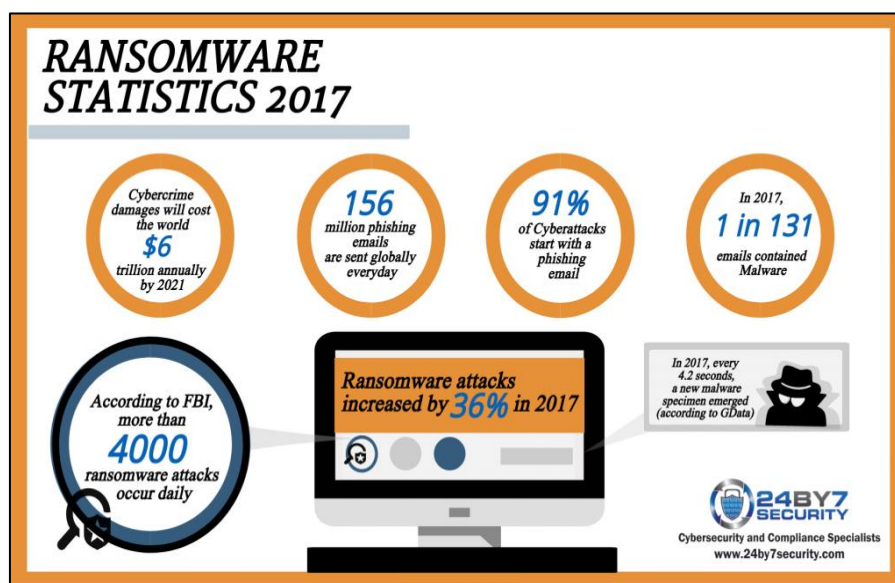


Figure 3-5. Ransomware statistics compiled by 24BY7 Security

- **Crypto malware/encryptors** is the most commonly seen form of ransomware today. Crypto ransomware has the capability to cause significant damage within a short time, as it simply encrypts as many files as possible matching the file extensions that the attacker chooses. These file extensions differ for each ransomware. After the files have been encrypted, the attacker extorts money in return for their decryption. An example of crypto ransomware was seen in the recent outbreak of the devastating WannaCry ransomware in 2017 [120] [121].
- **Lockers** infect the operating system in such a way that the legitimate user is locked out of the system. By modifying the bootloader of the OS, the attacker then holds an unlock key until the user pays the ransom. Modifying the bootloader is a tricky task and requires the Master Boot Record (MBR) to be changed. This is usually achieved by rootkits or trojans. After the bootloader has been modified, the attacker loads their own program/OS instead of having the system boot normally. Lockers in their true form have not been used much lately, because they are relatively easy to remove by reloading the bootloader [118]. However, more advanced crypto ransomware makes use of locker ransomware as well. An example of this is Not(Petya), where files are encrypted on the disk, and the bootloader is modified to not let the OS boot normally [88].
- **Scareware** is a form of ransom disguised as a genuine application. It claims to have discovered security vulnerabilities in a system but demands money to fix them. When the user refuses to pay, the software will display ads and pop-ups

that will cause the user to think the computer is infected and eventually convince them to pay [122]. Scareware is usually not widespread as it relies on a user to install the application rather than to merely click on a malicious link or email. These applications are generally used to steal user information like the sites you browse, personal files, and banking information. SpySheriff is an example of scareware posing as anti-spyware software [123]. What this scareware does, is to scan the disk and pretend that spyware has been installed, thus prompting the user to pay for the spyware to be removed.

- **RaaS (Ransomware as a Service)** is like a middleman for a ransomware attack. RaaS provides a ransomware service where malware is hosted and distributed anonymously by a group of skilled attackers. It is a service that is available to people to buy/rent on the dark web. The service providers of the ransomware take care of everything – from developing the ransomware, injecting it into the targets and spreading it on the network. They also manage the ransom payments and decryption process [88] [124]. An example of RaaS is Cerber, which has Command and Control (C2) servers that the ransomware would contact for remote execution and control [27].
- **Leakware/Doxware** is a form of ransomware that steals personal images and/or information from a computer and then demands a ransom as a form of blackmail. Leakware turns a user’s personal data against them, intimidating the user to pay as soon as possible so as to avoid their reputation from being tarnished.

Each of the above variants leverages a different method or builds on the flaw of another variant to make it more harmful and widespread. A descriptive summary of existing ransomware is presented in Table 3.1.

**Table 3.1.** A summary of trending ransomware

Name	Encryption algorithm	Method of propagation	Vulnerability exploited
WannaCry	AES-128, RSA-2048	EternalBlue	Windows Server Message Block (SMB) protocol
	<p><b>Remark:</b> WannaCry exploits the SMB protocol by using the EternalBlue exploit developed by the NSA. It misuses the way Microsoft Windows handles specifically crafted packets, which enable the execution of certain code from the payload. WannaCry encrypts each file with a different AES-128 key, which is further encrypted with an RSA key pair and then added to the header of each file. The private key of the Command and Control (C2) server is needed to decrypt the encrypted AES-128 decryption key. WannaCry also has a control mechanism called the kill-switch to stop the ransomware from propagating and spreading through networks.</p>		



Name	Encryption algorithm	Method of propagation	Vulnerability exploited
(Not)Petya	AES-128, RSA-2048	EternalBlue, Ukrainian tax software update	SMB, Master Boot Record (MBR)
	<p><b>Remark:</b> Similar to WannaCry, but more harmful. The infection process does not stop upon infecting system files, but also changes the bootloader to load the malware. This process bypasses the booting of the OS and uses the CHKDSK process where, instead of loading the OS, it loads the Petya ransomware. While this message is shown, it begins spawning processes in the background to encrypt the user files.</p>		
Locky	AES-128, RSA-2048	Phishing emails	Microsoft Word Macro
	<p><b>Remark:</b> Locky ransomware infects the system through social engineering in the form of a malicious Word macro. This macro then runs the trojan binary to start the encryption. The encryption used here adopts the same approach as that of WannaCry and Petya, creating a new trend. The Locky method of encryption is secure if the private key of the C2 server is not globally known. Thus, this method is unbreakable due to the mathematics involved in the RSA encryption algorithm.</p>		
Cerber	RC4, RSA-2048	Spam emails and ads	Microsoft Office Documents
	<p><b>Remark:</b> Cerber is a RaaS that provides a toolkit that works even if you do not have an active internet connection. This ransomware enters systems through infected office documents that load the malware through a VBScript. The Cerber form of malware is well controlled since specific hacker groups are working together to provide this service to less experienced hackers. They manage to propagate the ransomware faster by using affiliated programs and social engineering techniques. The service is for those criminals who lack the technical expertise to execute such attacks and are looking for quick profits.</p>		
Crysis	AES-128	Remote desktop service, VM environment	Weak or leaked accounts
	<p><b>Remark:</b> The Crysis attack is based on a user-oriented attack where remote desktop services are hacked. It gives attackers control of the machine, allowing them to manually install the ransomware. Crysis also makes use of a C2 that is used to manage and carry out the attack on a larger scale.</p>		
Grandcrab	RSA-2048, AES-256, RC4	JavaScript and Document Dropper	Phishing, email spam

Name	Encryption algorithm	Method of propagation	Vulnerability exploited
	<p><b>Remark:</b> Grandcrab has had many versions released – from V1 to currently V5. This is because each version had a flaw in the encryption that allowed decryption without paying the ransom. The analysis showed that Grandcrab is a RaaS with each version improving on the former. It uses RC4 encryption to communicate with the C2 server and uses custom packing to avoid detection by anti-malware tools. Grandcrab also uses commands to stop several important processes like MySQL and antivirus software from running. Grandcrab relies on emails and drive-by downloads to spread throughout the network.</p>		

The summary in Table 3.1 reveals that most types of ransomware exploit the lack of user education and the unpatched security vulnerabilities that exist in Microsoft Windows. Furthermore, it shows that AES-128 encryption is the most common encryption algorithm used by ransomware. Given that ransomware infects and hinders access to the system, post-mortem forensics (forensics performed after an incident has occurred) is not feasible. Consequently, a more pro-active approach is required to identify and potentially acquire any cryptographic evidence from a system. The integration of digital forensic readiness into an organisation can potentially provide a higher probability of successfully decrypting a ransomware-attacked system, as well as yield crucial information/evidence about the attack.

### 3.5 Conclusion

This chapter provided some of the necessary background knowledge to understand the objectives of the current research. Statistics also prove that malware is expanding continuously while ransomware does not show any signs of subsiding or dwindling. This is rather concerning, as the complexities of performing ransomware investigations are increasing. A summary and analysis of common ransomware were also presented in this chapter and provided some insightful information.

Part II entails the discussion of the proposed framework and prototype. The proposed framework on ransomware readiness is discussed in the next Chapter.

# PART III

## FRAMEWORK AND PROTOTYPE

## 4. CHAPTER 4: RANSOMWARE READINESS FRAMEWORK

### 4.1 Introduction

This chapter is the first of the contribution chapters and provides an overview of the framework emanating from this research. A framework, in the field of computer science, is the approach taken to solve a generic problem by defining the perspective, procedure or method with which it can be solved, using a high-level representation [125]. Typically, a framework provides the necessary detail required to create a viable solution that can address the requirements at a high level. Frameworks are therefore abstract and cover a much wider problem scope that can comprise one or multiple outcomes. A model, on the other hand, represents a specific solution, system architecture or composition of concepts [126]. One of many models can exist within a framework to instantiate the framework and further show its use. Models and frameworks are scientific research methods because they involve a proposed, described and evaluated methodology, which is repeatable, given the same conditions [125] [126]. The remainder of this chapter explains the proposed ransomware readiness framework and all of its subcomponents of identification, collection and secure storage.

### 4.2 Ransomware Readiness Framework (RRF)

The proposed ransomware readiness framework (RRF) for digital forensics consists of three main phases or parts: Parts A, B and C. Part A focuses on the identification of potential evidence sources for the purpose of ransomware forensics. Part B describes a trigger-based mechanism for the collection of potential digital evidence for ransomware forensics. Part C elaborates on secure storage of the collected potential digital evidence for ransomware forensics. The overall high-level view of the RRF is shown in Figure 4-1. A detailed discussion of this framework is presented in the subsequent sections.

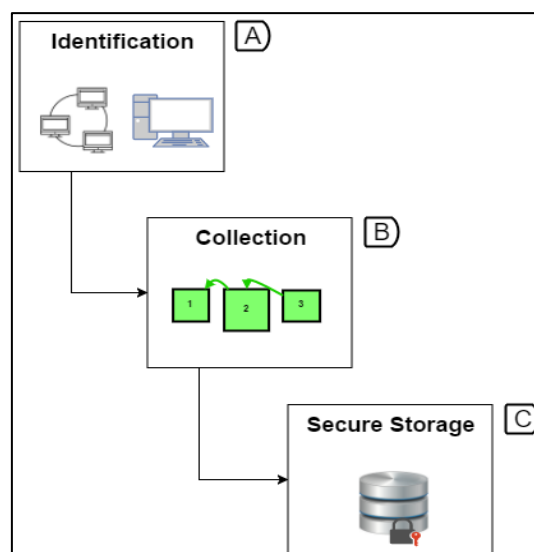


Figure 4-1. A high-level overview of RRF

## 4.2.1 Identification

Part A of the RRF involves the identification of digital sources that may host evidentiary information. Such sources can be any digital device from hard drives to cell phones, but mostly they are computing devices such as personal computers and laptops. As outlined in Chapter 2, conducting a digital investigation in digital forensics requires from an investigator to identify the digital media that need to be acquired for an investigation – which can lead to business disruption and unforeseen costs. Implementing DFR processes within an organisation can significantly decrease these costs. Therefore, with the adoption of the RRF, an investigator would beforehand know the details of the evidence sources to collect. The details of Part A of the RRF are shown in Figure 4-2. They comprise four main aspects that need to be identified as far as ransomware is concerned, namely network architecture, computing devices, operating systems and evidence sources. A description of each component and its corresponding composition are presented in the subsequent subsections. In each section, a reference to the component is indicated with a unique number (A1-A6) to aid the reader's understanding regarding the part of the framework that is being discussed.

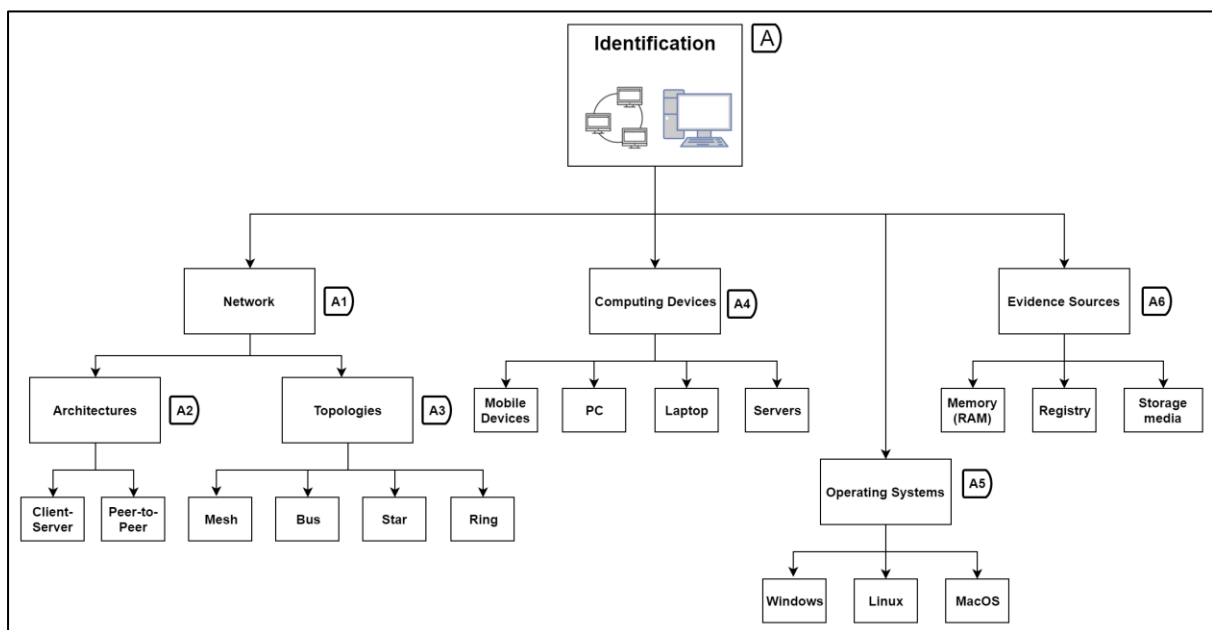


Figure 4-2. Part A - Overview of identification phase

### 4.2.1.1 Network

This section relates to the network component (A1) of the proposed RRF. Classically, the physical and logical design of how computing devices are connected is referred to as the network architecture. The way in which these devices are physically connected is referred to as topologies. Network connections can utilise various protocols for communication like Transmission Control Protocol (TCP) and Internet Protocol (IP).

TCP is well known for its reliability of data transfer, while the User Datagram Protocol (UDP) is known for its speed of data transfer. There are two main types of network architectures (A2), namely client-server and peer-to-peer (P2P), which are further depicted in

Figure 4-3 and

Figure 4-4 respectively. These architectures were identified as the most widely used and they carry the necessary information to and from services rendered on the internet. It is necessary to understand the way these architectures and topologies work to know how data can be acquired from the network. For example, with a P2P network, backups are difficult to achieve as each node may have a different state. For this reason, backups will require more storage and computation time.

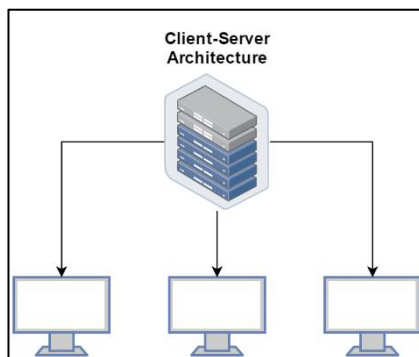


Figure 4-3. Client-server architecture

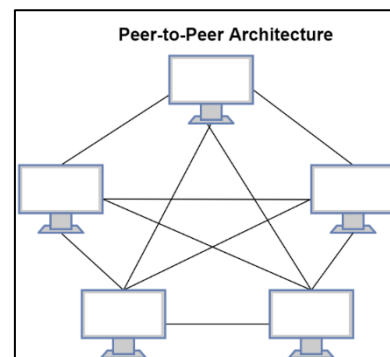


Figure 4-4. Peer-to-Peer architecture

There are several possible topology (A3) setups in a networking environment, with the top four commonly used topologies being bus, ring, star, and mesh (see Figure 4-5). Each topology has an advantage over the other and making a choice depends on the organisational policies. The topology of the network needs to be known in order to effectively find devices within the network and to trace where network traffic is going towards. Knowing the topology is particularly helpful in identifying where network packet loss and where malware samples could have originated from within the network. It also helps to define countermeasures in case a network threat has been detected, because the infected machine needs to be taken off the network to avoid further infection of other services. For example, if a machine was infected with malware and was removed from the network following a bus topology, there would be an open connection on the network and network traffic will not flow correctly.

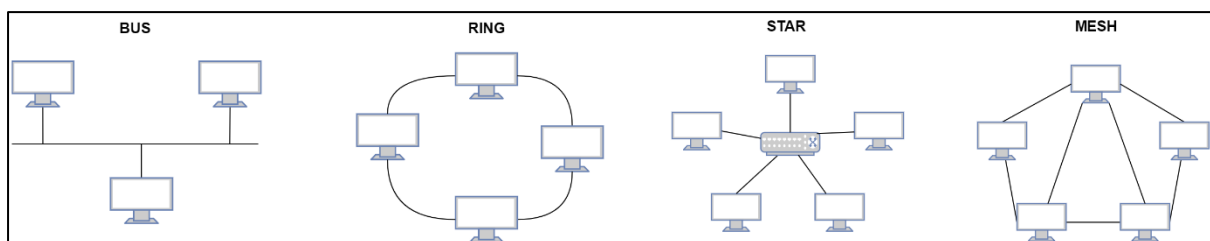


Figure 4-5. Common network topologies

#### **4.2.1.2 Computing devices**

A computing device (A4) is anything that can perform computations. It comprises a Central Processing Unit (CPU) that processes data and performs calculations. Computing devices are commonly used in modern technology. It is important to identify the type of computing device on the network as this helps with identifying what sources of information might be available. For example, with a laptop, the process architecture type and components can be determined beforehand, rather than to try and identify the device post-incident. Whilst there is a large list of computing devices, the most common devices are personal computers (PC), laptops, servers, and mobile phones. Each computing device uses an underlying Operating System (OS) to process and organise data so as to achieve a flow and structure of operations.

#### **4.2.1.3 Operating Systems**

Operating Systems (A5) play an integral role in any computer architecture [127]. In order for an application to interface with hardware, an OS is needed to control input-output process (I/O) as well as communication between components. The OS also controls the execution of programs as well as resource allocation and management. The latter is particularly important to be identified as it indicates what analysis and acquisition processes need to be performed. Microsoft Windows has been around for more than four decades and it has made an impact on our daily lives [128]. Applications in a Windows environment have to be in certain formats and conform to certain standards. For example, an executable (.exe) file is the most common file type for any application in Windows. There are several versions of the Windows OS with each providing better features and performance than its predecessor. Therefore, it is important to identify the OS and the version, so that the knowledge base (where information is stored, the vulnerabilities and features) is known beforehand. Such knowledge is essential for forensic and security-related exploration, for example the identification and extraction of evidentiary information.

#### **4.2.1.4 Evidence sources**

Identifying evidentiary sources (A6) is a challenge for DFR as it is difficult to pinpoint what sources will contain the necessary evidentiary value. However, the registry within a system holds the configurations and metadata about the system as well as its state. In a Windows OS, the registry holds a plethora of important information that investigators use to identify and extract evidence from a machine. Sources of digital evidence in a Windows OS are further discussed in the sections to follow.

##### **4.2.1.4.1 Memory**

The dynamic and static memory of a Windows OS contains a vast amount of information that pertains to the current state of the machine. It can potentially provide an investigator with significant information about the active user, running processes, as well as any malicious processes that may be running in the background, unknown to the user. Most damages that a malicious program can do to a system usually occur

in its memory [11] [129]. The memory is a repository for data and program code that is systematically structured. This structure is similar to a linked-list (see Figure 4-6), which consists of fixed block sizes where chunks of allocated data are slotted and stored. The location of these newly allotted data slots is determined by using lookup tables to find and locate data within this structure, giving direct access to a specific address for faster access [129].

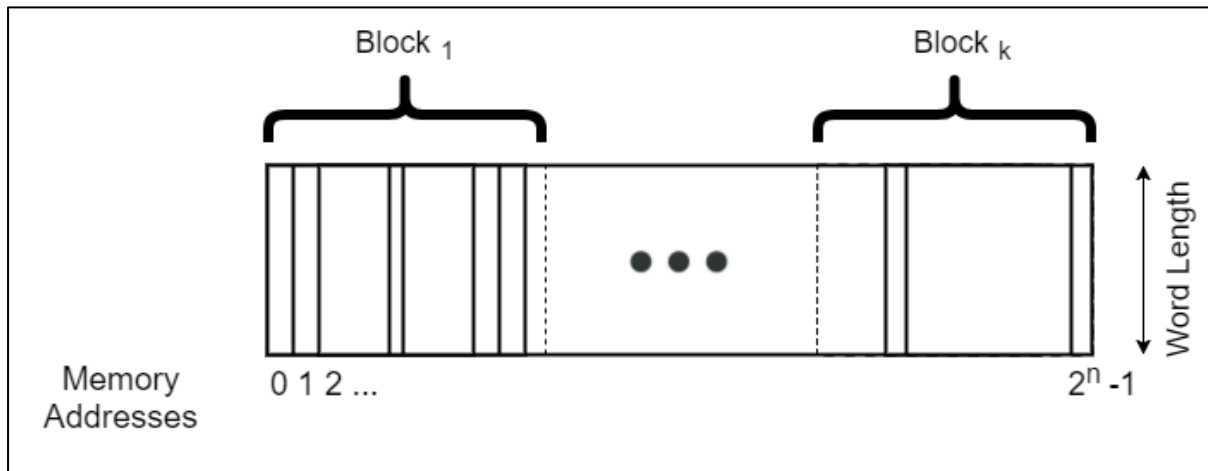


Figure 4-6. Memory structure

Given that the physical memory of the Windows OS has a limited capacity, virtual memory is created. Virtual memory is a chunk of allocated memory that exists in secondary storage, and it extends the amount of data that can be stored [130]. The process of swapping from virtual memory to main memory is called paging [130]. With paging, however, gathering potential digital evidence becomes inherently complex due to the lookup and address mapping that needs to occur before information can be extracted. One such case is the extraction of cryptographic keys that can potentially be found in memory [129] [131] [132]. The cryptographic key could be split up and stored in different pages, and it can possibly be split between several non-contiguous pages, making it more difficult for an investigator to manually scavenge these pages for such keys. For instance, using the memory structure, search optimisation can be performed through virtual address reconstruction. This is achieved by rebuilding addresses that are paged out of primary memory to secondary storage where virtual memory is located. Due to the fact that primary memory is volatile and secondary memory is not, searching through virtual memory would be more effective, since there would still be a chance of finding potential keys. However, such a search will work effectively only in a post-incident scenario. This is because the incident needs to have occurred in order to analyse both the physical and virtual memory space of a process. Identifying what memory information would be relevant in near real-time, can significantly help to speed up the investigation and the acquisition processes.

#### 4.2.1.4.2 Registry

The Windows Registry is a hierarchical database that consists mainly of the configurations and user metadata of the Windows OS [133]. The registry is a



structured, complex database that contains entries known as registry key-value pairs. The registry is a rapidly and constantly growing repository that grows significantly in size over time. With such rapid growth, it requires significant storage capacity [23]. Searching and identifying information within the registry involves a manual search process. This process often leads to an array of difficulties for a DFI, especially in scavenging for evidence during an investigation. Furthermore, the manual process is vulnerable to the error of omission and commission, while subjecting the investigator to the complexity of data analysis. Studies have been conducted to attempt to automate the process of scavenging for potential information from a Windows registry [23] [134]. By automating the process of identifying evidential registry keys and time lining of registry events, the amount of data to search through and collect should be minimised.

#### **4.2.1.4.3 Storage media**

Storage media include all devices that have the capacity to store data. This data can be in any format, depending on the device and the size of storage. Typically, storage media are mostly available in larger quantities. Identifying where storage devices are located and how to obtain data dumps or extract information from these media can significantly help an investigator to reduce the amount of time required to survey the sources.

The identification of the evidential sources mentioned above constitutes Part A of the proposed framework. The next section discusses Part B of the ransomware readiness framework, which involves the collection of digital evidence from the identified sources.

### **4.3 Collection**

The collection of Potential Digital Evidence (PDE) can be a rather complex process. This difficulty can be attributed to the monitoring of a system while ensuring system stability and integrity, as well as preventing or reducing the memory footprint that the collection mechanisms could induce on the system (external modifications to the system). For example, dumping the contents of memory with a tool like DumpIt [135], may potentially invalidate the machine's integrity, as the tool may have some influence on the memory and affect the state of the machine. Extracting information from a system to aid the digital forensic process mostly relies on a dynamic data extraction approach, due to the volatility and constant changing of information.

A diagrammatic depiction of the collection phase appears in Figure 4-7. The collection phase involves two main types of information, namely dynamic information (B1) and static information (B9). Dynamic information focuses on the collection of information that is only obtainable from runtime of a system or executable. Static information on the other hand is information that is extractable from the executable prior to its execution. Further details of each of these types are discussed in the subsections that follow.

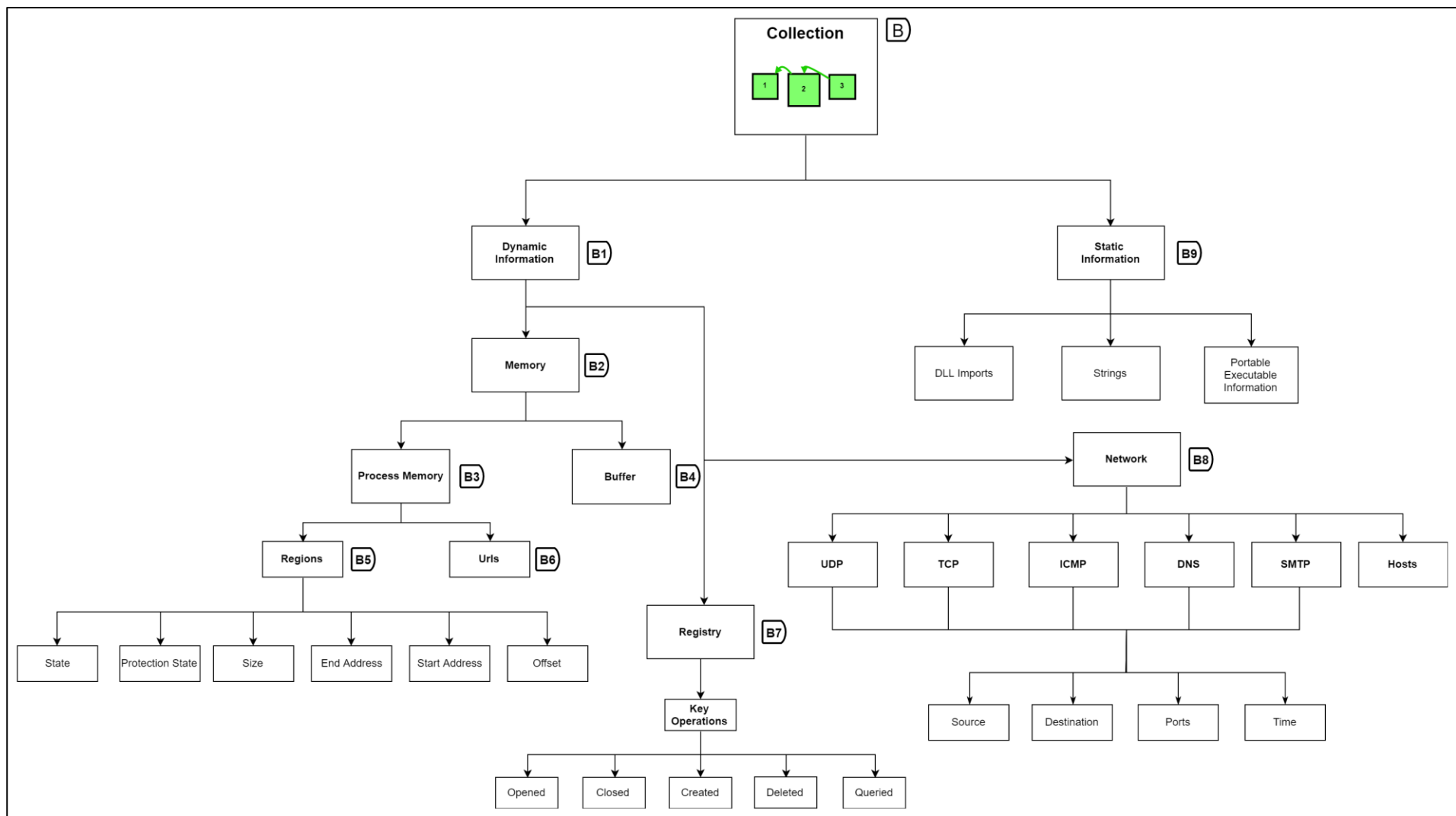


Figure 4-7. Part B – Overview of the collection phase

### 4.3.1 Dynamic information

When dealing with dynamic information (B1), the first location where this information can be found would be volatile memory. This is because volatile memory changes as the OS performs tasks, and as programs and processes are loaded and executed. This process has however grown increasingly in complexity from the older versions of the Windows OS to newer ones. This is particularly true for proactive forensic processes where the access barrier is created by the security restriction enforced by the OS. Methods and techniques of how information can be extracted from memory, as well as the approach proposed in this study, are presented next.

#### 4.3.1.1 Potential evidence collection from memory

A proactive approach for the collection of memory information (B2) has been proposed in recent studies [136]–[138]. More specifically, however, the need for a trigger-based approach has been identified as a potential technique for effective proactive information collection [24]. This is because such techniques can reduce process overheads whilst maintaining system stability and performance, as opposed to constantly scanning through memory.

A context-aware trigger-based approach to data collection is explored in this study. This technique is similar to the concept of crowdsourcing where there are several components that provide information to provide context [139]. Examples include monitoring different aspects of a computer, such as storage and processor usage, which, when combined, provide a better understanding of the overall system health. This trigger can actively monitor the system by using the least amount of processing power whilst minimising the overhead cost of searching [129]. This is achieved by registering events within the Windows OS. Alternatively, it can also be done by scanning through all processes to determine newly created processes in a highly efficient manner. The signature of a known malware can furthermore be added to the logic of the trigger mechanism to reduce the unnecessary computation of searching. The criteria for the trigger mechanism focus on, but are not limited to the following:

- Entropy changes in files
- Autorun entries added to the registry
- Scanning through files and mounted drives
- Loading of the Windows crypto library
- Detecting the deletion of shadow volume copies

These criteria were met through extensive analysis of the execution of malicious samples such as WannaCry and Petya ransomware and extracting their key characteristics. The criteria can also be used for ransomware detection, as discussed later in Section 5.2. As asserted in findings in [129] [140] [141], volatile memory represents one of the most reliable sources of forensic evidence pertaining

to cryptographic keys and active malicious processes. This is because all the buffer data and computations are loaded into memory for brief periods of time. A detailed discussion about how process memory can be useful appears in the next subsection.

#### 4.3.1.2 Process memory

Collecting process memory (B3) can be useful, particularly when a process is writing to restricted memory or writing into another process memory space. A process has access to several memory regions (B5) [130] that contain much evidential information. Memory regions are used in executing instructions and performing operations within memory for Windows OS. A process memory region is a region within memory that is specifically allocated to a process for its execution. In the collection phase, process information is collected based on three main segments or regions, namely text, data and stack [136].

Instructions are loaded in the text segment and are to be executed from the byte code of the executable. This helps with manual tracing of where certain malicious instructions can be loaded into memory. If this segment is marked as read-only so that the instructions set cannot be manipulated, the reason for it is usually to prevent system instability and exploitation.

The data segment consists of data on which the process needs to run. Such data can be anything from constants to statically allocated memory, or an initialised memory for global storage throughout the process execution.

The stack segment is used by the process for storage and function call information. When a process is allocated, each memory region has a start and end address, as well as a state, offset, type, size and access rights. The start and end address signify where such memory region is located within memory, while the offset specifies the number of addresses after the start address of the entry was collected. This offset may sometimes indicate that the address may be found in the paging table and a lookup needs to be performed. The type, on the other hand, specifies the type of memory, which is classically categorised into a heap or stack space.

The state refers to the total paging or frame size, which equates to the virtual memory per process. Within the process memory, several Uniform Resource Locators (URL) (B6) can be found, which may have evidential value. For instance, if a process attempts to download content from the web (e.g. a worm malware downloading the payload from a URL and then executing it) such information can be useful for malware forensics. Dumping all the URLs can help an investigator to detect malicious targets that could potentially be used to determine whether a process is malicious or not.

Buffer information (B4) from a process can be helpful in identifying cryptographic keys in memory during a ransomware attack. This is possible intuitively, given that cryptographic functions for creating cryptographic keys store the result in temporary buffers. This could logically provide the potential to extract the cryptographic keys from a ransomware attack. While this objective is theoretically possible, there are some concerns in capturing cryptographic keys from memory in terms of the legal objection, privacy claim, as well as anti-forensic methods that prevent the plain key from being stored in a buffer by using encryption. Furthermore, other techniques that integrate explicit buffer clearance induce greater complexities during the capturing of buffer information.

The next section focuses on what interaction ransomware has with the registry – ranging from reading registry information to writing to registry. The details of this activity are discussed below.

#### **4.3.1.3 Registry**

Ransomware leaves traces of itself in the registry (B7) as it uses the registry to query and modify some system configurations in order to control and manipulate the system [28]. In most cases, the ransomware creates and modifies a few keys with one common key (Computer\HKEY\_CURRENT\_USER\Control Panel\Desktop\Wallpaper) where the ransomware can set the background image to elicit a prompt response for the ransom. Advanced ransomware creates registry entries that can instruct the system to automatically run the malware if the system is rebooted. The registry can also point to the location of the malware. Therefore, the registry could be a good non-volatile source of information to successfully identify ransomware metadata and other evidential information. Collecting information that was changed by the process can further help identify the malicious context of the executable.

Several operations that can be performed within the Windows Registry. These operations are based on what can be done to the keys, for instance, keys opened, queried, deleted and created. They are also self-explanatory, and each operation has a specific process. Collecting information on the event-based operations can help identify the common keys queried and opened. It can also provide a forensic sequence of events to determine key characteristics. Thus, there is forensic importance in determining whether certain applications were installed, or certain information is available within the system.

#### **4.3.1.4 Network**

Monitoring and collecting information from the network (B8) provide a better understanding of what a process is doing. Such knowledge can be used to corroborate evidence collected to support a claim. Information collected from this phase encompasses the source IP, destination IP, offset in which it can be located in the pcap file (a common name for a collation of network packets), the source and

destination ports, as well as the time it took to connect and transfer data. This information is collected for major networking protocols like TLS, UDP, DNS, HTTP, ICMP, SMTP and TCP. From this, a collation of the various destination IP addresses can be formed as the connected hosts. It can provide an investigator with a summary of all the destination resources and the processes contacted or communicated with. The collation can also be used to identify malicious agents that steal information from a machine, determine potential criminal machine destinations and thus block their access to the network.

#### **4.3.2 Static information**

Information collected about the executable prior to its execution is referred to as static information (B9) and can be extracted from the binary of the executable. When collecting static information, extracting 'strings' from the binary can help to find any signature or pattern in the binary construction of fixed input request. The term 'strings' refers to any consecutive number of textual characters that seem to form a word or phrase. Identifying and collecting Dynamic Link Libraries (DLLs) that are statically instructed to be loaded can further identify any suspicious libraries. Patterns of libraries loaded can also be extracted to gather knowledge of DLL load sequences and determine malicious patterns. However, the limitation of static DLL collection is that hackers have adapted to dynamically loading DLLs at runtime. This technique provides the ability to detect malicious DLLs without execution of the malicious process.

Portable Execution (PE) is a file structure typically followed by a Microsoft executable. From this structure, the dispatcher (a program within an OS that schedules process execution) will load the executable into memory. The process memory segments as mentioned in Section 4.3.1.2 predetermine the size of the data, virtual address space, name of the segment, virtual size and entropy of the executable. The entropy of the segment can be used to ascertain if certain segments consist of random characters. If there is a significant amount of randomness, the entropy value will be higher, signifying that those segments may be encrypted. The information to be collected from the various computer device sources using these techniques, will be stored in conformity with forensic standards. The storage process, which constitutes Part C of the proposed framework, is presented in the next section.

#### **4.4 Secure storage**

The last phase, Part C, of the proposed framework, is presented in this section. Structural composition of the phase is further depicted in Figure 4-8. The secure storage process consists of several security and data integrity mechanisms.

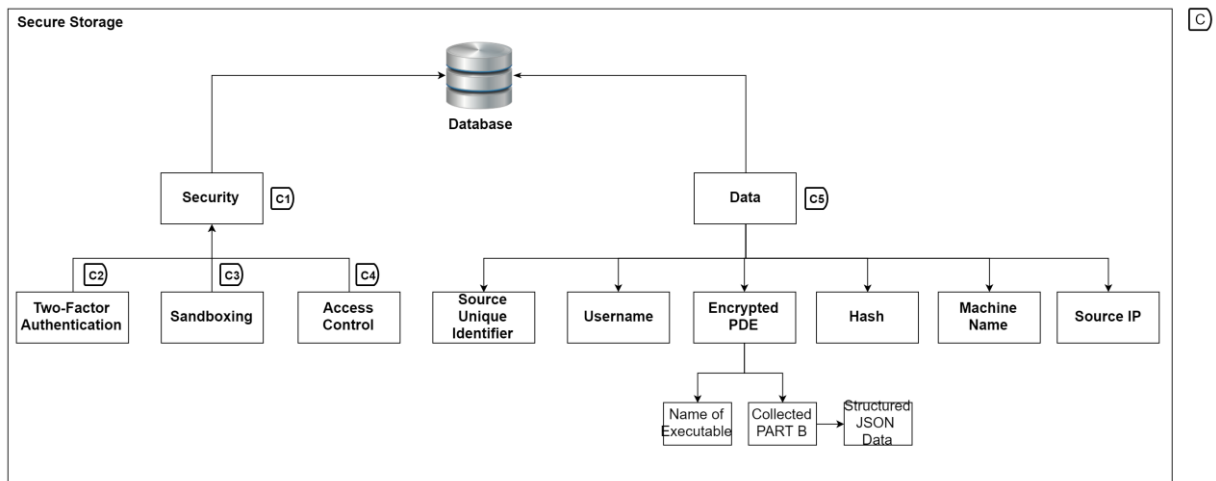


Figure 4-8. Part C - Overview of Secure Storage Phase

## 4.4.1 Security

To ensure security (C1) of the data and the storage process, the integrity of the data needs to be maintained at all times. For this, the best practices and security mechanisms have been integrated into the system, using verbose logging and secure access processes. The composition and discussion of the security setup adopted for the secure storage phase are presented next.

### 4.4.1.1 Two-factor authentication

In the current digital age, it has become easier for hackers to phish users and use social engineering techniques to trick users into divulging their credentials [142] [143]. This susceptibility, therefore, necessitates the need for a multi-factor authentication (MFA) process [52] [142]. Enabling Two-Factor Authentication (2FA) (C2) on the storage system further foils attackers from brute-forcing passwords and login details. Support for various types of 2FA can be considered, for example, an authentication token sequence that gets randomly generated after a few seconds by adding a QR Code to an app like Google Authenticator or Authy [144]. Hardware authentication can also be factored into the storage process to further protect data in the system.

### 4.4.1.2 Sandboxing

By incorporating a sandboxing (C3) strategy, the storage system can be isolated from the host machine. This is achieved by running the secure storage system in a virtualised environment, thus making it more stable and improving security because privileges are more restricted, and the entire system can run in isolation. In addition, the virtualised platform provides a medium for stable-process implementation because it is decoupled from the OS itself. Sandbox virtualisation is adopted for this process. Utilising a sandbox environment prevents unauthorised access from low privilege escalation attacks and insecure system setup. Therefore, having the

storage system run in a virtualised environment provides the ability to remain stable whilst ensuring a degree of security.

#### 4.4.1.3 Access control

In any system, access control (C4) is necessary to allow for users to have different privileges and features. For example, since this system is intended only for admins and investigators, the privileges of an admin will be more than the privileges of an investigator. The investigator may be a third party and will therefore not be able to change anything. However, since everything is logged, it is easier to corroborate evidence in the case of data having been tampered with.

#### 4.4.2 Data

In this phase, the data (C5) that is collected, along with some metadata from the system, is stored with integrity verification. The detail collected, along with the integrity measures incorporated is discussed next.

- a) **Unique source identifier** – this identifier is used to determine the source from which the stored information originated. This can be verified by using device identifiers such as an API key, MAC address and serial numbers.
- b) **Username** – this is the username of the user account on the system from which the data originated.
- c) **Encrypted PDE** – the collected information from Part B is collated into a structured data format and then encrypted with symmetric key encryption to further protect the data from unauthorised use.
- d) **Hash** – a hash is generated once the data has been added to the secure storage by calculating the hash value prior to storing the PDE to disk. This further ensures the integrity of the data.
- e) **Machine name** – this is the name of the machine from which the data originated, since there can be multiple users on one machine. It is used in conjunction with the username to identify a person in the case of litigation.
- f) **Source IP** – this is the IP address from the data originated and it ensures integrity through verification of known IP addresses.

This secure storage phase of the framework provides a baseline for evaluating the forensic soundness of evidence. It serves as a platform to evaluate the evidential weight of potential evidence, as well as for un-encrypted ransomware-related forensic data.

## 4.5 Conclusion

Chapter 4 presents the various components of the proposed ransomware readiness framework as a novel approach for ransomware readiness and investigation. To complete the picture, the entire framework is depicted in Figure 4-9. This composition begins with the introduction of the identifying scenarios and potential digital evidences, followed by the methods and items that are to be collected (which



may have potential digital evidences). Lastly, this collected information is securely stored for an investigator to perform further analysis if needed.

This concept of ransomware forensics is a complementary approach to ransomware investigation and attack prevention. To reflect on the research questions that were proposed in this study, the RRF framework answers Q1 and Q2. In Q1, it was asked whether a ransomware readiness framework could be created and to what degree can it be implemented. The second question enquired what potential digital evidence can be collected during a ransomware attack. The RRF framework addresses both questions in detail in the Parts A and B respectively.

The next two chapters focus on the proposed prototype tools and their respective models that support and implement the proposed RRF framework. For this purpose, two proof-of-concept tools were developed. The first tool forms part of the Identification (A) and Collection (C), while the second tool forms part of the Secure Storage (C).

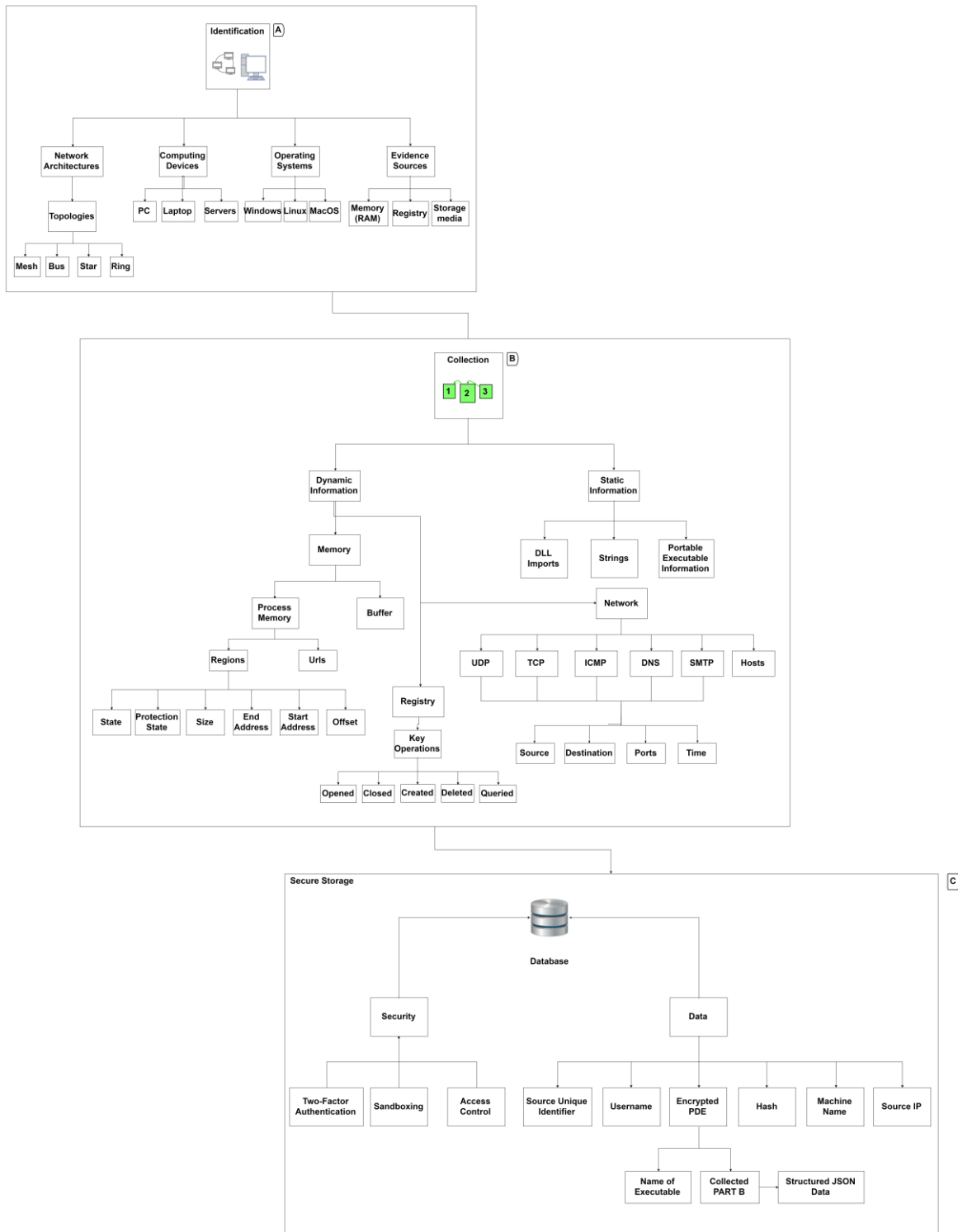


Figure 4-9. Second layer high-level view of RRF

## 5. CHAPTER 5: WINDOWS REGISTRY AND RAM COLLECTOR (W2RC)

### 5.1 Introduction

In this chapter, a proof-of-concept prototype system was developed to realise the proposed Ransomware Readiness Framework (RRF). The developed system was split into two parts (tools): Windows Registry and RAM Collection (W2RC), and Windows Registry and RAM Readiness Storage (W3RS). The mapping between the prototype and the ransomware readiness framework is shown in Figure 5-1. W2RC addresses Part A and B of the ransomware readiness framework and W3RS addresses Part C.

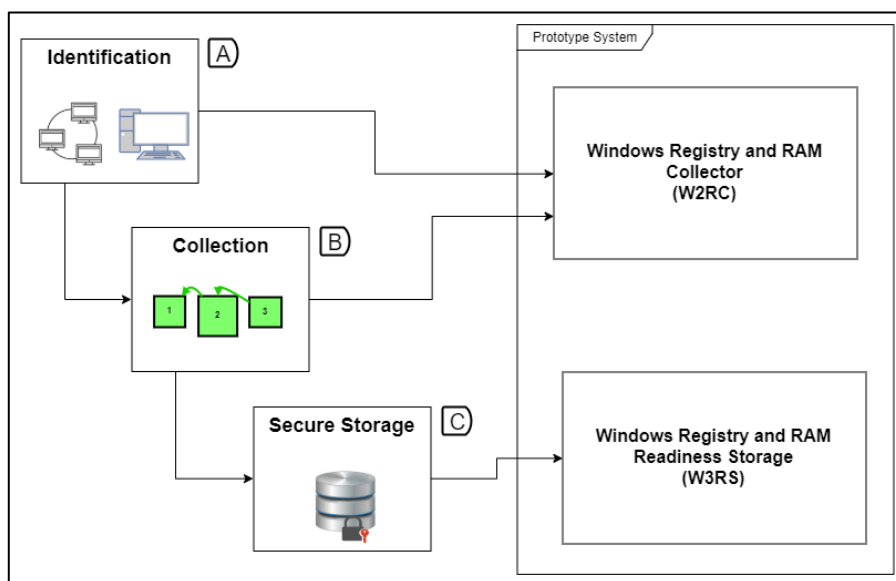


Figure 5-1. RRF mapped to the prototype system

The next section describes the model and criteria that were employed to successfully detect a ransomware attack and collect information from identified sources. This is followed by a discussion of the architectural design of the tool, the requirements specification for the proposed collection tool, as well as the implementation details.

### 5.2 Windows Registry and RAM Collector (W2RC)

The overall model adopted for the tool development process is presented in Figure 5-2. Note that the particular block number, as shown in this Figure 5-2 is indicated in brackets for easy reference to the figure. The tool starts by identifying the computing device (1), in this case, a personal computer (PC). The next step is to detect a Windows Operating System (2) in which this tool will run. The tool is developed to function in any network architecture, by using IP-based communication with a client-server model that supports many architectural designs. To be able to detect malicious behaviour, processes (3) are used.

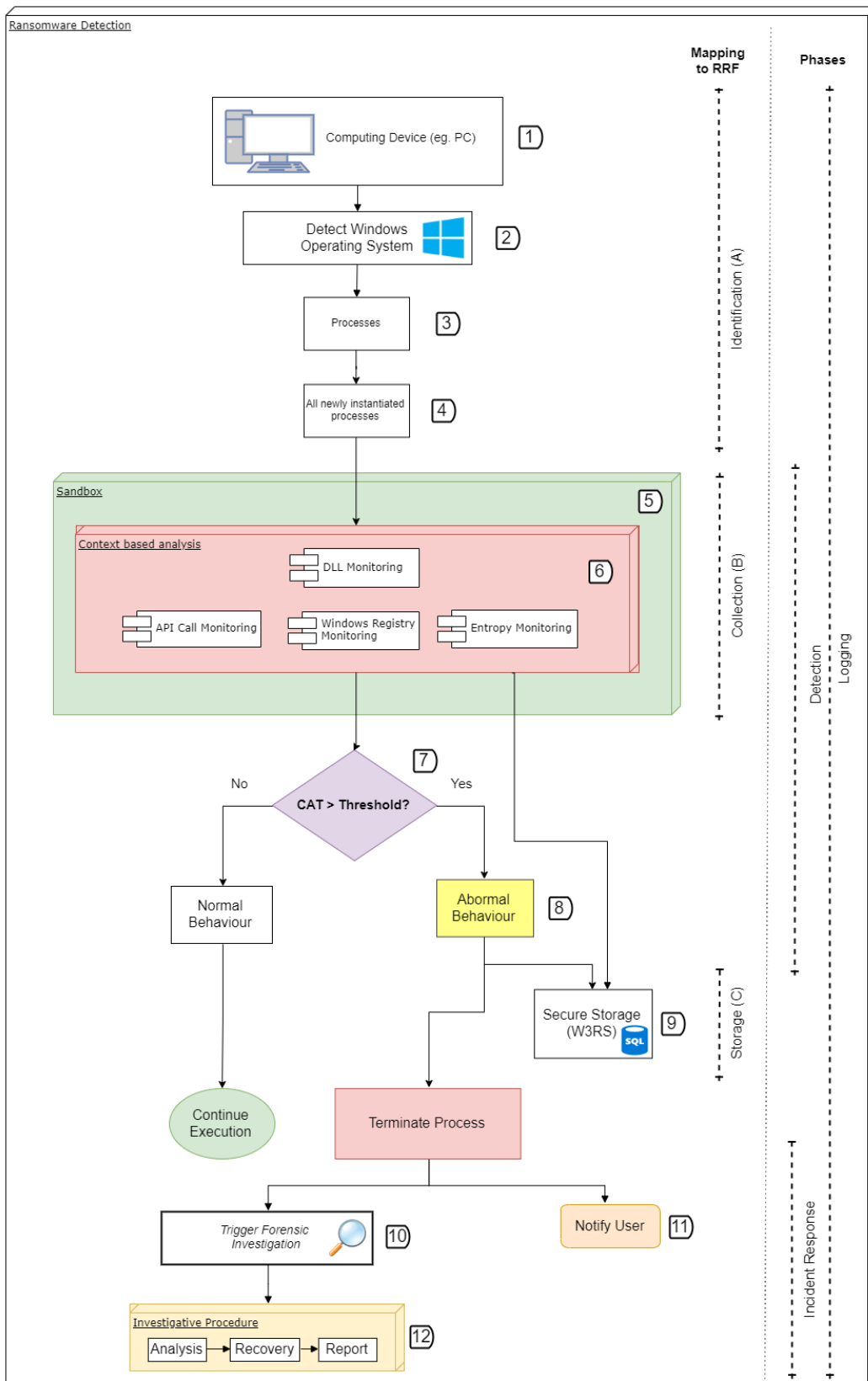


Figure 5-2. Model for ransomware forensics

To prevent unnecessary analysis and reduce the time taken to detect malicious behaviour, only newly instantiated processes (4) are monitored from the time the tool is installed. This is to ensure that the process is not ignorantly omitted. Such an

approach could potentially prove to be inefficient as it involves the continuous scanning and maintaining of the processes that are to be analysed, thus reducing the overall performance. However, this approach was used to convey the proof-of-concept of the prototype and did not necessarily focus on the performance at this stage. The tool has the ability to create a list of 'safe' processes that are taken from a clean installation of the Windows environment. From the time the tool is installed, a list of seen processes is also monitored; thus, a process is monitored only if it has changed. Steps 1 to 4 form part of the identification phase, which maps directly to Part A of the Ransomware Readiness Framework (RRF). Once a new process has been identified, it is quickly suspended in the system while the processes executable is forwarded to a sandbox environment (5).

The model makes use of a sandbox environment to ensure that the process to be executed does not have permanent effects on the system itself. For example, if it is a ransomware process that is executing, the encryption of the files will only occur in the sandboxed environment and the system will be guarded against any infections. This is possible since the executable is run in a controlled virtual environment with snapshots (an image of the VM) so that the effects of a potential malware infection can easily be contained when analysis is performed. Any infections can eventually be removed by restoring a previous snapshot. To achieve this, the tool makes use of the Cuckoo sandbox [104], which allows the easy execution of files through an Application Programming Interface (API) as well as by performing analysis. The Cuckoo sandbox was chosen to determine the behavioural characteristics of the process whilst ensuring the integrity of the system. This sandbox also allows custom analysis to be conducted, as shown by Context-Based Analysis (6).

In order to evaluate, quantify and analyse the behavioural characteristics of a process, several methods of Context-Based Analysis (CBA) (6) were taken into consideration. CBA is defined in this study, as the procedure of performing analysis based on the state of the machine, while exploring different areas of the OS with respect to a particular malware under investigation [24] [145]. The CBA involves the monitoring of dynamic link libraries (DLLs), API Calls, the Windows Registry, as well as process entropy. From this CBA, behavioural signatures from the malware under analysis are extracted and compared to known malicious activity. The CBA is explained later in Sections 5.2.1 and 5.2.4. An initial incremental database (a database that grows over time when each sample is analysed) of the known signatures is developed through an experimental process that helps prevent unnecessary analysis of samples that are already marked as safe – such as known system processes. The experimental process involves close observation of patterns that can be seen prior to, during and after a ransomware attack. These patterns include the DLL Monitoring, API Call Monitoring, Windows Registry Monitoring, and Entropy Monitoring. DLL Monitoring involves monitoring loaded DLLs to find commonalities in libraries that malicious samples use. API Call Monitoring analyses function calls to determine what the processes are doing in terms of functionality. Windows Registry Monitoring performs analysis as to what registry events are

occurring and the intent of these registry events. Steps 5 to 6 form part of the collection phase of the model, which maps to Part B of the RRF.

Steps 5-8 form part of the detection phase of the model. After the CBAs have completed, an average value is computed and forms the Context-Aware Trigger (CAT) value that quantifies the result to determine if the process is malicious. In Step 7, a decision is made based on the CAT value. If the CAT value is less than a calculated threshold, it would indicate normal behaviour, and the process will be resumed from the suspend state on the user system and continue executing. However, if the CAT value is greater than the threshold, it is flagged as abnormal behaviour (8). From here, two events occur concurrently: the results from the analysis phase are sent to the Secure Storage (W3RS) (9) prototype for storage, and the process is terminated. Step 9 maps to Part C of the RFF and is discussed later in Chapter 5. After the process has been terminated, another concurrent event happens where the user is notified (11), and a forensic investigation is triggered (10). From Step 9 onwards, the traditional investigation procedure (12) is carried out, as explained in Section 2.3.

The main processing of this model comes from the CBA processes. The detail of each CBA process is explained and the metrics of how it is calculated is presented in the next subsections. The CBA processes start with DLL Monitoring, then follows with API Monitoring, Windows Registry Monitoring, and Entropy Monitoring. The results of each CBA are discussed later in Chapter 7, where the results are computed and explained in detail.

### **5.2.1 DLL monitoring**

In an OS, when a process is created, several libraries are loaded into the volatile memory. These libraries are required by the process for successful execution. They are used to prevent unnecessary code duplication, thereby reducing the size of the program and the time it takes to operate. Libraries are typically optimised to load faster and provide optimised code for added efficiency. They are used in conjunction with the system in order to allow the process to perform the assigned task [146]. The libraries are loaded depending on the requirements and imports defined by the executable. These requirements are generally found in the executable's PE section, which is a specific structure that executables have to aid successful execution by the OS dispatcher. DLLs are generally protected against reverse engineering using static analysis. This is because attackers modify libraries to perform malicious tasks on their behalf by overwriting some libraries. However, it is possible to determine the names of the loaded libraries at the relevant addresses in memory, as well as some of the function calls to the library once the process is running (dynamic analysis). DLLs can also be loaded dynamically by other libraries at runtime or even from the executable itself. This feature is used by attackers to dynamically inject malicious libraries into safe executables, thereby corrupting them or giving the attacker control of the process that is executing.

Ransomware developers typically utilise the built-in cryptographic libraries within a Microsoft OS to reduce the effects of incompatible cryptographic functions and portability issues that may arise. By monitoring the loaded DLLs, an indication of the behaviour of the process can be examined and extracted. The common libraries that are generally used in a ransomware attack are shown in Table 5-1. DLL monitoring provides little insight towards making a definitive decision to determine the malicious behaviour of an application. This is because an application will make use of some Windows-specific libraries that provide the same functionality. For example, a kernel32.dll allows user-mode access to the kernel. Furthermore, since these cryptographic libraries can be used by most benign encryption programs, indicators for the malicious process are difficult to establish. However, it is theoretically possible to determine whether a program has the potential to perform encryption. This process can therefore provide some background as to what the process might be doing.

**Table 5-1.** DLL cryptographic commonalities for ransomware

<b>DLL Name</b>	<b>Explanation</b>
cryptbase.dll	Microsoft Cryptographic library that performs the most commonly used encryption algorithms
kernel32.dll	Handles memory management, provides 'user mode'" access to the kernel
kernelbase.dll	Gains Kernel-level functions and provides the ability to gain administrator rights to perform restricted operations
wow64cpu.dll	Switches the processor from 32-bit to 64-bit and vice versa
advapi32.dll	Microsoft Windows package for cryptography
shell32.dll	Executes shellcode within the operating system
msvcrt.dll, msvcp60.dll	Microsoft C Runtime Library module
cyptography.dll	Cryptography processes, procedures and extended samples of C and Visual Basic programs, using CryptoAPI functions and CAPICOM objects
Bcryptprimitives.dll	Crypto library for performing the bcrypt hashing algorithm
CRYPTSP.dll	Another crypto library for key generation and verification

From the commonalities shown in Table 5-1, only libraries that contain the word 'crypt' generally provide some sort of cryptography. Each of these libraries contains several cryptographic functions that are made available to the process. Therefore, Equation (5.1) can be used to characterise the DLL monitoring analysis. DLL

monitoring (DM) is the summation of the number of cryptographic functions called, divided by the summation of the number of cryptographic libraries (cl).

$$DM = \frac{\sum cf}{\sum cl} \quad (5.1)$$

In order to extract the necessary information, the *ListDLLs* [147] and Cuckoo DLL analysis package [104] were used. DM proposed in this study provides additional metadata for detection the level of encryption functionality in a given process. DLL monitoring maps to the static information of Part B of the proposed framework.

The next subsection expands on API monitoring and how it is achieved. The monitoring of API's provides meaningful interpretation to the analysis, as it monitors the function calls of the process in execution.

### 5.2.2 API call monitoring

The monitoring of the sequence of function calls can give more information as to what a program in execution is doing, thereby providing the ability to extract and compare signatures to known malicious activity [148]. API monitoring can also provide significant information because all function calls to libraries and operations can be seen, thus it is possible to detect malicious call sequences. However, API monitoring is a trivial process, as malware can use detour and trampoline functions to avoid detection through pattern matching.

Trampoline functions are functions that are used to pass on information to another function or perform a seemingly redundant operation so as to avoid detection. It can also be tricky if a program has protection against process hooking, for example through administrative privileges, thus terminating the API monitoring hook/process [9]. Jung and Won [9] proposed the monitoring of API sequences in terms of kernel logs by using the time difference between each log entry to detect detour and trampoline functions. Since these functions simply pass on the information to the actual destination function, little processing is needed. Seeing that there is no human interaction, the time difference could be relatively insignificant. However, attackers can easily manipulate this time difference by performing unnecessary computations or sleep functions to increase the time differences. This implies that the approach suggested by Jung and Won would not be able to detect this. Besides the difference in the time sequence, this study observed the ratio of the number of function calls to the number of loaded libraries.

Based on these observations, API monitoring analysis can be redefined as expressed in Equation (5.2): the total number of function calls ( $nf$ ), divided by the change in the time between API calls ( $\Delta c$ ), multiplied by the total number of libraries ( $N$ ). This gives an overall quantifiable value that can help establish what an executable is doing to determine its malicious nature. This ratio can also be used to



observe the number of functions the process uses for each library and so explain the operating process through its dynamic behaviour. API monitoring encapsulates the majority of the dynamic information in Part B of the proposed framework. This includes monitoring the API calls from memory, as well as analysing and collecting information from both process memory and memory buffer.

$$AM = \frac{nf}{\Delta c \times N} \quad (5.2)$$

The next CBA process is Windows Registry Monitoring, which involves analysing Windows Registry function calls and operations that occur related to the registry.

### 5.2.3 Windows Registry monitoring

The Windows Registry is a vital part of the Windows OS, as it provides a central data repository for all operations in the OS [23] [149]. Ransomware exploits this by manipulating and creating registry keys to make the effects permanent [100]. This is done to prevent ransomware deletion through a user reboot by creating a registry key that can automatically start the ransomware upon a system start-up. By monitoring how the registry keys are queried, created and deleted, it gives more context as to what a given executable is doing [101]. From the frequency of registry changes that occur over time, a deduction can be made to determine malicious intent.

Equation (5.3), which can be used to define a context-based windows registry monitoring process, is defined as the number of registry events per second (*reps*), multiplied by the summation of all the keys deleted (*kd*), divided by the summation of all the created keys (*kr*), plus the number of queried keys (*kq*), multiplied by the summation of keys closed (*kc*), divided by the number of keys opened (*ko*). To the best of the researcher's knowledge, no research has so far been done to isolate or give registry monitoring preference over other forms of monitoring. However, registry monitoring provides meaningful information as to what a process is doing and what I/O operations are being performed. In comparison to Jung and Won's [9] work, registry monitoring provides a measure of how much system information is being modified and queried. This is helpful to detect persistence in ransomware as well as to provide the ability to detect abnormal registry interactions. In the system-specific analysis of the keys that are related to the system, such information is generally found in the HKEY\_LOCAL\_MACHINE hive. Registry monitoring maps directly to the registry section of Part B of the ransomware framework.

$$RM = reps \times \frac{\sum kd}{1 + \sum kr} + kq \times \frac{\sum kc}{1 + \sum ko} \quad (5.3)$$

The final CBA process involves monitoring the amount of randomness to detect if information is likely to be encrypted. Further details on entropy monitoring are discussed in the next subsection.

#### 5.2.4 Entropy monitoring

In information theory, entropy is defined as the amount of uncertainty or randomness in producing information [9]. Entropy has been a major factor in encryption, due to the amount of random data that a computer can generate. The main disadvantage from a security point of view is that entropy is probabilistic based and therefore can be predicted if given a sufficient number of input data samples. Since entropy is deterministic, it can also be represented mathematically. Generally, entropy is represented by Shannon's Entropy, as expressed in (5.4).

$$H(X) = \sum_{i=1}^N P x_i I x_i = \sum_{i=1}^N P x_i \log_2 \frac{1}{P x_i} = - \sum_{i=1}^N P x_i \log_2 P x_i \quad (5.4)$$

In Equation (5.4),  $H$  denotes the entropy value, and  $X$  is a set with possible values. Furthermore,  $N$  is the number of values,  $P$  is the probability of the occurrence of the value, and  $I$  describe the amount of information. When the  $H(X)$  results in 0, it implies there is no uncertainty. Conversely, when the result is at the peak of certainty, it will result in  $\log_2 N$ . This indicates that the probability that such information exists is only  $1/N$ . High entropy means prediction is less possible.

Encryption has the property to change from low entropy before encrypted to a high entropy. The entropy value, in this case, represents the minimum number of bits per character. In order to perform entropy monitoring, two approaches were explored. The first approach was to generate the entropy value for the entire executable, and the other was to calculate the entropy value of the PE (Portable Executable) sections of an executable. The first approach is computationally expensive for larger file sizes, whilst the other approach is relatively inexpensive. However, the second approach relates to performing some static analysis. The entropy of each PE section is considered if the executable is encrypted or has secretly hidden parts. In Equation (5.5), entropy monitoring ( $EM$ ) is defined as the sum of the entropy of the executable ( $Ex$ ) and the average sum of the entropy of the PE sections ( $PES$ ).

$$EM = Ex + \frac{\sum_{i=0}^{Npes} PES}{Npes} \quad (5.5)$$

Now that all the CBA process have been discussed, the notion of a Context-Aware Trigger is proposed, and its details are presented in the next subsection.

#### 5.2.5 Context-Aware Trigger

Context-aware technology comes from a behavioural analysis and incorporates the environment and the current working of the system. In a ransomware scenario, the key aspects that can be used to identify ransomware were earlier discussed in Sections 5.2.1 to 5.2.4. By combining these monitoring techniques and dividing them by the number of techniques ( $\dot{N}$ ), the study developed a context-aware trigger

(CAT) mechanism for ransomware forensics. This can be represented mathematically, as expressed in Equation (5.6). The results of this CAT are presented in Chapter 7.

$$CAT = \frac{DM + AM + RM + EM}{N} \quad (5.6)$$

The model concludes with the investigative procedure presented in the next subsection. This process follows the traditional investigation process where an incident is detected and PDE is provided for an investigator to corroborate evidence and make it admissible in a court of law.

### **5.2.6 Investigative procedure**

From Step 12, the output of the CAT process is used to evaluate the need for a forensic investigation. In cases where the forensic investigation is evoked (a positive detection), the logging and analysis processes will be securely processed and stored whilst the user and forensic investigator are notified to conduct the traditional investigative procedure. The process is then terminated. The data collected is extracted by the investigator to perform analysis and generate a report. For ease of analysis, the data is structured using a JSON data exchange format suitable for any application. Depending on the severity of the attack, a recovery plan can be put in place by the investigator through manual analysis.

In order to evaluate the proposed proof-of-concept tool, a system requirements specification was drafted based on software engineering principles and forensic processes. The next section highlights the specifications of the testing processes following the Computer Forensics Tool Testing (CFTT) Program [\[150\]](#) of the National Institute of Standards and Technology (NIST) [\[151\]](#).

## **5.3 W2RC system requirements specification**

This section discusses the requirements specifications of the developed collection tool (W2RC). In order to provide a high-quality and useful proof-of-concept prototype, it is essential that requirements specific to digital forensics be drafted to define the integrity and the process of how the tool performs and operates. This also helps when evidence sourced from this tool is to be used in a court of law, since it conforms to the requirements of the various forensic processes. These requirements were derived from the proposed model in Figure 5-2 and are divided into two categories: Secure Collection Core Requirements (SC-CR) and Secure Collection Optional Requirements (SC-OR). The SC-CR states the necessary requirements to develop the system and provide the base functionality and capability of the prototype. The SC-OR sets additional (optional) requirements that will add value to the prototype but are not essential for it to work correctly. This

method for requirements specifications was derived from the NIST CFTT [150] program as part of their validation cycle. The validation of the system is presented in Chapter 8, where more details on the NIST CFTT are discussed, as well as how the prototype system itself was validated. However, it is necessary to now define the system requirements prior to the system implementation in order to gauge whether all the core requirements were satisfied and whether the system does what it is supposed to do.

### 5.3.1 Secure Collection Core Requirements (SC-CR)

- **SC-CR-01:** The tool shall monitor all running processes.
- **SC-CR-02:** The tool shall perform logging at every action/process that occurs.
- **SC-CR-03:** The tool must collect information concurrently.
- **SC-CR-04:** The tool must show consistency in the collection.
- **SC-CR-05:** The tool must suspend a new process once detected.
- **SC-CR-06:** The tool must efficiently collect the executable and send it for analysis.
- **SC-CR-07:** The tool shall perform quick data processing and data handling.
- **SC-CR-08:** The tool must show what processes are currently being analysed.
- **SC-CR-09:** The tool must provide hashes of the database to ensure integrity.
- **SC-CR-10:** The tool must distinguish between previously seen processes and newly created processes.
- **SC-CR-11:** The tool must list the time and the name of the process it found.
- **SC-CR-12:** The tool must securely send the collected information to the storage server.
- **SC-CR-13:** The tool must work on all Windows NT platforms.
- **SC-CR-14:** The tool must work on 32-bit and 64-bit systems.
- **SC-CR-15:** The tool must request administrator privileges.

### 5.3.2 Secure Collection Optional Requirements (SC-OR)

- **SC-OR-01:** The tool must allow a user to resume a process.
- **SC-OR-02:** The tool must allow a user to add safe processes.
- **SC-OR-03:** The tool must provide the status of the analysis of the process.
- **SC-OR-04:** The tool must provide an option to separate the storage server from the analysis server.
- **SC-OR-05:** The tool must provide the ability to see the CAT value of analysed processes.
- **SC-OR-06:** The tool must display notifications when a process is being analysed.
- **SC-OR-07:** The tool must remove a process from a seen and whitelisted database.

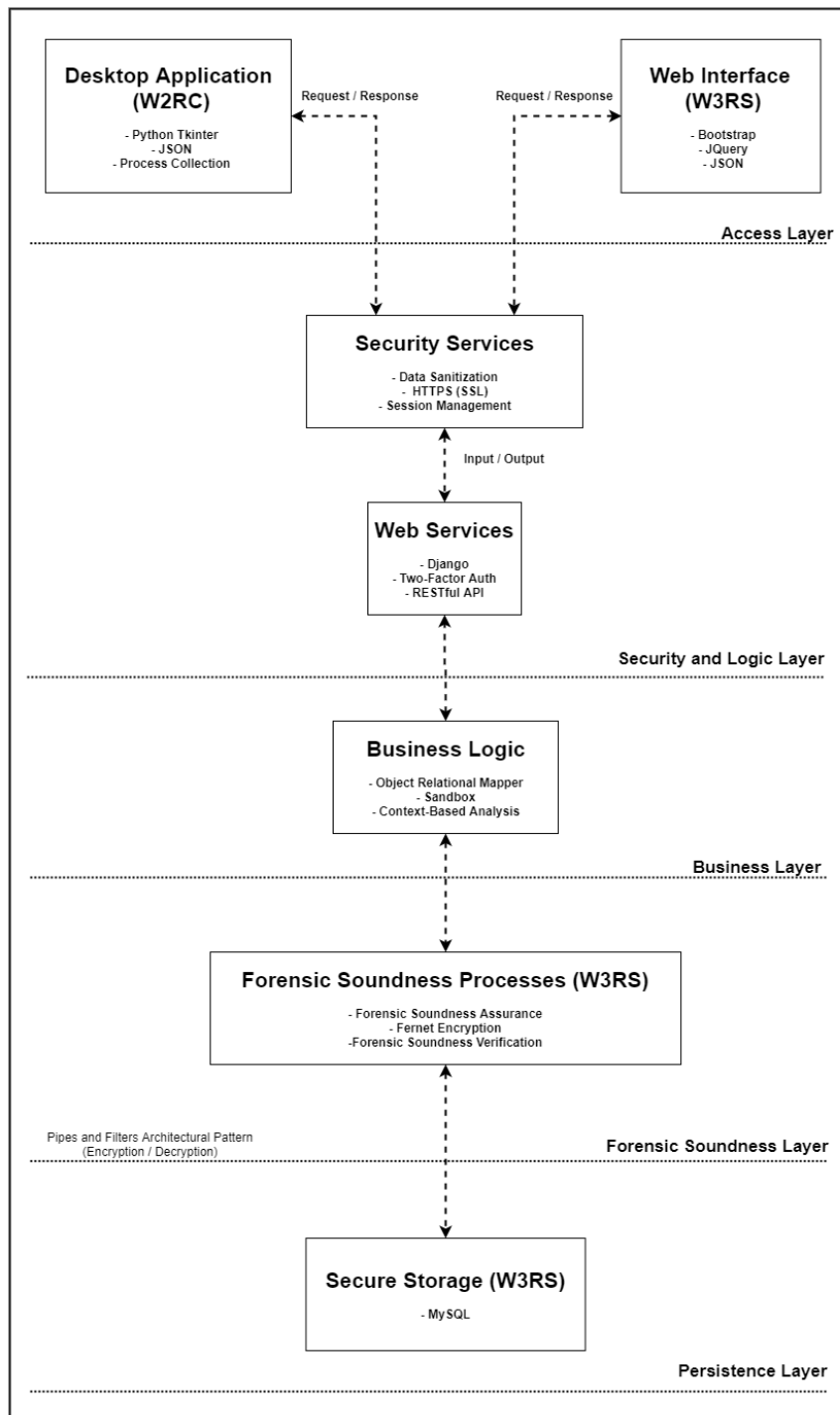
Now that the requirements have been defined, the next section presents the architectural design of the entire prototype system. In order for the prototype to have more value, the architecture needs to be defined in such a way that the steps and design of the prototype can be repeated, thus rendering the process scientific.

#### **5.4 Architectural design**

The architectural design of a system refers to the structure in which a system will operate, showing all the components that are involved in the system, as well as some of the technologies used [152]. There are several types of architectures in which these components interact and are structured. For the proposed prototype system, a 5-tier layered architecture with Model View Controller (MVC) as well as pipes and filters were used. A layered system provides separation of concerns and separates components between the various layers. The MVC was used to connect the application interfaces to the business logic. This was done so that the application interfaces would not interact directly with the back-end storage or business logic. This technique provides additional security as there is a layer in between that restricts access. Pipes and filters are used when the output of a process is the input to another process, and this is mostly used for encryption and authentication.

The architectural design of the system is shown in Figure 5-3. For each of these layers, some technologies are used that are only discussed later in this chapter (see the implementation phase in Section 5.5). Based on the proposed architecture, the access layer provides the interfaces that an end user would use to access the system. In this case the two interfaces were the desktop application of W2RC and the web interface of the W3RS tools. The next layer is the Security and Logic layer where the logical processing of information and operations as well as security processes are conducted. In this layer, there are two services, namely security and web services.

Security services focus on data sanitisation where the cleaning of the data is performed and removing basic attacks like SQL and XSS injection. This layer also provides the HTTPS protocol with session management. The web services are the web technologies and frameworks that were adopted for this system, namely Python's web framework Django, two-factor authentication, as well as RESTful services. These web technologies are discussed in Section 5.5 at the implementation stage of the tool. The next layer is the business layer where the business logic exists, and the context-based analysis takes place in a sandboxed environment.



**Figure 5-3.** Architectural design of the proposed system

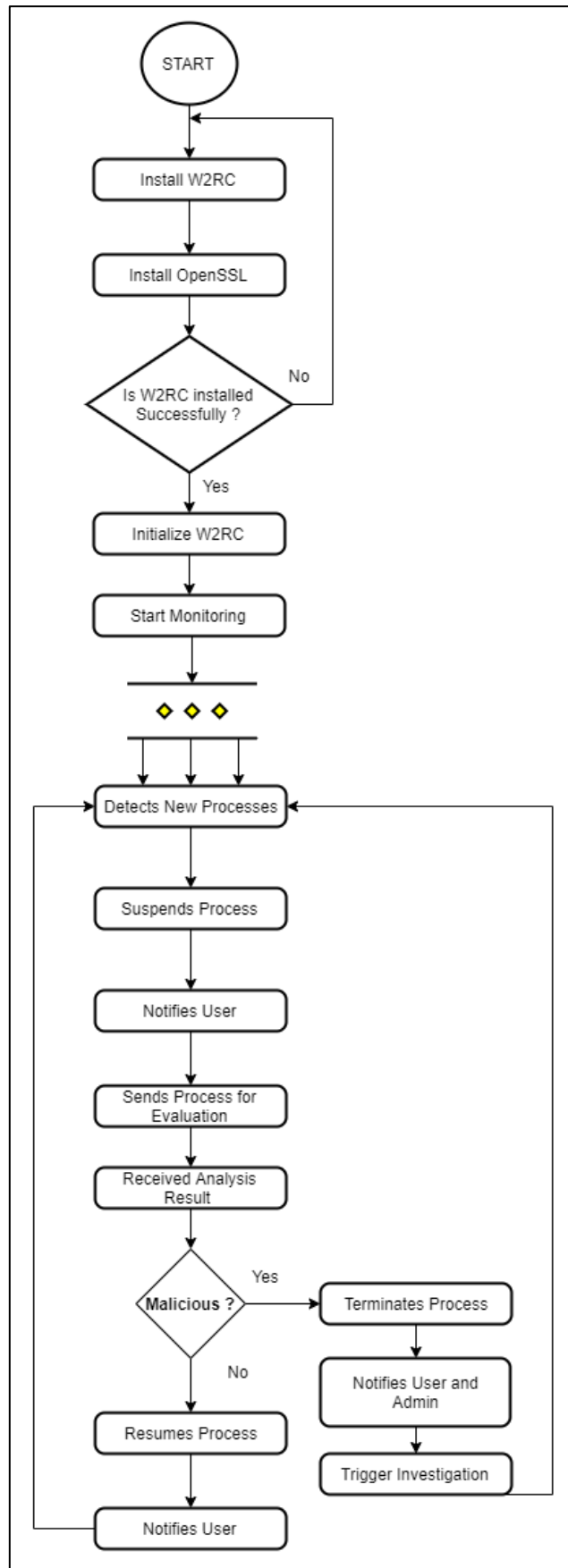
The fourth layer, which comprises the forensic soundness layer in which the forensic soundness processes exist, is presented and discussed later in Chapter 5. In this layer, another architecture called ‘pipes and filters’ is used to achieve fernet encryption and the forensic soundness processes (also discussed in Chapter 5). The last and final layer of the 5-tier architecture is the persistence layer which focuses on the storage of data that was collected. This layer encompasses the secure storage where the data is stored in a MySQL database.

Following the description of the requirements and architecture, the next section presents the system implementation and the technologies used to achieve the purpose of the prototype W2RC tool. The system was implemented in accordance with the above requirement specifications, attempting to address all the requirements proposed whilst ensuring compliance with forensic processes, best practices, as well as standardisation.

## **5.5 W2RC system implementation**

Given the above requirements, the system was implemented using a modular approach, by using Python [153], as the coding language of choice. Python was used because of its ease of usage as well as its ability to be easily packaged. The tool uses the TkInter library [154] for creating a Graphical User Interface (GUI) of all of the components. The tool was packaged into a Windows Installation Package file, also known as an MSI file, for ease of installation. The installation of the OpenSSL library [155] was required in order to establish a secure connection between the storage server and the analysis server. (The source code, user and installation guide can be found in Appendix B.) The installation guide also provides a quick summary of how to use the tools. The tool allows a user to modify the analysis server and storage server IP address location in the event that the analysis and storage servers are not the same. This was done to allow more control of where the data would be stored in the event that the sandbox server environment had to be separated from the storage due to organisational policies or security concerns.

The lifecycle of the W2RC tool is shown in Figure 5-4. The user interface of the W2RC is shown in Figure 5-5. The lifecycle of the tool starts off by checking if the W2RC system is installed and if OpenSSL is installed. The next step is 'Initialise W2RC', which is where the tool analysis server and storage server IP addresses need to be specified. Once the user clicks on the "Start monitoring" button, the monitoring process is initiated (see Figure 5-4). When the system detects new processes, a thread is created allowing concurrency and preventing the system from waiting before the process is sent for analysis. This is done so that new processes can always be detected with faster accuracy. The user interface for this process is shown in Figure 5-6. The next step, 'Send the process for evaluation', will inform the user that the process has been suspended and that it is being analysed. A view of the user interface of the tool is presented in Figure 5-7. After the analysis has been performed, the result is sent back to the tool and it is determined whether the analysis process is malicious or not. If the process is not malicious, the process is resumed, and the user is notified that the process is safe. If the process is malicious, it is terminated, and the user and admin are notified to trigger an investigation.



**Figure 5-4.** Lifecycle of W2RC





## **5.6 Conclusion**

The implementation process of the proposed ransomware forensic framework was presented in this chapter. More specifically, the identification and collection phases of the framework were developed using the proof-of-concept tool W2RC. Novel techniques and approaches for the development of the proof-of-concept tool were also detailed in this chapter, but the results of these techniques will only be presented in Chapter 7. This chapter justified the novel idea proposed for ransomware forensics through the adoption of a digital forensic readiness approach. The validation of the implementation and impact of the model and proposed ransomware readiness framework will be evaluated in Chapter 8.

Chapter 6 constitutes the last section (Part C) of the proposed RRF. A process model for secure storage, following all security standards, is presented alongside implementation of the model which maps directly to Part C of the RRF.

## **6. CHAPTER 6: WINDOWS REGISTRY AND RAM READINESS STORAGE (W3RS)**

### **6.1 Introduction**

In this chapter, the second part of the proof-of-concept prototype was developed in order to realise the proposed framework. This part of the prototype, Windows Registry and RAM Readiness Storage (W3RS), addresses Part C of the Ransomware Readiness Framework (RRF). Recall from Figure 4-9 that, as discussed in detail in Chapter 5, Part A detailed 'Identification' and Part B detailed 'Collection' of the proposed RRF. Chapter 6 now makes up Part C and concludes the proposed RRF. Part C focuses on how the collected information is stored, abiding by Cyber Forensic Services discussed in Chapter 2. Forensic soundness is also ensured through integrity checks and backups.

Next follows a discussion of the W3RS part of the prototype and modelling of the system for secure storage.

### **6.2 Windows Registry and RAM Readiness Storage (W3RS)**

The high-level process model for Part C of the RRF is shown in Figure 6-1. The storage process consists of smaller subprocesses, namely data ingestion, forensic soundness assurance, PDE storage, and forensic soundness verification. A brief overview of these processes is provided first, followed by more detail in each of the relevant subsections.

The data ingestion process involves consuming structured data from various sources. The types of sources do not play a role in the ingestion process as the latter provides a medium for data being received. As long as the data is in a format that the process can interpret, any data can be consumed.

The data ingested is further parsed for forensic soundness assurance. The forensic soundness assurance process focuses on gathering and processing the information in such a way that no evidence is altered in an unauthorised manner. Thus, the integrity of the collected information is guaranteed, in compliance with evidence admissibility criteria. The assurance process is conducted in memory before any information is stored. This is to ensure that the data received that is in memory is the same before and after it has been stored, so that the integrity of the data remains intact, making it free from any modifications whatsoever.

The next process focuses on storing the PDE physically on the server in a forensically sound manner. During the PDE storage process, the data will be securely stored in an encrypted format within a secure folder on the server. The corresponding metadata is not yet stored in the database as the PDE needs to go through the forensic soundness verification process to ensure that its integrity is maintained.

The forensic soundness verification process next validates and confirms the forensic soundness of the collected PDE. Once this evidence has been verified successfully, the metadata is inserted into the database together with the location where the PDE is stored on disk. Since logging is performed during each process, it becomes a concurrent process. Logs are maintained for assurance and traceability purposes.

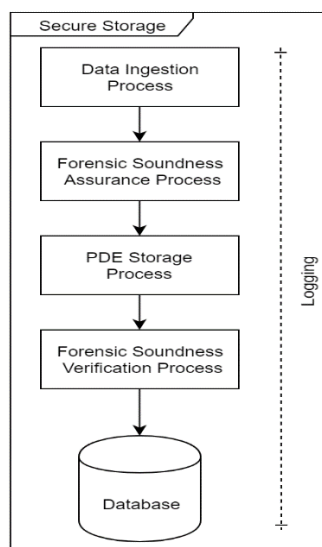


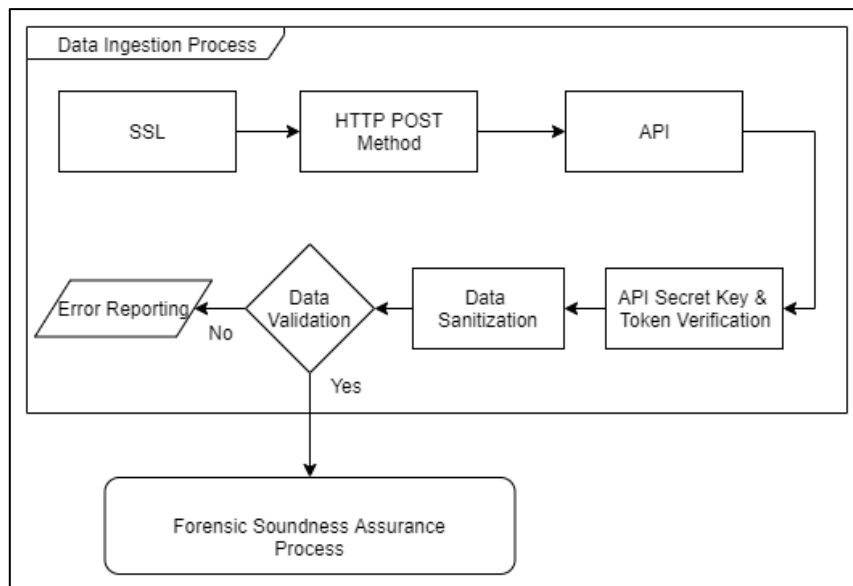
Figure 6-1. High-level process model of W3RS

Now that an overview of the entire process has been briefly sketched, each of these processes is discussed in more technical detail. The technical details of data ingestion process are presented next.

### 6.2.1 Data ingestion process

The details of the data ingestion process appear in Figure 6-2. This process makes use of a secure socket layer (SSL) for encrypted communication between the client and the server. Using an SSL is considered best practice by security standards in the sense that if data is intercepted (by a man-in-the-middle attack for example), it will be potentially unusable to the attacker because it has been encrypted.

There are two typically used HTTP methods for data exchange, namely GET and POST. The GET method relies on sending parameters through the URL using URL encoding. This renders the data being transferred clearly visible in the URL and an easy target for network sniffers (i.e. attackers that monitor the network activity looking for sensitive data like passwords). However, the POST method sends all the parameters in the body of the request. Allowing only the HTTP POST method and using SSL (denoted by 'HTTPS', i.e. 'secure' HTTP) ensures that the body of the request is encrypted.



**Figure 6-2.** W3RS data ingestion process

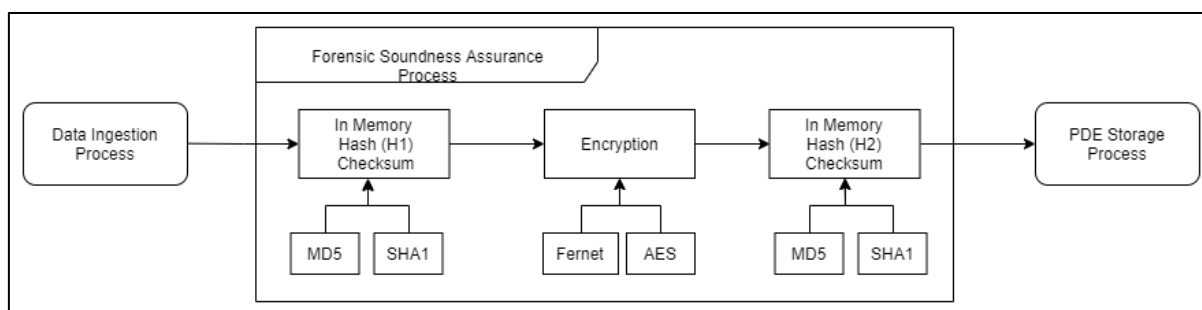
Typically, Application Programming Interfaces (APIs) are used to expose a designated route or web URL to which data can be transmitted and received. This is known as an API endpoint. The endpoint is available on the server so that data can be ingested and stored securely. In an attempt to make the ingestion process faster and more standardised, an API endpoint is exposed to the storage tool. An API key is used to ensure that only authorised parties are able to add data to the storage engine. This API key is comprised of two parts: a prefix and randomly generated characters (key). The prefix is used to identify the device from which the request is derived (i.e. the origin of the request). The randomly generated characters also act as an identifier. However, the main purpose is to ensure better security practices by using random characters to make a longer key length that will hold up against brute-force attacks. This API key has to be registered on the storage system in order to serve as a unique identifier. The system stores the API key in its two parts to ensure better security. The key itself is hashed using SHA 256 and stored in the database; thus, if the database is breached, the API keys are still secure.

When new data is ingested on the API endpoint, the request is first checked to determine whether the necessary access and authorisation are met (by validating the API key) and to ensure that the key has not been revoked. The status of the API key is set to 'revoked' if the key has been leaked or is no longer needed. Once an API key has been revoked, it can no longer be used, and any newly created API key cannot be the same as any revoked key. Thereafter, the data undergoes sanitisation to remove any malicious data, SQL or JavaScript injections and to prevent Cross-Site Scripting (XSS) attacks [87] [156]. This is done to prevent attackers from sending malicious payloads to the server because the API endpoint is known. Data sanitisation is a standard practice of any secure web application or server that deals with data processing [156] [157].

After the data has been successfully sanitised, the next process is data validation. In this process, data is parsed to ensure that the data structure and format is correct. This is important, as an unexpected runtime error can occur if an invalid character or structure is processed and the necessary attributes are not present. If the data does not pass the validation phase, the process is stopped, and error reporting is triggered. If the data was successfully validated, the data gets sent onto the next phase for forensic assurance.

## 6.2.2 Forensic soundness assurance process

The forensic soundness assurance process generates the relevant information to be used to prove forensic soundness of the data collected (in other words to prove that the integrity of the data is the same as the original data) [158]. The details of the forensic soundness assurance process are presented in Figure 6-3



**Figure 6-3.** W3RS forensic soundness assurance process

At this stage, this process is still conducted in memory, and nothing is stored in the database or on the disk as yet, because the process is performed to provide assurance that no data is modified upon receipt. To ensure forensic soundness once the data has been validated, an in-memory hash (H1) is calculated for the PDE using any checksum hash algorithm. MD5 and SHA1 are the two hashing algorithms that are most used as a checksum integrity measure in the industry. This is because of their efficiency in computing a hash – which is used as an integrity measure and not for storing passwords in cleartext (the hash function's other frequent use in the industry). These algorithms, mostly used because they are quick to calculate, are often called one-way hash algorithms, meaning that it is impossible to reverse the hashing process to get the original data back from the computed hash digest that was produced by the algorithm. Therefore, by taking the originally calculated hash (call it hash 1) and then again calculating another hash (call it hash 2) over data that was modified, the hashes would be distinctly different (i.e. hash 1 will not be equal to hash 2). Also, due to the amount of data being hashed, it is practically impossible for attackers to brute-force the hash, as they will not be able to get the original data from the hash.

After the hash has been generated, the next process is to secure the data collected by performing symmetric key encryption. Using symmetric key encryption is more

suites when there could be multiple users – in this case, admin users or investigators – because there is only one key and not multiple key pairs as in the case of asymmetric encryption. For this purpose, fernet encryption, which also uses AES encryption, was used based on best practices [159] [160]. The details of fernet encryption are further discussed in Section 6.4.

Once the PDE has been encrypted, the next step is to generate another in-memory hash (H2) of the encrypted PDE. This is done to make sure that the encrypted PDE was not modified at any point and serves as an integrity check. This H2 will be used as an input to the forensic soundness verification process, to be discussed in Section 6.2.4.

### 6.2.3 PDE storage process

In this process, the encrypted PDE is stored to disk. A detailed overview of the PDE storage process is shown in Figure 6-4.

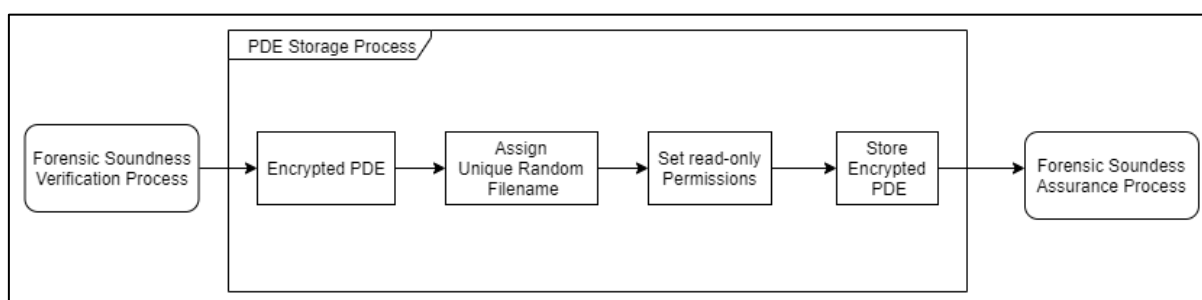


Figure 6-4. W3RS PDE storage process

First, the encrypted PDE is assigned a unique random filename to ensure that the system is immune to URL manipulation. If an attacker is aware of the naming structure used, for example, "report-date-number.json", it will be easy to try and download files using this structure by merely changing the number of the report through a brute-force approach that manipulates the URL. For example, if a report was named "report-2019\_10\_11-7.json", an attacker could brute-force the file name by trying to download other reports like "report-2019\_10\_11-9.json". Furthermore, a random filename prevents PDE from being easily identified by a system admin who could be involved in the criminal activity and tamper with the potential evidence. It also prevents an admin from being negligent, because without the metadata that is stored in the database, there would just be random file names with no relation to which user it came from. This metadata is only stored after the forensic soundness verification process has successfully completed. Permissions are added to the file, making them read-only on the system so that no process can change them. This ensures that the file cannot be modified easily and that it remains forensically sound. After the permissions are set, the encrypted file is securely stored on the disk within a directory that can be accessed only by an administrator. Although an admin can

change the permissions of the file or folder<sup>1</sup>, this does not have any effect because the hash integrity and logging measures are in place. Once the encrypted PDE has been successfully stored, it is ready for verification and integrity confirmation. Details of the forensic soundness verification process are presented in the next subsection.

#### 6.2.4 Forensic soundness verification process

Following the standard forensic practice of evidence storage and verification – i.e. maintaining a chain of custody, storing evidence in a secure environment and providing the original hash to verify that the evidence has not been tampered with – this subsection presents the forensic soundness verification process. This process ensures the integrity of the PDE by verifying that the information that was received in memory is equivalent to what is stored on disk. To confirm that the integrity of the stored PDE is intact, a hash process (H3) is computed from the stored encrypted PDE. This is to verify forensic soundness from the Encrypted PDE to the Store Encrypted PDE process (see Figure 6-5).

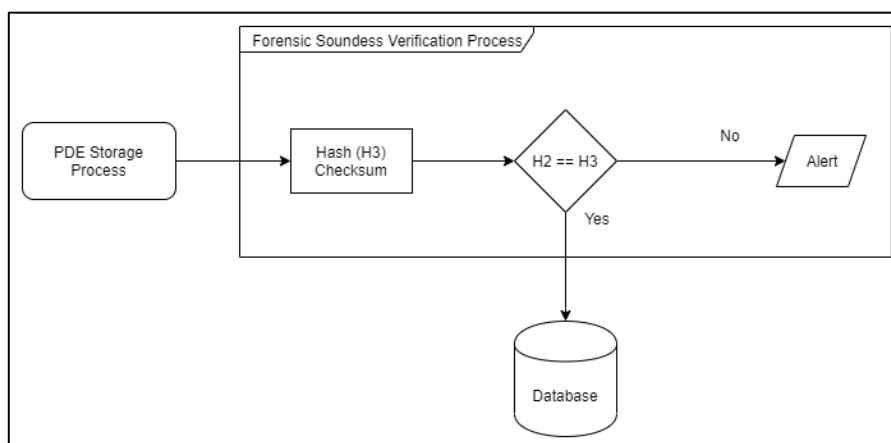


Figure 6-5. W3RS forensic soundness verification process

The in-memory hash of the encrypted PDE (H2) is compared to the newly created hash (H3) to determine if the hashes of H2 and H3 are still the same. If H2 and H3 are identical, no deliberate or accidental manipulation of the PDE occurred, and it is verified as forensically sound. Thus, this entry is inserted into the database. The entry contains the metadata and the location to the stored PDE, and not the actual PDE itself. The storage approach adopted in this study conforms to best practices, as storing a file in a database as binary data (blob) is extremely inefficient [161]. It also expands the attack vector (i.e. making it more difficult for an attacker to get access to the PDE) by separating the stored PDE from its metadata. For example, if an attacker manages to get unauthorised access to the database, but the PDE itself is not stored there, only metadata about the PDE can be extracted, which is not sufficient (on its own) for malicious intent (i.e. further exploitation by an attacker). If the hashes do not match, it can be assumed that an external party modified the

<sup>1</sup> Typically, this is a problem with any system, which is why administrators have to be trusted.



PDE during the process or that some other accidental situation (electricity spikes, bad disk sectors, etc.) occurred, thus invalidating the forensic soundness. Such an event will trigger an alert to warn system admins to investigate what could have caused the violation of the PDE's forensic soundness. This investigation is a manual process, as the violation would have occurred under unknown circumstances, and therefore it falls outside the scope of this research. The integrated process model illustrated from Figure 6-2 to Figure 6-5 for the W3RS proof-of-concept prototype is presented in Figure 6-6.

The tool was developed using agile software development methodology that involves an iterative, incremental development process. During the process, requirements and objectives are broken down into smaller milestones to achieve a big task effectively [162]. Following an agile software development process, the development of the requirement specification and usability function for the W3RS process model (see Figure 6-6) is considered in the subsequent sections. This is to ensure that the development process adheres to measurability checks, as well as to objective metrics of tool testing.

In order to evaluate the proposed secure storage proof-of-concept tool, a system requirements specification was drafted based on good practice software engineering principles and standard digital forensic processes. The next section highlights the specifications of the testing processes which follows the Computer Forensics Tool Testing (CFTT) Program [150] of the National Institute of Standards and Technology (NIST) [151].

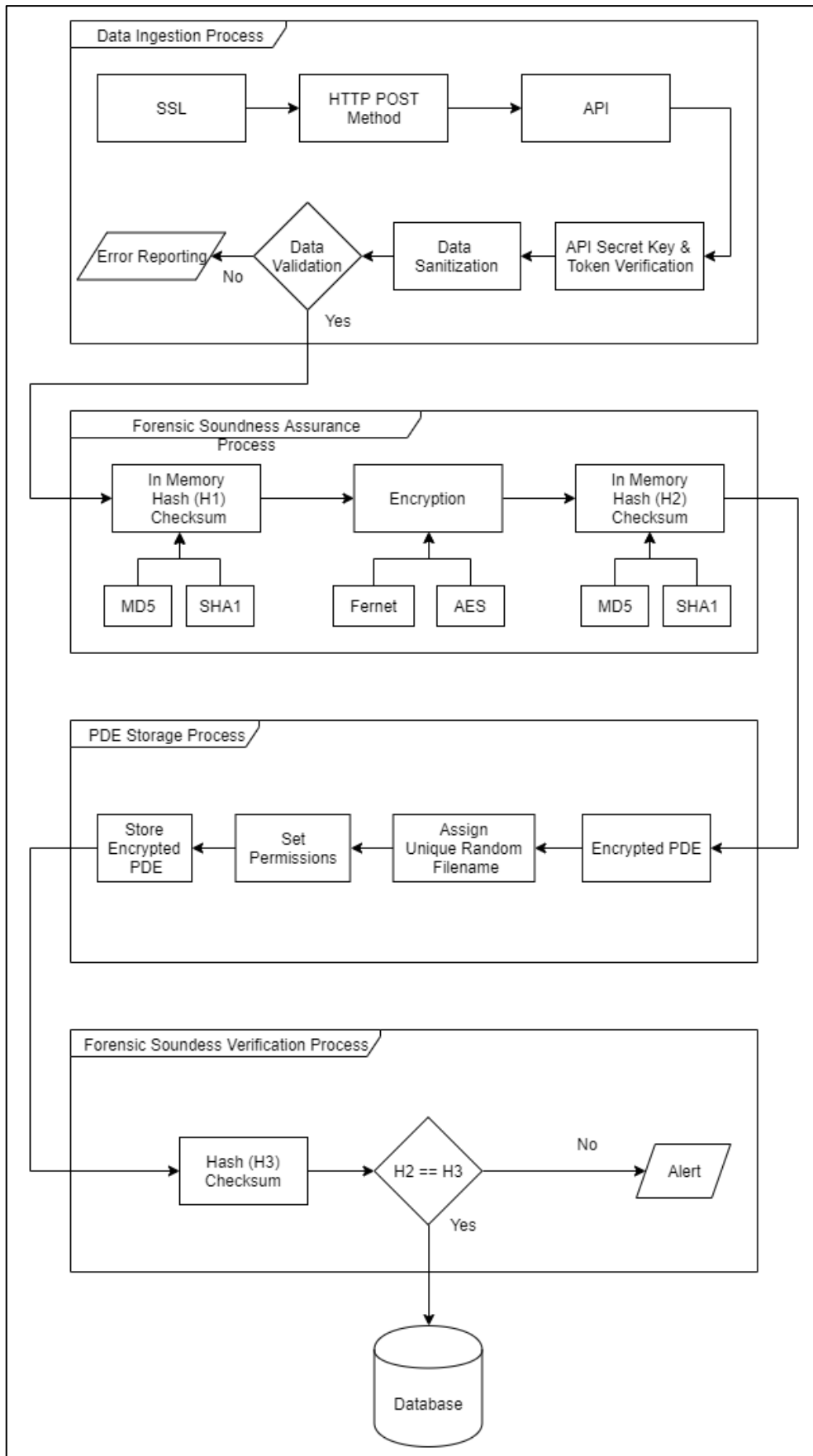


Figure 6-6. W3RS detailed process model

## 6.3 W3RS system requirements specification

Following the same process as in Section 5.3, this section defines the requirements for the storage tool W3RS. The aim is to provide a high quality and useful proof-of-concept prototype. Just like in Chapter 5, the requirements are drafted based on the proposed model – in this case, the process model in Figure 6-6. The system requirements are again partitioned into two categories, namely Secure Storage Core Requirements (SS-CR) and Secure Storage Optional Requirements (SS-OR).

### 6.3.1 Secure Storage Core Requirements (SS-CR)

- **SS-CR-01:** The tool shall ingest data from an API endpoint.
- **SS-CR-02:** The tool shall perform logging at every action/process that occurs.
- **SS-CR-03:** The tool must ingest data concurrently.
- **SS-CR-04:** The tool must show consistency in data storage.
- **SS-CR-05:** The tool must hash the ingested data.
- **SS-CR-06:** The tool must sanitise data ingested.
- **SS-CR-07:** The tool must perform hashing on the collected data.
- **SS-CR-08:** The tool must show the hash digest and metadata.
- **SS-CR-09:** The tool must provide digests of the encrypted PDE to ensure integrity.
- **SS-CR-10:** The tool must distinguish between different PDE.
- **SS-CR-11:** The tool must list the information collected.
- **SS-CR-12:** The tool must validate the data ingested.
- **SS-CR-13:** The tool must verify user authentication details.
- **SS-CR-14:** The tool must securely download PDE.

### 6.3.2 Secure Storage Optional Requirements (SS-OR)

- **SS-OR-01:** The tool must encrypt all metadata.
- **SS-OR-02:** The tool must decrypt PDE on access.
- **SS-OR-03:** The tool must list all the stored PDE.
- **SS-OR-04:** The tool must clearly show detected malicious PDE.
- **SS-OR-05:** The tool must perform 2FA authentication for PDE download.

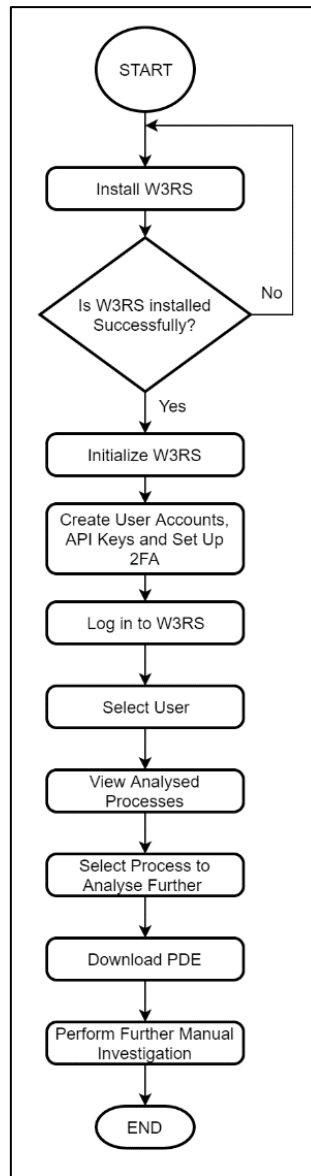
## 6.4 W3RS system implementation

Using the stated requirements, the system was implemented based on a modular approach. Python was used as the coding language because of its ease of use and ability to be easily packaged. W3RS uses the Django web framework to achieve this. (The code and installation process can be found in Appendix B.) The tool makes use of the MD5 hashing algorithm as the integrity checksum and of the Django REST framework for RESTful API functionality. The term Representational State Transfer (REST) is a software architectural style that provides constraints on how

data is transferred by using a stateless protocol for better performance, reliability and scalability. Unlike its predecessor, Simple Object Access Protocol (SOAP), REST also allows for easy updates, redeployments, management and the ability to structure data without the restrictions of providing a schema (description of the data). The Django REST framework furthermore provides authentication based on the API key. A secure API key is generated from the Admin panel, and this unique key is used when making an HTTP POST request to the API in order to verify the authenticity of the request. Django models make use of encrypted fields that encrypt all the metadata of the stored PDE data in the database. This ensures that if the database was breached through unauthorised access, the data inside the database is still encrypted.

The Django encrypted file field was chosen to secure the PDE using the Fernet encryption scheme. Fernet symmetric encryption is a symmetric key algorithm that makes sure that the encrypted message cannot be manipulated, brute-forced or read without the key. It also uses URL safe encoding for the keys by using the Advanced Encryption Standard (AES) 128-bit Cipher Block Code (CBC) mode. This means that any reserved, unprintable or non-ASCII characters are replaced, so that no errors occur when handling the keys that an attacker could potentially exploit. Fernet also makes use of Public Key Cryptographic Standards number 7 (PKCS7), which is used to encrypt messages under a Public Key Infrastructure (PKI) padded with Hash-based Message Authentication Code (HMAC). HMAC is used to simultaneously verify the integrity and authenticity of a message. To ensure better security, HMAC was used with Simple Hashing Algorithm (SHA) 256-bit hashing.

The W3RS application was also set up to be ready for Docker [\[163\]](#) (a containerised approach to hosting services), thus making it scalable as well as platform independent. Using Docker furthermore makes it easier to port between servers and even provides the ability to perform load balancing and multiple instances. A high-level flow chart is shown in Figure 6-7 to aid the explanation and show how the W3RS system lifecycle works.

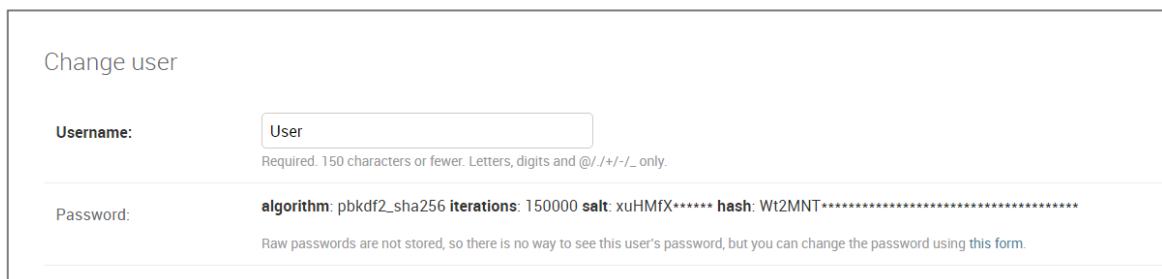


**Figure 6-7.** High-level lifecycle of W3RS

The lifecycle of W3RS starts off with ensuring that the system is installed successfully. The next step is to initialise W3RS. This is done in Django by creating a superuser that provides all admin functionality such as creating users and setting access roles. In this particular implementation, the admin user is the same as a Django superuser. Django models are fully customisable to assign permission to certain functionality, based on the role of a user.

After the W3RS system is initialised, the next step is to create user accounts, set up two-factor authentication (2FA) and create API keys. It was decided that only an admin user can create users and API keys in which every activity is logged. The tool makes use of two-factor authentication (2FA), which is simply adding another factor to traditional login systems in the form of a second key that is generated dynamically on the server and sent via email or SMS to the user [52]. 2FA was used to download a PDE, and currently only supports token generators and Yubi physical keys. These

two methods are further discussed in more detail, because they are the common ways of performing 2FA. Token generators make use of the Time-Based One-Time Pin (TOTP) algorithm [164] that generates 6-8 unique digits based on the current time and some secret key that is added when the device is registered. This token is changed after every 30 seconds to prevent attackers from brute-forcing the token. Google Authenticator [165] and Authy [144] are the most commonly used applications to store 2FA codes where a Quick Response (QR) code needs to be scanned and registered to the device on which the application is installed. A Yubi key [166] is a physical key that fits into a USB port of a computer to perform hardware authentication. The system also supports SMS as an additional authentication factor. User credentials are stored using Django's default password field that uses strong SHA 256-bit hashing [167]. To increase the difficulty in brute-forcing the hash, the password is typically hashed multiple times. The algorithm is performed the default number of 150000 iterations, as specified from Django documentation [167]. A password salt value (randomly generated characters) is appended to the password before it is hashed so as to further increase the difficulty for an attacker to brute-force the password. This process is designed to prevent any attempt by a system admin or hacker to gain unauthorised access to user passwords because they are never stored in plain text. The hash as well as the salt is redacted, as shown in Figure 6-8, further making it impossible for an admin to get access to user credentials.



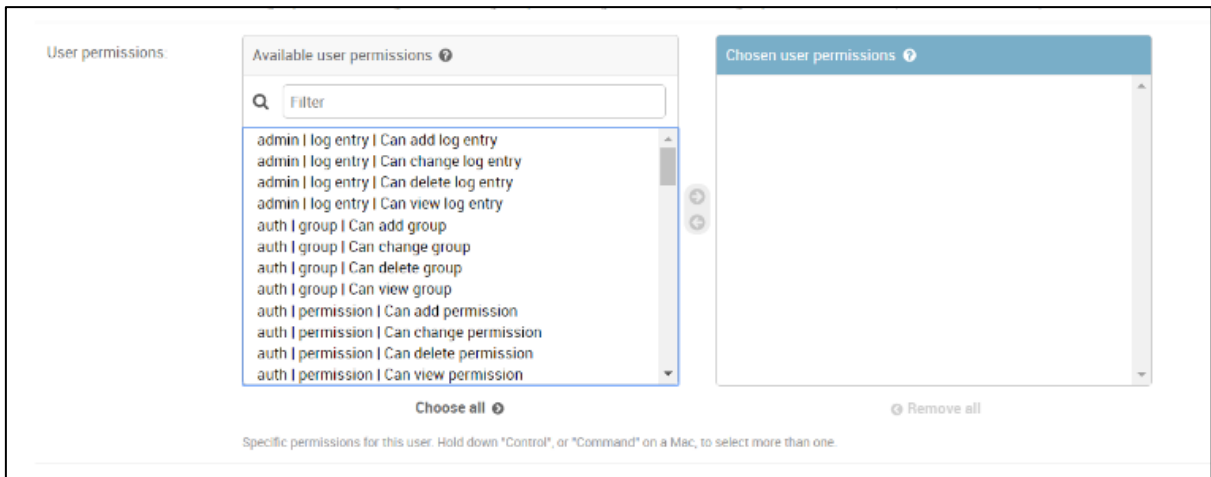
Change user

Username:   
Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

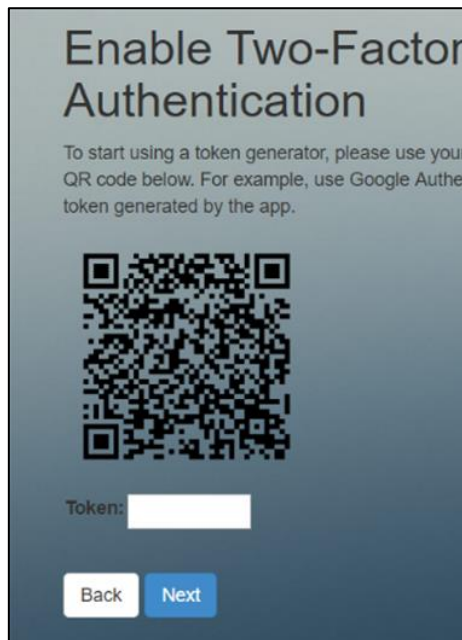
Password: **algorithm: pbkdf2\_sha256 iterations: 150000 salt: xuHMFx\*\*\*\*\* hash: Wt2MNT\*\*\*\*\***  
Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

**Figure 6-8.** W3RS user password standards

User permissions can also be set for added security and additional granularity. In Figure 6-9, a view is shown where an admin can set permissions for a specific user based on the Django models and functionality. Two-factor authentication can be set up by scanning a QR code through a token generator app like Authy, as shown in Figure 6-10.



**Figure 6-9.** W3RS user permissions assignment



**Figure 6-10.** W3RS adding 2FA

The next step is for an investigator to log in to the W3RS system where the investigator is presented with a list of users for which the W2RC collection tool has been installed and configured. Figure 6-11 contains an example of the user interface after a forensic investigator has logged in successfully. When an investigator selects a user to investigate or view, the details of each analysed process are displayed to the investigator. This is shown in Figure 6-12, where a redacted list of all the scanned processes that were collected is displayed – showing the relevant metadata such as the machine name, IP address, CAT value, MD5 checksum, username as well as the time. The IP address in the figure was redacted for anonymity purposes.

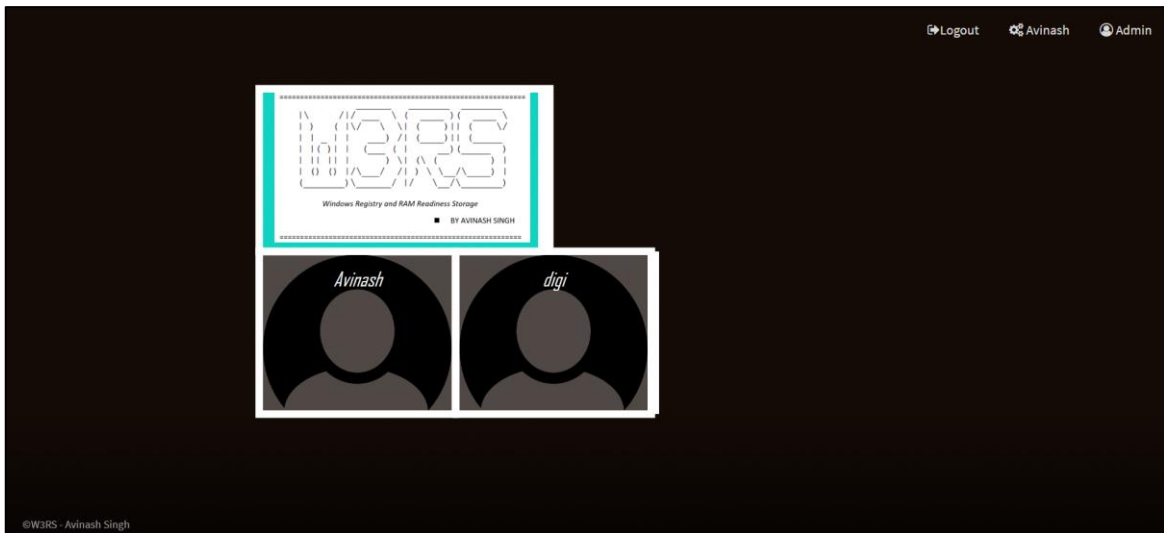


Figure 6-11. W3RS user interface

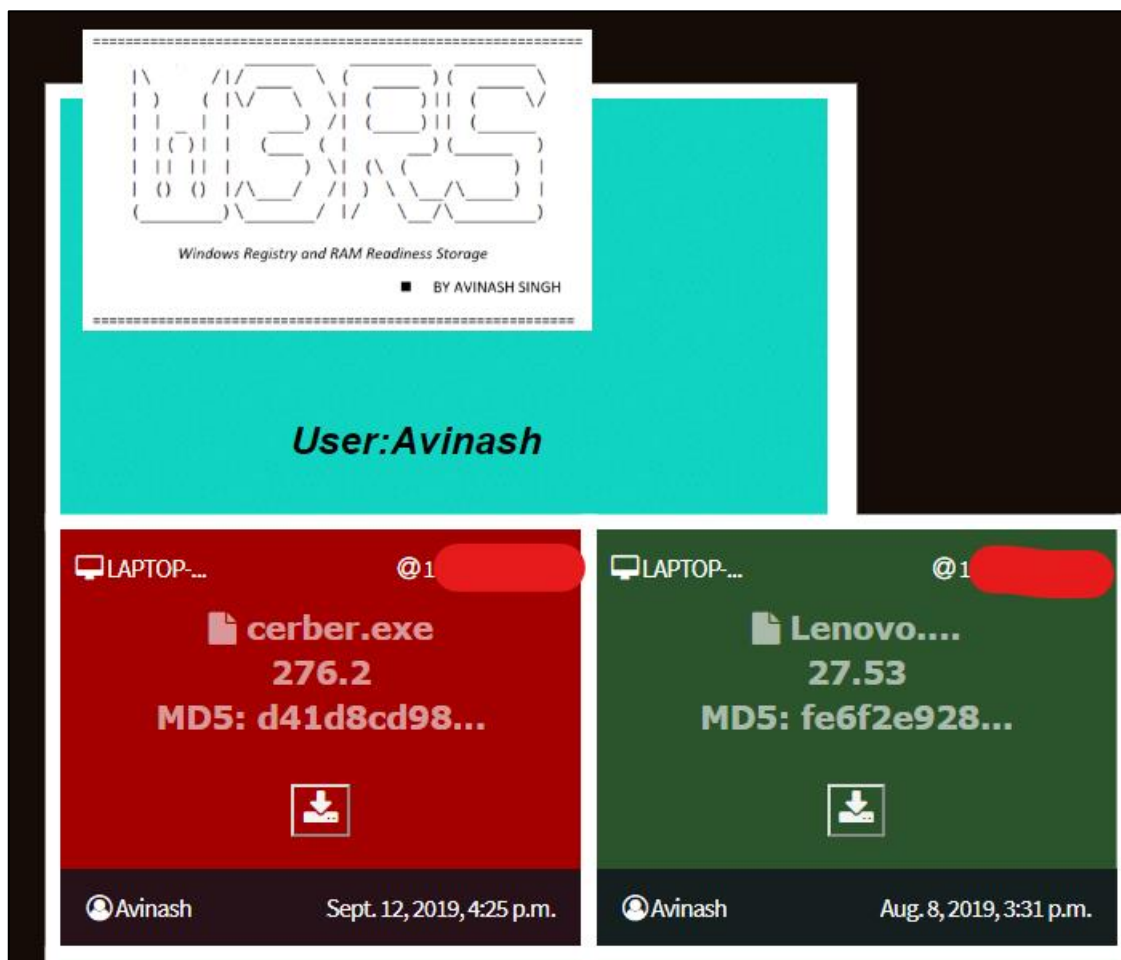


Figure 6-12. W3RS redacted view of stored PDE

When selecting the process that is to be further investigated, the investigator can click on the download icon to download the PDE. The user session will be validated by checking if the logged-in user has the required permissions and if the user has



an active session. This is done by checking the inactivity time and if 2FA is enabled. After the session has been validated successfully, the PDE can be downloaded and an investigator can investigate the collected PDE data to further corroborate the findings. A PDE file contains detailed information about what was collected, as well as the details of process memory, buffer information, registry, network, DLL imports, and other information. An example snippet of the process memory is shown in Figure 6-13. Another example snippet showing network activity can be seen in Figure 6-14. More details and snippets of the information collected appear in Appendix C.

```
"procmemory": [
  {
    "regions": [
      {
        "protect": "rw",
        "end": "0x00020000",
        "addr": "0x00010000",
        "state": 4096,
        "offset": 24,
        "type": 262144,
        "size": 65536
      },
      {
        "protect": "rw",
        "end": "0x00021000",
        "addr": "0x00020000",
        "state": 4096,
        "offset": 65584,
        "type": 131072,
        "size": 4096
      }
    ]
  }
],
```

Figure 6-13. PDE sample showing process memory

```
"network": {
  "tls": [],
  "udp": [
    {
      "src": "192.168.56.101",
      "dst": "192.168.56.255",
      "offset": 1068,
      "time": 3.083926200866699,
      "dport": 137,
      "sport": 137
    },
    {
      "src": "192.168.56.101",
      "dst": "192.168.56.255",
      "offset": 8340,
      "time": 9.07862901687622,
      "dport": 138,
      "sport": 138
    }
  ]
},
```

Figure 6-14. PDE sample showing network activity

## **6.5 Conclusion**

Chapter 6 presented the overall model for the secure digital forensic storage mechanism for the Windows Registry and RAM Readiness Storage (W3RS) component of the prototype. The interaction, techniques and logical assertions required for the effective development of the prototype were also presented in this chapter. Therefore, the developed prototype presents a tool that can be used to store potential digital artefacts that can be used for ransomware forensics.

Part IV serves as an evaluation of the proposed framework and developed proof-of-concept tools (i.e. RRF, W2RC and W3RS). The evaluation process involves real-world case studies, results and interpretation, and finally, critical evaluation.

# PART IV

## EVALUATION

## 7. CHAPTER 7: RESULTS AND INTERPRETATION OF THE PROTOTYPE SYSTEM

### 7.1 Introduction

In this chapter, the overall proposed ransomware readiness framework (RRF) was implemented using the developed proof-of-concept prototype tools. The proposed framework was evaluated through a testing process where samples identified were tested to determine reasonable threshold values to accurately detect malicious behaviour based on the metrics derived from Chapter 5. Recall that these metrics from Chapter 5, Equations (5.1) – (5.5), were derived from manual experimentation and observation to determine key characteristics and commonalities between ransomware samples. Therefore, the results presented in this chapter test and evaluate these metrics, based on five benign and four malicious samples. Since every process is monitored, a sample refers to a process executing in memory. The results of each Context-Based Analysis (CBA) is presented in the subsections below. Each CBA is presented with its respective result to further provide the necessary insight that was gained during the testing process.

### 7.2 Results obtained from the testing phase

The malicious samples considered in this study focused on popular variants of ransomware that hampered organisations, such as Cerber, WannaCry, CryptoLocker and TeslaCrypt [27] [88]. The benign samples comprised of TrueCrypt, VeraCrypt, 7zip, Microsoft Word and Adobe Reader. In order to correctly measure the behaviour as well as accurately perform testing, two samples, i.e. TrueCrypt and VeraCrypt, were chosen based on the popularity of the tools as well as their ease of use [168] [169]. The other three benign samples, 7zip, Microsoft Word and Adobe Reader, were chosen based on everyday usage statistics as well as the fact that these programs are often used as a vehicle to deploy malware in a system [74]. In other words, these programs are often infected with malware. Ransomware executables were obtained from theZoo [170], Hybrid-Analysis [103] and VirusTotal [102]. Experimental tests were conducted on a server set up with Windows 7 Virtual Machine (VM) sandbox using the Cuckoo framework.

Since W2RC works by sending every newly created process to a sandbox for evaluation, there was no need to draft a baseline of operations in the environment to get an unbiased result. This baseline refers to what Hampton et al. [100] proposed when they defined series of tasks to perform to test each sample and determine normal behaviour. Examples of baseline operations included opening Windows Explorer, installing MS Office 07, running MS Excel, opening Internet Explorer and so forth. Every process (whether benign or malicious) was evaluated based on its characteristics of execution. Each process then underwent a 15-second analysis using the Cuckoo framework to determine the CBA. The 15 seconds represent a soft timeout. In other words, if it would be detected that the executable is performing

unusual tasks within the first 15 seconds of its execution, analysis will continue for a little longer, with a max timeout of 60 seconds of total execution time so as to be more certain whether the process is actually suspicious or not. These execution times were chosen based on traditional anti-virus software sample scanning to prevent unnecessary waiting and high resources usage while analysis continues in the sandbox. The maximum timeout used was also the minimum recommended time by the Cuckoo framework for quick analysis [104]. The results from the testing process of each CBA metric, DLL monitoring, API monitoring, registry monitoring and entropy monitoring that was proposed in Chapter 5 are further explained in the subsections below.

### 7.2.1 DLL monitoring results

The results of the DLL Monitoring (DM) technique are shown in Table 7-1. Recall from Equation (5.1) the details on how the DM value is computed. From the summation of cryptographic libraries (cl), it is observed that benign samples without any encryption functionality do not contain any cryptographic libraries. The malicious samples, on the other hand, have an average number of cryptographic libraries ranging between 2 and 3. This is because most of the cryptographic functionality that is needed is found in built-in libraries used by these malicious samples. VeraCrypt had a cl value of 8, due to the various cryptographic functionalities that it provides, as opposed to TrueCrypt that only exhibits a cl value of 1. The number of cryptographic functions (cf) used makes a significant contribution to the final DM calculation. This is because these functions not only perform the encryption of the files, but also generate key-pairs for the cipher suite used. For example, Cerber and WannaCry have a significant number of cryptographic functions, as opposed to TeslaCrypt and CryptoLocker. After TeslaCrypt and CryptoLocker had been manually analysed, it was observed that TeslaCrypt and CryptoLocker performed several checks to see if they were executing in a virtual environment and whether they had a debugger attached or not.

**Table 7-1.** DLL monitoring results

<b>Samples</b>	<b>Benign / Malicious</b>	$\sum cl$	$\sum cf$	$DM = \frac{\sum cf}{\sum cl}$
TrueCrypt	Benign	1	1	1
VeraCrypt	Benign	8	78	9.75
7zip	Benign	0	0	0
Word	Benign	0	0	0
Adobe	Benign	1	0	0
WannaCry	Malicious	2	710	355
CryptoLocker	Malicious	3	22	7.33
Cerber	Malicious	3	214	71.33
TeslaCrypt	Malicious	1	3	3

This behaviour shows the evasive techniques that this ransomware employs, as well as their anti-forensic capabilities. For example, a malicious sample could detect if it was executing in a virtual environment (by checking the processor version, or VM libraries present) and then perform another task or even stop the execution, thus bypassing several sandboxed behavioural analysis processes. Since this is a limitation of all behavioural-based detection mechanisms, further research needs to be conducted. A threshold value of 49.712 was computed based on the average DM value for all samples to determine whether a sample is malicious or not. In other words, a good range for a DM value would be from 0 – 49.712 and a bad range anything greater 49.712. However, this would not accurately detect all malicious behaviour as TeslaCrypt and CryptoLocker would not be flagged as malicious. Therefore, it is not sufficient to only monitor DLLs, as malicious samples have become sophisticated enough to bypass certain levels of detection.

The next section comprises the second CBA, API monitoring, which involves monitoring API function calls from the executable to determine the behaviour of the function. API monitoring produces much of the dynamic behaviour of the executable because it involves the actual logic and flow of the executable in terms of what it is doing and how it is doing it.

## 7.2.2 API monitoring results

The results of the API Monitoring (AM) technique are shown in Table 7-2. Recall from Equation (5.2) the details on how the AM value is computed. The values required to compute the AM value are displayed in Table 7-2, with the last column reflecting how the AM value was calculated.

**Table 7-2.** API monitoring results

<b>Samples</b>	<b>Benign / Malicious</b>	$\Delta c$	<b><i>N</i></b>	<b><i>nf</i></b>	<b><math>AM = \frac{nf}{\Delta c \times N}</math></b>
TrueCrypt	Benign	15	42	9542	15.1460
VeraCrypt	Benign	5.953	39	903	3.8894
7zip	Benign	9.516	21	1685	8.4319
Word	Benign	9.3280	9	2106	25.0858
Adobe	Benign	5.141	13	781	11.6858
WannaCry	Malicious	2.25	58	63771	488.6667
CryptoLocker	Malicious	16.813	18	127602	421.6380
Cerber	Malicious	0.468	58	6525	240.3845
TeslaCrypt	Malicious	2.766	29	29163	363.5650

The results show that the total number of function calls (*nf*) plays a big role when calculating the AM value, due to the many function calls that happen in a short time. This makes it easier to detect automated background activity like ransomware, which is performing numerous I/O operations like reading and writing files as well as performing encryption functions. Cerber, on the other hand, has the second

highest  $n_f$  but, due to the short time changes, yields a bigger result when compared to WannaCry. Based on the change in the time between API calls ( $\Delta c$ ) given in seconds, it is observed that the malicious samples have a low time value (i.e.  $\Delta c$  value). This means that the malicious samples perform rapid function calls that are indicative of encryption and excessive CPU utilisation. For instance, Cerber ransomware has a  $\Delta c$  value of 0.468 in comparison to 7zip with a  $\Delta c$  value of 9.516, which shows that malicious samples perform numerous function calls in a short period of time. It was also observed that benign samples' AM value is generally low (between 3.8894 and 25.0858). Therefore, the average AM value for benign samples is 12.848. However, Word computed a higher AM value in comparison to other benign samples. This can be attributed to a large number of function calls from its total number of libraries (N), which can be due to background functionalities like macros and utilities loaded by Word, and the mere fact that Word is a large application. Consequently, the malicious samples differ from the benign samples, with the malicious samples' AM value averaging on 378.563. From the results it is clear that a good range for an AM value is between 0 and 175.388, and a bad range is greater than 175.388.

The next section presents the third CBA, Windows Registry monitoring, and it involves monitoring registry calls from the executable to determine what system configurations are being queried and analyse what information is being requested. This information is mostly used by malware to determine how to execute by adapting to the environment it is executing in.

### **7.2.3 Registry monitoring results**

The results of the Registry Monitoring (RM) technique are shown in Table 7-3. Recall from Equation (5.3) the details on how the RM value is computed. Table 7-3 is structured in such a way that all the variables needed to calculate the RM value are represented in the columns of the table. The last column displays how the final RM value is computed.

From these results, it seems that none of the samples inhibited any registry key deletion (kd). Deletion of keys shows that an application is attempting to remove traces and demonstrating destructive behaviour, which could make the system unstable. The reason why ransomware typically does not perform registry key deletion is that if the system becomes corrupt or unstable, the decryption application demanding the ransom payment may not work correctly. To better understand what is going on in Table 7-3, an example interpretation is now given. From the TrueCrypt sample, it is known that this sample is benign. The average number of registry events per second (reps) is calculated by counting the average number of registry operations that occur in one second.

**Table 7-3.** Windows Registry monitoring results

Samples	Benign / Malicious	<i>reps</i>	$\sum kd$	$\sum kr$	<i>kq</i>	$\sum kc$	$\sum ko$	$RM = \frac{reps \times \frac{\sum kd}{1 + \sum kr} + kq \times \frac{\sum kc}{1 + \sum ko}}$
TrueCrypt	Benign	1197	0	29	55	1851	7727	108.0735
VeraCrypt	Benign	139	0	1	71	130	189	189.0789
7zip	Benign	21	0	0	0	14	30	21.0000
Word	Benign	3	0	0	0	0	4	3.0000
Adobe	Benign	10	0	0	0	2	8	10.0000
WannaCry	Malicious	18	0	8	33	568	1464	47.7945
CryptoLocker	Malicious	13	0	10	0	34	80	1.1818
Cerber	Malicious	15	0	6	346	942	1794	529.7206
TeslaCrypt	Malicious	99	0	11	0	77	135	8.2500

Each event has a corresponding timestamp, and the *reps* for TrueCrypt is 1197. This means that an average of 1197 registry events occur in one second, which is a relatively high value in comparison to the other benign samples. After manually analysing the data, it was determined that TrueCrypt performs many registry key open operations that pertain to the configuration of the system. Most of these operations had to do with reading the policies of the environment. TrueCrypt has a sum-of-keys-created (*kc*) value of 1851. This means that many registry keys were created, and manual analysis found that most of the keys created involved the setting of uninstallation configurations within registry and registering the application in various parts of the system. For example, when an application is installed, many configurations are set in registry to ensure that the application works correctly. This behaviour was also seen with Cerber, which tried to make itself persistent in the system. The keys queried (*kq*) is a method that Windows Registry provides to read values from keys. These values are numeric or textual information about the key. For example, reading the computer name from the registry key HKEY\_CURRENT\_USER\\Software\\Microsoft\\Windows\\CurrentVersion. VeraCrypt displayed an above-average value of 71. This means that VeraCrypt queries a lot of registry information. Upon further manual investigation of VeraCrypt, it was found this queried information aimed to enhance the user experience, such as querying multiple languages and font types. However, with the other encryption tool, TrueCrypt, the corresponding value was lower, meaning that it did not require much registry information in comparison to VeraCrypt. It can be concluded that encryption tools need to read values from within the registry to operate correctly. An example of this is reading registry keys to get a list of user profiles, as well as getting a list of registered file types (such as txt, exe, pdf, etc.) within the system.



The benign encryption tools yielded a high RM value. This can be due to reading settings and metadata from the registry. The malicious samples had lower registry events per second (reps) but a high number of keys opened (ko), meaning that much information is being read in registry searching for items like system settings, installed applications and user profiles. A high number of keys created (kc) implies that the executable is creating system configurations and trying to instantiate itself within the system permanently. This relates to the ransom payment demands that are persistent, even after a system reboot. This is achieved by adding the ransom demand tool as a start-up application as well as modifying the desktop wallpaper of the system. The average RM value for all the benign samples is 66.23. Using this average, a good range would be from 0 to 66.23 and a bad range anything greater than 66.23. However, based on this range, the detection would be extremely bad, because neither WannaCry, CryptoLocker nor TeslaCrypt would be detected.

From the above, it can be concluded that RM alone is not sufficient to perform accurate detection, due to registry events not being fixed for each sample. It solely depends on the application itself, since Windows registry serves as a repository for information. It can also be concluded that typical ransomware does not need to read much of system configurations to perform their task. However, Cerber has a high RM value because it is a RaaS (Ransomware as a Service). RaaS is so designed that it can be controlled and managed remotely by attackers in a cloud environment. This means that more functionality is needed for attackers to control what the ransomware does, and that more system registry information is needed to successfully embed itself within the registry and the operating system as a whole.

The next section deals with the fourth and final CBA – entropy monitoring. It involves monitoring entropy values obtained from the executable to determine the amount of randomness in the executable and whether it is performing encryption.

#### **7.2.4 Entropy monitoring results**

In order to determine the average entropy, a small test was conducted by encrypting different file types and then performing Shannon's entropy [171] [172] on each file to determine the amount of randomness in each. Recall from Section 5.2.4 where Shannon's entropy was defined. The file categories chosen for this test was based on most commonly used file types such as documents, media and archives. File types selected for documents were PDF and TXT, whereas media files were IMG and MP4, and archives were ZIP files. To get a better estimate of entropy in certain file types, ten files were chosen for each file type. These files were chosen at random to avoid any file type from being biased. A comparison of authentic vs non-authentic encryption entropy analysis is presented in Table 7-4.

**Table 7-4.** Authentic vs non-authentic encryption using entropy analysis

<b>File type</b>	<b>Sum file size</b>	<b>Average entropy without encryption</b>	<b>Average authentically encrypted entropy</b>	<b>Average ransomware (WannaCry) encrypted entropy</b>
PDF	9.85 MB	7.9434395	7.9992378	7.9992195
TXT	39.80 KB	4.7150093	7.4754678	7.7509205
IMG	4.42 MB	7.8063508	7.9984814	7.998468
MP4	162 MB	7.9726569	7.999988	7.999984
ZIP	10.00 MB	7.9564831	7.9917306	7.989744

Each file was encrypted using standard OpenSSL AES-256-CBC, and the ransomware encryption process was carried out using WannaCry. The results showed that the usage of encryption changes the entropy of the file.

Typically, the American Standard Code for Information Interchange (ASCII) values of a text file is stored in Unicode Transformation Format (UTF) when encrypted. The maximum number of bits required to represent a byte is 8 bits. However, the average entropy represents the average number of bits needed to represent one byte. Therefore, from standard ASCII text data, the average number of bits needed is 4.715 to represent a byte; however, when this text gets encrypted, the average number of bits needed changes to 7.475. Thus, a text file would have a high entropy change, as observed in Table 7-4. However, a small change in entropy relative to the average entropy without encryption was observed from the average authentically encrypted entropy of the encrypted files. When inspecting the headers of a normally encrypted file vs a ransomware encrypted file it is clearly distinguishable, since a ransomware file header is abnormally structured to prevent the user from recovering the encrypted files. This structure is defined by the authors of the ransomware itself and is usually kept a secret to prevent security researchers from finding a way to decrypt the files.

The results of the proposed Entropy Monitoring (EM) techniques appear in Table 7-5. From the entropy of the executable file (Ex) with values generated using Shannon's entropy [171], it was observed that the executables of the malicious samples have high entropy (as seen from WannaCry and CryptoLocker). On the other hand, some samples that perform encryption like TrueCrypt and VeraCrypt are in the high 6's, whereas Adobe and 7zip have an entropy value in the low 6's. Adobe and 7zip also perform encryption as one can password-protect an archive as well as a PDF file. A non-encryption-based sample – Word – has a very low entropy value. This can be attributed to Word not needing many bits to represent a byte of information, as Word documents typically contain ASCII textual data that does not need many bits to store in comparison to UTF.

**Table 7-5.** Entropy monitoring results

<b>Samples</b>	<b>Benign / Malicious</b>	<i>Ex</i>	$\sum_{i=0}^{Npes} PES$	<i>Npes</i>	$\frac{\sum_{i=0}^{Npes} PES}{Npes}$	$EM = Ex + \frac{\sum_{i=0}^{Npes} PES}{Npes}$
TrueCrypt	Benign	6.8838	25.7149	4	6.4287	13.3125
VeraCrypt	Benign	6.6805	26.8070	5	5.3614	12.0419
7zip	Benign	6.1280	26.3258	6	4.3876	10.5156
Word	Benign	3.7468	17.4730	7	2.4961	6.2429
Adobe	Benign	6.0943	26.3965	5	5.2793	11.3736
WannaCry	Malicious	7.9954	25.5234	4	6.3819	14.3773
CryptoLocker	Malicious	7.1322	11.3990	3	3.7997	10.9319
Cerber	Malicious	5.0923	18.9669	4	4.7417	9.834
TeslaCrypt	Malicious	7.1163	19.8661	4	4.9665	12.0828

The average entropy of the Portable Executable Section (PES) describes whether the executable is trying to run in stealth and avoid anti-virus programs from analysing the executable. This can be seen in WannaCry having a relatively large (25.52) PES value relative to Cerber and TeslaCrypt. The EM value therefore provides the ability to see if an executable contains encrypted parts and avoids its true execution when performing static analysis. This can be seen with CryptoLocker where most of the data in the executable is encrypted and the PE sections are generally not encrypted (as opposed to WannaCry). The number of PE sections (*Npes*) shows that malicious samples generally have fewer PE sections between 3 and 4 so as to minimise the number of sections to load into memory and prevent early detection.

The average entropy of the PE sections as represented in the second-last column shows that the entropy of malware is not that much different to the entropy of benign samples. This is largely due to some PE sections not being encrypted in malware to enable anti-virus to just scan those sections and not flag it as harmful. However, when adding the entropy of the executable to the sum of the PE sections entropy, it is observed that most encryption-like tools have an EM value greater than 10. This means that there is more randomness of the data, which is indicative that the data is encrypted. The average entropy for benign samples is 10.697; however, this includes the two benign encryption tools. With this average Cerber would not be classified as malicious and Adobe would be classified as malicious. Therefore, by removing the two encryption tools when calculating the average, a value of 9.377 is obtained; however, this induced a higher number of false positives, as 7zip and Adobe would be classified as malicious. Therefore, a good range would be between 0 and 10.697, and a bad range would be greater than 10.697.

Now that all the CBAs have been discussed, the next section evaluates how the Context-Aware Trigger value is calculated to determine if an executable may be

harmful. A discussion follows on how the CBAs provide insight into an executable and on how stealthy malware has become.

## 7.2.5 Context-Aware Trigger results

The results of all the Context-Based Analyses (CBAs) are shown in Table 7-6.

**Table 7-6.** Experimental results from well-known applications and ransomware

Samples	Context-aware analysis techniques						Result
	Benign / Malicious	DM	AM	RM	EM	CAT	
TrueCrypt	Benign	1	15.1460	108.0735	13.313	34.383	Normal
VeraCrypt	Benign	9.75	3.8894	189.0789	12.042	53.6901	Normal, High
7zip	Benign	0	8.4319	21.0000	10.516	9.98688	Normal
Word	Benign	0	25.0858	3.0000	6.2429	8.58218	Normal
Adobe	Benign	0	11.6858	10.0000	11.374	8.26485	Normal
WannaCry	Malicious	355	488.667	47.7945	14.377	226.460	Abnormal, High
CryptoLocker	Malicious	7.33	421.638	1.1818	10.932	110.270	Abnormal
Cerber	Malicious	71.33	240.385	529.721	9.834	212.817	Abnormal, High
TeslaCrypt	Malicious	3	363.565	8.2500	12.081	96.7245	Abnormal

Since DM and RM do not accurately distinguish between benign and malicious samples, the combination of other forms of analysis in conjunction with DM and RM provides a better result. Therefore, an average value is computed from these CBAs to determine the Context-Aware Trigger (CAT) value. This value produces a quantifiable result to identify samples as being benign or malicious. Furthermore, more information and insight can be obtained from each CBA. For example, the DM and EM values can give a good estimation of whether the process (sample) is performing some encryption. A process that performs encryption on a large scale would likely generate higher values, as the entropy change would be significantly greater.

Furthermore, the number of cryptographic libraries loaded helps an investigator to decide whether the executable is performing any form of encryption and therefore requires further manual analysis. The DM and EM analyses mostly focus on static analysis as this information can be extracted from the executable prior to its execution. This is done so that a quick understanding of the executable can be gained. Complementarily, the AM and RM focus mainly on behavioural analysis, as it is dynamic and involves the process execution in a sandbox virtual machine. Analysing its behaviour provides additional information about the processes in near real-time. Based on such information, the combination of the static and behavioural (dynamic) analysis can then be used to formulate a threshold for determining normal

or abnormal behaviour. From the phases of the different CBAs, it can be determined that TrueCrypt and VeraCrypt are encryption tools based on their DM and EM values, while the DM and RM values indicate that 7zip and Word samples are not encryption tools and have an average RM compared to the others. WannaCry, Cerber, TrueCrypt and VeraCrypt have high RM values. This is because they query, create and open keys rapidly to make samples persistent in the system. Furthermore, WannaCry and Cerber have a high AM value, which implies that several function calls are initiated within a short duration of time.

Based on the experimental result as presented in Table 7-7, an average CAT value of all samples equated to 84.575. It was observed that abnormal behaviour has a CAT value greater than this average of 84.575. The average CAT value of the benign samples was 22.98 and forms a baseline for defining a threshold. The current study therefore asserts that a given sample is defined as abnormal if the CAT value is greater than 84.575, while samples with CAT value lower than 22.98 are considered normal. A CAT value between 22.98 and 84.575 is considered “Normal, High”. The average CAT value for malicious samples is 161.568. Therefore, a value greater than 161.568 is considered “Abnormal, High”. A summary of the threshold values for each CBA is presented in

Table 7-7. Based on these thresholds, the proposed CAT technique achieved a significant accuracy of 100%, since all samples that were malicious were detected as abnormal and all benign samples were classified as normal. Therefore, the proposed CAT technique can be used to evaluate ransomware. If the CAT value of a given process is higher than the pre-defined threshold, the process can be considered malicious. This observation is used as the requisite trigger mechanism for incident response and forensic investigation as it alerts a system admin and user of an incident. It also forms part of the pre-analysis phase for digital forensic readiness that can be integrated as a pro-active security mechanism against a ransomware (or other forms of malware) attack.

**Table 7-7.** A Summary of thresholding values

	<b>DM</b>	<b>AM</b>	<b>RM</b>	<b>EM</b>	<b>CAT</b>
<b>Average of all Samples</b>	49.712	175.3881	102.011	11.19028	84.575
<b>Average Benign Samples</b>	2.15	12.84778	66.2305	10.6973	22.981
<b>Average Malicious Samples</b>	109.165	378.5636	146.737	11.8065	161.57
<b>Abnormal Threshold (&gt;)</b>	49.712	175.3881	66.2305	10.6973	84.575

### 7.3 Conclusion

After rigorous testing of the overall framework and prototype tools, significant prowess was shown in collecting PDE as well as performing ransomware incident detection. Hence, if this framework can be adopted and implemented within an organisation, it will better prepare the organisation to be ready for and potentially prevent ransomware attacks. The framework also lays the groundwork for other malware types, as it can be applied for worm malware detection and incident response using digital forensics.

The study found that ransomware is becoming more advanced and that attackers use evasive techniques to detect virtual environments as well as debuggers to hide the malware's intent. Fortunately, these evasive techniques limit the widespread use of the ransomware. For example, if a malware detects that it is in a virtual environment, it might not execute. However, many enterprise architectures rely on virtualisation for rendering their services, which could be a major setback for malware authors.

This chapter presented the analysis and detection part of the framework as a whole and provided insights based on the CBAs performed. It offered the digital forensic investigator with a guideline on what further manual investigation may need to be conducted. For example, when the DM value is higher than the threshold, encryption functions are called that are indicative of encryption and that prompt the investigator to see what function calls were made. Further investigation can be carried out on the collected PDE by analysing it on a lower level to determine the intent of the malware as well as whether it was flagged as a false positive. This result can be used in future to tweak the thresholds to minimise the number of false positives that might occur.

The next chapter in this Part (Chapter 8) contains real-world case studies and shows how the proposed framework and its supporting tools could have been used by organisations to avert crisis.

## **8. CHAPTER 8: REAL-WORLD CASE STUDIES**

### **8.1 Introduction**

This chapter focuses on real-world case studies by referring to known ransomware attacks on organisations and companies. It provides insight into real-world cases and further shows the usefulness of the proposed ransomware readiness framework (RRF). The chapter is divided into three chosen case studies with a brief discussion of how ransomware attacks could have been detected and potentially prevented with the adoption of the proposed RRF. In each case study, the method of attack is replicated as far as possible and the two prototype tools are used to gauge how the situations would have been averted. This is predicated on the assumption that the framework – if implemented within an organisation – could have detected and potentially prevented attacks.

### **8.2 Real-world case studies**

Three case studies were taken into consideration in this section. The case studies were chosen based on the large amount of damage they caused and the ransomware that was used in the attack. The damage was measured in terms of the amount of downtime, and the amount of cost required to recover the systems after the attack. Cases were chosen to demonstrate different ransomware (i.e. not just cases involving WannaCry, but also other variants of ransomware) and to show the robustness of the proposed framework and developed process model, as well as the capabilities of the prototype. All scenarios were tested in a Windows 7 Service Pack 1 (SP1) computing environment. This was chosen because most of these organisations used Windows 7 and Windows 7 has a large user base [13]. There are also several exploits available for Windows 7 since it has been in use for many years [97] [173]. Organisations often have trouble updating their systems as some legacy systems are not compatible with later versions of Windows. The analysis configuration server used Windows 7 SP1 and each malware sample was executed and analysed for 15 seconds with a soft timeout, as explained before. Sample PDE snippets in Appendix C show the signatures that are matched and some of the information that is calculated and stored to aid an investigator when conducting an investigation. These signatures are based on the samples' behaviour and were taken from Cuckoo's community [174] of signatures.

Each of the three case studies is structured under three headings, namely scenario definition, ransomware detection and results, and discussion.

#### **8.2.1 Case study 1: WannaCry**

This specific case study explored WannaCry ransomware.

### 8.2.1.1 WannaCry scenario definition

Sourced from the National Audit Office [175]:

*“On Friday 12 May 2017 a global ransomware attack, known as WannaCry, affected more than 200,000 computers in at least 100 countries. In the UK, the attack particularly affected the National Health Service (NHS), although it was not the specific target. At 4 pm on 12 May, NHS England declared cyber-attack as a major incident and implemented its emergency arrangements to maintain health and patient care. On the evening of 12 May, a cyber-security researcher activated a kill-switch so that WannaCry stopped locking devices. According to NHS England, the WannaCry ransomware affected at least 80 out of the 236 trusts across England, because they were either infected by the ransomware or turned off their devices or systems as a precaution. A further 603 primary care and other NHS organisations were also infected, including 595 General Practitioners (GP) practices. Prior to the WannaCry attack, the Department of Health (the Department) and its arms-length bodies, had work underway to strengthen cyber-security in the NHS. For example, NHS Digital was broadcasting alerts about cyber threats, providing a hotline for dealing with incidents, sharing best practice and carrying out on-site assessments to help protect against future cyber-attacks. The NHS England had also embedded the 10 Data Security Standards (recommended by the National Data Guardian) in the standard NHS contract for 2017-18 and was providing training to its Board and local teams to raise awareness of cyber threats. In light of the WannaCry attack, the Department announced further plans to strengthen NHS organisations’ cyber-security.”*

### 8.2.1.2 WannaCry detection and results

The WannaCry ransomware sample was retrieved from Hybrid-Analysis [103], and information about this sample appears in Table 8-1. When the ransomware was run on the system, it was immediately detected and suspended by the W2RC prototype tool and sent to the analysis server for investigation. After the sample was analysed, it was determined that it was indeed malicious by providing a Context-Aware Trigger (CAT) value of 1439.21, which is extremely high.

**Table 8-1.** WannaCry sample information

Sample Information	
<b>SHA256</b>	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
<b>Type</b>	PE32 executable (GUI) Intel 80386, for MS Windows
<b>Size</b>	3.35 MB
<b>Source</b>	theZoo [170]

The results of calculating the CAT value are shown in Table 8-2 and discussed in the section that follows.



**Table 8-2.** WannaCry case study

Trigger	Computation	Result
<b>DLL Monitoring (DM)</b>	$\frac{\sum cf}{\sum cl} = \frac{710}{2}$	355
<b>API Monitoring (AM)</b>	$\frac{nf}{\Delta c \times N} = \frac{63771}{2.25 * 58}$	488.6667
<b>Registry Monitoring (RM)</b>	$\begin{aligned} &reps \times \frac{\sum kd}{1 + \sum kr} + kq \times \frac{\sum kc}{1 + \sum ko} \\ &= 18 * \frac{0}{8} + 33 * \frac{568}{1464} \end{aligned}$	47.7945
<b>Entropy Monitoring (EM)</b>	$Ex + \frac{\sum_{i=0}^{Npes} PES}{Npes} = 7.9954 + \frac{25.5234}{4}$	14.3773
<b>Context-Aware Trigger (CAT)</b>	$\frac{DM + AM + RM + EM}{N}$	<b>226.4597</b>

### 8.2.1.3 WannaCry discussion

If it had adopted and implemented a readiness framework, the NHS would have been better prepared for this WannaCry ransomware attack. The NHS had no way of containing the ransomware attack, and if not for a cybersecurity researcher who initiated the global kill switch, the WannaCry ransomware attack would have been significantly worse. Ransomware attacked the operation of the NHS as a health care organisation, as research has shown that health care and schools are the first attack points [62] [176] [177]. The health care sector deals with sensitive and vital data that should always be available, thus making them a prime target. Life support and other medical equipment need to be online in order for medical services to be rendered. The extent of the WannaCry attack even crippled general practitioner (GP) practices, which was a major problem because a massive 595 practices were rendered unavailable. Investigating such a huge attack vector would be an extremely lengthy process and can lead an investigator into dead ends with information. Having a digital forensics readiness process in place would have allowed for faster case processing and the potential detection of a ransomware attack before major business disruption occurred.

Implementation of the RRF would have rendered the NHS better prepared for this attack. Firstly, the results in Table 8-2 show that the DLL Monitoring (DM) value of WannaCry is 355 which is fairly large and above the defined threshold of 28.27, which shows that there is definitely some encryption happening. The API Monitoring

(AM) value achieved a score greater than 488, which suggests many function calls, which are indicative of I/O operations – the usual behaviour of large-scale encryption. The second highest value was for Registry Monitoring (RM) with a computed value greater than 47. From looking at the calculations for RM, it was evident that the number of keys opened (ko) was relatively high in comparison to the keys read (kr). This is due to the keys-opened function call that maintains a handle to the registry key itself for faster and multiple operations that can be performed on the opened registry key. It can be concluded that a vast number of keys were opened, and some operations were performed on it, which is indicative of malicious behaviour (cf. Chapter 7). Further manual investigation of the collected PDE revealed that WannaCry contained shellcode, console writing, system config checks, memory check, and packers (i.e. hiding an executable within another executable). It even started up a server on several ports, which is not normal for a benign process. The malicious executable would immediately have been killed and blacklisted by W2RC, preventing the system from being compromised and infecting other users over the network.

## **8.2.2 Case study 2: Dharma**

This specific case study explored Dharma ransomware.

### **8.2.2.1 Dharma scenario definition**

Sourced from ZDNet [[178](#)]:

*“The Altus Baytown Hospital (ABH) has revealed a ransomware outbreak, which may have led to the leak of patient data. In a statement on its website, the Texas-based hospital said that ABH discovered an unauthorised threat actor rifling through the organisation's systems on roughly September 3rd, 2018. The "unauthorised party" deployed malicious code and infected the hospital's systems with a strain of ransomware. The ransomware at fault for the infection is known as Dharma. As with most strains, the malware was able to encrypt files and then demanded a ransom payment in return for access. Many of the hospital's records were encrypted due to the attack, and these included files containing patient information such as names, home addresses, dates of birth, social security numbers, driver license numbers, credit card information, phone numbers, and medical data. It would be unusual for ransomware to encrypt and then exfiltrate information should the malware's purpose be simply to secure a blackmail payment. However, as the threat actor was present on ABH servers and details are thin on the ground, it is possible this data has made its way into the wrong hands. ABH has not revealed how many patients may be affected. In addition to the hospital itself, affiliate parties including Altus Women's Center of Baytown, Oprex Surgery (Baytown), Clarus Imaging (Baytown), LP, Clarus Imaging (Beaumont), Zerenity Baytown, and Altus Radiation Oncology Baytown are involved in the incident as information from these entities was stored on the same systems. After the ransomware executed, the hospital chose not to pay the ransom; instead, ABH hauled in external cybersecurity help, which was able to*

decrypt backup files and restore ABH's servers. Dharma was then eradicated from the compromised systems.”

### 8.2.2.2 Dharma detection and results

The RobbinHood ransomware sample was retrieved from VirusTotal [102], and information about this sample is provided in Table 8-3. The ransomware was subsequently run on the system, suspended by W2RC and sent to the analysis server for investigation. On analysis, the sample was found to be somewhat malicious as it provided a CAT score of 85.7319, which is not significantly higher than the threshold of 84.575. However, it was still detected as abnormal, because Dharma does not rely much on using API function calls to perform the encryption. The detailed results of calculating the CAT score are shown in Table 8-4.

**Table 8-3.** Dharma sample information

Sample Information	
<b>SHA256</b>	315fbec706c3445ab51140be348c51761a3556f5c473b92f03c135fa82e070a
<b>Type</b>	PE32 executable (GUI) Intel 80386, for MS Windows
<b>Size</b>	0.09 MB
<b>Source</b>	VirusTotal [102]

**Table 8-4.** Dharma case study

Trigger	Computation	Result
<b>DM</b>	$\frac{\sum cf}{\sum cl} = \frac{271}{2}$	135.5
<b>AM</b>	$\frac{nf}{\Delta c \times N} = \frac{188516}{61.688 * 27}$	113.1837
<b>RM</b>	$reps \times \frac{\sum kd}{1 + \sum kr} + kq \times \frac{\sum kc}{1 + \sum ko} = 87 * \frac{1}{1} + 0 * \frac{4}{8}$	87
<b>EM</b>	$Ex + \frac{\sum_{i=0}^{Npes} PES}{Npes} = \frac{21.7317}{3}$	7.2439
<b>CAT</b>	$\frac{DM + AM + RM + EM}{N}$	<b>85.7319</b>

### 8.2.2.3 Dharma discussion

In this scenario, the ransomware attack was somewhat contained, but the confidentiality of the encrypted data was not known. If the ransomware stole the information from the hospital's databases and sold this data to third parties, the matter would have been significantly worse as it targeted the confidentiality and availability of the hospital's data. A ransomware sample was executed on the host machine, and the API monitoring was relatively low for a ransomware attack,

probably due to the evasive techniques that the ransomware employed. On deeper introspection, the number of function calls was found to be excessive; however, the time difference between each function call was larger, thus making the AM metric of evaluation less effective. The ransomware used a delay in API function calls to avoid detection. A portion of the time delay could be attributed to the process of encrypting ransomware files, making it inefficient but more subtle to avoid detection. Dharma ransomware did not have a high RM value because this ransomware does not modify and query the Windows registry as much. The number of registry events made a major contribution to the RM value. Therefore, many registry operations were performed, but none had a major effect on the integrity of the registry repository.

### **8.2.3 Case study 3: RobbinHood**

This specific case study explored the RobbinHood ransomware.

#### **8.2.3.1 RobbinHood scenario definition**

Sourced from Vox Recode [\[179\]](#):

*“Hackers targeted the city of Baltimore on May 7, 2019, using ransomware called RobbinHood, which, as NPR explains, makes it impossible to access a server without a digital key that only the hackers have. The Baltimore hackers’ ransom notes, obtained by the Baltimore Sun, demanded payment of three bitcoins per system be unlocked, which amounts to 13 bitcoins to unlock all the seized systems. The note threatened to increase the ransom if it wasn’t paid in four days and said the information would be lost forever if it wasn’t paid in 10 days. The city government refused to pay, meaning that the government email systems and payment platforms that were attacked remained offline. The attack has also harmed Baltimore’s property market because officials weren’t able to access systems needed to complete real estate sales. Baltimore Mayor Jack Young, who’s officially been in his office less than a month, said in a statement that city officials are “well into the restorative process” and have “engaged leading industry cybersecurity experts who are on-site 24-7 working with us.” The FBI is also involved in the investigation. “Some of the restoration efforts also require that we rebuild certain systems to make sure that when we restore business functions, we are doing so in a secure manner,” Young said. He did not offer a timeline for when all systems will come back online. The Baltimore City Council president also plans to form a special committee to investigate this latest attack and try to ensure it doesn’t happen again. A similar attack using RobbinHood hit government computers in Greenville, North Carolina, in April. A spokesperson for Greenville told the Wall Street Journal that the city never wound up paying and that while its systems aren’t entirely restored, “all of our major technology needs are now being met.” More than 20 municipalities in the US have been hit by cyberattacks this year alone. And such attacks can be expensive, perhaps especially if targets say they won’t pay. In 2018, hackers demanded that Atlanta pay about \$50,000 in bitcoins as part of a ransomware attack. The city*

refused, and according to a report obtained by the Atlanta Journal-Constitution and Channel 2 Action News, the attack wound up costing the city \$17 million to fix.”

### 8.2.3.2 RobbinHood detection and results

The RobbinHood ransomware sample was retrieved from Hybrid-Analysis [103], and information about this sample is presented in Table 8-5. The ransomware was run on the system, suspended by W2RC and sent to the analysis server for investigation. After the sample was analysed, it was determined that it was indeed malicious, providing a CAT score of 96.93973, which is above the abnormal threshold. However, it is not higher than other strains of ransomware like Cerber and WannaCry. The results of calculating the CAT score are shown in Table 8-6.

Table 8-5. RobbinHood sample information

Sample information	
SHA256	3bc78141ff3f742c5e942993adfbef39c2127f9682a303b5e786ed7f9a8d184b
Type	PE32 executable (console) Intel 80386 (stripped to external PDB), for MS Windows
Size	2.72 MB
Source	Hybrid-Analysis [103]

Table 8-6. RobbinHood case study

Trigger	Computation	Result
DM	$\frac{\sum cf}{\sum cl} = \frac{38}{1}$	38
AM	$\frac{nf}{\Delta c \times N} = \frac{581}{0.266 * 12}$	182.0175
RM	$reps \times \frac{\sum kd}{1 + \sum kr} + kq \times \frac{\sum kc}{1 + \sum ko} = 160 * \frac{1}{1} + 1 * \frac{16}{34}$	161.4571
EM	$Ex + \frac{\sum_{i=0}^{Npes} PES}{Npes} = \frac{87.97974}{14}$	6.2843
CAT	$\frac{DM + AM + RM + EM}{N}$	96.93973

### 8.2.3.3 RobbinHood discussion

RobbinHood ransomware seems to be fairly new as it was first encountered in early May 2019. The above scenario suggests that the majority of the city’s infrastructure was brought to its knees – a situation that could have been averted. Security firms and researchers propose that victims of ransomware attacks should not pay the ransom and so mitigate the ransomware attacks and incidents. However, reports on the above scenario indicate that the cost of recovering from a ransomware attack is

significantly higher than merely paying the ransom. This is because loading backups is a timeous process. Care must be taken to ensure that all systems are properly operating, and the ransomware has to be investigated and removed from the network. By adopting and implementing a ransomware readiness framework, the cost of investigation is significantly reduced. The framework could also prevent ransomware attacks, which further increases system hardening and security.

When RobbinHood was executed on the host machine with the implemented framework, the ransomware was detected as malicious because it scored a CAT value of more than 70. The contributing factors to this ransomware were the AM and RM. The API monitoring revealed a relatively low number of function calls, while the RM showed a fairly large number of registry events. This was due to the system configurations that RobbinHood queried from registry to determine the control set architecture. This particular ransomware (RobbinHood) performed rigorous querying to check if it was executing in a VM, thus implementing anti-forensics to avoid reverse engineering. Another trademark of all ransomware executables is that they always attempt to delete backups on the system through a command (cmd) prompt. The RobbinHood ransomware also attempted to encrypt other drives available on the computer.

### **8.3 Conclusion**

The capability of the proposed approach for ransomware forensics was demonstrated in this chapter. Using three real-life case studies, the chapter revealed the potential of the developed prototype to detect and prevent a ransomware attack, as well as to limit the spread of a ransomware attack. The study found that the proposed framework could be leveraged to address ransomware investigation challenges (by providing a data repository for an investigator) and that it actively produced insights into the ransomware through the defined CBAs.

The next chapter describes the evaluation process that was adopted to evaluate the ransomware readiness framework, as well as the developed prototype system.

## 9. CHAPTER 9: CRITICAL EVALUATION

### 9.1 Introduction

Chapter 9 discusses how the framework and prototype system were evaluated. For this, several approaches were identified to be used for proof-of-concept tool evaluation. These approaches generally fall in the category of formal laboratory-controlled experimentation, focus groups evaluation, field study for experimental data acquisition, benchmarking to standardised processes, and expert reviews [180] [181]. Chapter 7 reported on the controlled experimental process to evaluate the probability of identifying useful information for ransomware detection and investigation. Furthermore, the results obtained from the experimental process support the fundamental principle of digital forensic readiness. In this chapter, three approaches to evaluation are considered: software verification and validation; expert review and benchmarking. Each approach is discussed in detail in the sections that follow.

### 9.2 Software verification and validation process

In this section, the software verification and validation processes were conducted based on the NIST Computer Forensics Tool Testing Program [150], which comprises five phases in the validation cycle (see Figure 9-1). The phases include defining requirements, defining test assertions, defining test cases, defining test methodology and finally validation testing.

- First, the defining requirements phase has two components, namely core requirements and optional requirements, and they define the software in terms of functionality and features. The core requirements specify the components that the software needs and must contain, whereas the optional requirements are elements that will be nice to have.
- The second phase involves defining test assertions. This means that tests are defined on what functionality to check and what the expected outcome should be to ensure the correctness of the software. Test assertions also comprise two components for optional and core test assertions and map to the core and optional requirements.
- Once it is known what should be tested, the third phase is to define test cases that specify the testing conditions, environment and how these assertions will be tested.
- The fourth phase is defining test methodology, in other words the method used for conducting the tests.
- The last phase is validation testing, where the reflection of the testing process is validated by checking to see if the test assertions are satisfied and the core requirements have been met.

Since the prototype system was separated into two parts (Collection and Storage), the validation of each part is presented with Secure Collection (SC) mapping to the collection tool (W2RC) and Secure Storage (SS) mapping to the storage tool (W3RS). The validation criteria are further explained in the subsections to follow.

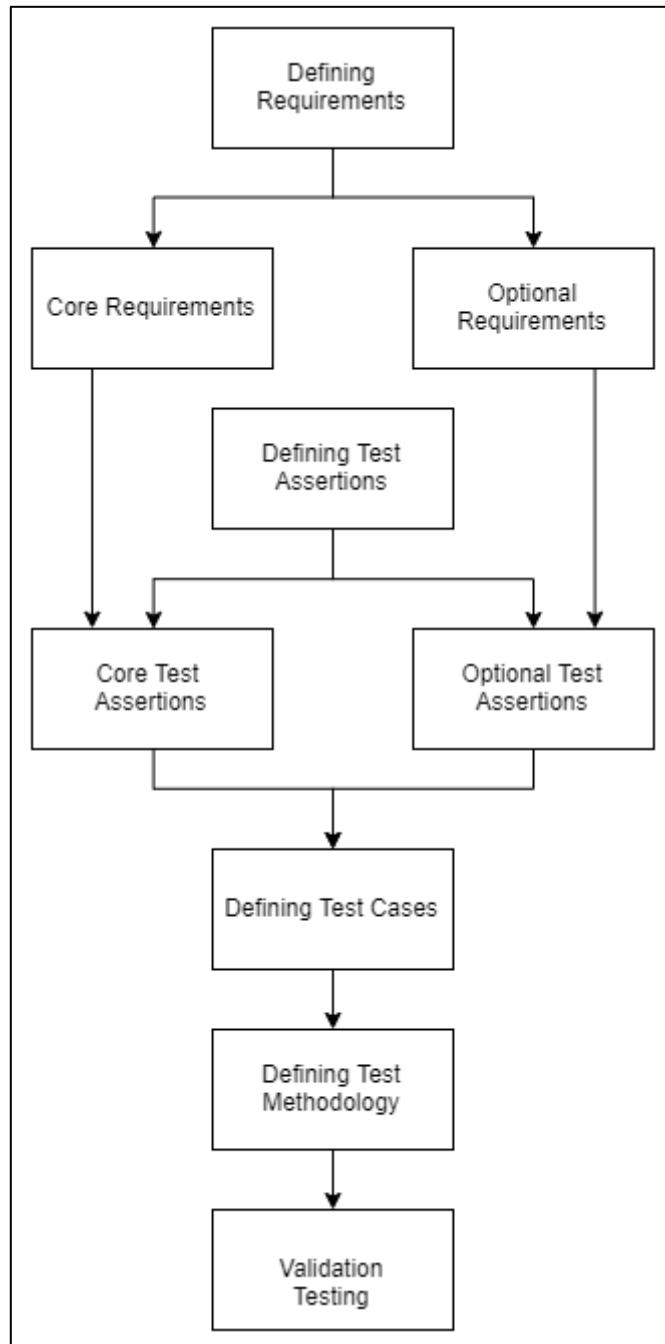


Figure 9-1. NIST validation cycle

### 9.2.1 Secure collection validation

This section was validated based on the NIST validation cycle and is structured as follows: Secure Collection Core Test Assertions (SC-CA); Secure Collection Test



Cases (SC-TC); and the Secure Collection Compliance Matrix (SCCM). Recall that the collection requirements specifications were defined in Section 5.3 prior to the implementation phase of the tool.

#### 9.2.1.1 Secure Collection Core Test Assertions (SC-CA)

- **SC-CA-01:** The tool shall hash each database as well as log and display these hashes each time the tool is run.  
**Justification:** This ensures the verifiability, reliability and integrity of the databases to prove that they have not been modified in any way.
- **SC-CA-02:** The tool shall detect when the storage and analysis server is not online.  
**Justification:** This ensures that the analysis and storage can be performed, and that no unexpected circumstances will arise.
- **SC-CA-03:** The tool shall log any user actions as well as internal processes that the tool is performing.  
**Justification:** This ensures the repeatability as well as verifiability of an incident where the authenticity of the analysis is required.
- **SC-CA-04:** The tool shall verify the collected information before the storage process occurs.  
**Justification:** This ensures the reliability, verifiability as well as authenticity of the collected information that is going to be sent to the server. Identifiers are also added to the collected information to ensure trackability.

#### 9.2.1.2 Secure Collection Test Cases (SC-TC)

- **SC-TC-01:** Start the monitor normally and see if processes are being monitored.
- **SC-TC-02:** Verify the logs and the hashes present to ensure the integrity of databases.
- **SC-TC-03:** Modify the storage server destination to see if it can store elsewhere.
- **SC-TC-04:** Check if monitoring does not continue if the server cannot be reached.
- **SC-TC-05:** Resume a process that was sent for analysis.
- **SC-TC-06:** Remove a process from the seen database and check if it is monitored again.
- **SC-TC-07:** Verify the timestamps and task identifier.
- **SC-TC-08:** Check if the analysis is performed and stored successfully.
- **SC-TC-09:** Run the tool on different versions of Windows OS to see if it works.
- **SC-TC-10:** Perform analysis on network architectures.
- **SC-TC-11:** Test using running benign programs and determine the result.
- **SC-TC-12:** Test using running ransomware programs and determine the result.
- **SC-TC-13:** Run more than one program and determine if the tool can detect and process concurrently.
- **SC-TC-14:** Add a safe process and check if it is analysed.

The testing methodology adopted was to install the collection tool on a Windows 10 and Windows 7 machine and perform the test cases that were defined. To perform validation testing, a compliance matrix was adopted from Zareen et al. [182] where the authors mapped the requirements to the test cases and compared the results of the testing process to the test assertions (defined in a tabular format). This compliance matrix is further discussed in the next section.

### 9.2.1.3 Secure Collection Compliance Matrix (SCCM)

The compliance matrix provides a tabulated result to see if the test assertions were met and complied with the requirements that had been defined, and to determine how the implementation compared to the specified requirements. Recall that the requirements for Secure Collection were defined in Section 5.3. For convenience, the requirements are duplicated in Table 9-1 to avoid looking back and forth. This compliance matrix serves to identify limitations as well as to verify that a viable implementation is presented. The results for the compliance matrix are presented in Table 9-1, and each requirement defined has an associated test case.

For each test case performed, a test assertion provides validation and verification that the requirement was met. This is represented in the table by providing the test assertion label or by performing a manual check operation. For example, the first row (number 1) in Table 9-1 checks if the requirement SC-CR-01 is satisfied by the associated test assertion for the corresponding test case SC-TC-01. In this specific case, no test assertion was defined because this was a simple manual check to see if the requirement is met. The 'Result of Test Assertion' column can have three feasible outcomes, the label of the test assertion, 'check' and 'non-compliant', where 'check' means that a manual test was conducted, and the conditions were satisfied. 'Non-compliant' means that the test was not satisfied, and the requirement was therefore not met. From this table, there were no 'non-compliant' test assertions, which suggests that all the requirements defined have been fulfilled.

**Table 9-1.** W2RC SCCM

<b>Number</b>	<b>Requirement</b>	<b>Test Case</b>	<b>Result of Test Assertion</b>
1	SC-CR-01 - The tool shall monitor all running processes.	SC-TC-01	--check--
2	SC-CR-02 - The tool shall perform logging at every action/process that occurs.	SC-TC-02	SC-CA-01
3	SC-CR-03 - The tool must collect information concurrently.	SC-TC-13	--check--
4	SC-CR-04 - The tool must show consistency in the collection.	SC-TC-02, SC-TC-08, SC-TC-04	SC-CA-01, SC-CA-02, SC-CA-03, SC-CA-04
5	SC-CR-05 - The tool must suspend a new process once detected.	SC-TC-11	SC-CA-02

<b>Number</b>	<b>Requirement</b>	<b>Test Case</b>	<b>Result of Test Assertion</b>
6	SC-CR-06 - The tool must efficiently collect the executable and send it for analysis.	SC-TC-08	SC-CA-04
7	SC-CR-07 - The tool shall perform quick data processing and data handling.	--check--	--check--
8	SC-CR-08 - The tool must show what processes are currently being analysed.	SC-TC-01	SC-CA-03
9	SC-CR-09 - The tool must provide hashes of the database to ensure integrity.	SC-TC-02	SC-CA-01
10	SC-CR-10 - The tool must distinguish between previously seen processes and newly created processes	SC-TC-05, SC-TC-06	SC-CA-03, --check--
11	SC-CR-11 - The tool must list the time and the name of the process it found.	SC-TC-01	--check--
12	SC-CR-12 - The tool must securely send the collected information to the storage server.	SC-TC-02, SC-TC-03, SC-TC-07, SC-TC-08	SC-CA-03, SC-CA-04
13	SC-CR-13 - The tool must work on all Windows NT platforms.	SC-TC-09	--check--
14	SC-CR-14 - The tool must work on 32-bit and 64-bit systems.	SC-TC-01	--check--
15	SC-CR-15 - The tool must request administrator privileges.	SC-TC-01	--check--
16	SC-OR-01 - The tool must allow a user to resume a process.	SC-TC-05	--check--
17	SC-OR-02 - The tool must allow a user to add safe processes.	SC-TC-14	SC-CA-03
18	SC-OR-03 - The tool must provide the status of the analysis of the process.	--check--	--check--
19	SC-OR-04 - The tool must provide an option to separate storage server from the analysis server.	SC-TC-03	SC-CA-02
20	SC-OR-05 - The tool must provide the ability to see the CAT value of analysed processes.	SC-TC-08	--check--
21	SC-OR-06 - The tool must display notifications when a process is being analysed.	--check--	--check--
22	SC-OR-07 - The tool must remove a process from seen and whitelisted database.	--check	--check--

## 9.2.2 Secure storage validation

This section was also validated using the NIST validation cycle and is structured as follows: Secure Storage Core Test Assertions (SS-CA), Secure Storage Test Cases (SS-TC) and the Secure Storage Compliance Matrix (SSCM). Recall that the requirements specifications for Secure Storage were defined in Section 6.3 before the implementation phase of the tool.

### 9.2.2.1 Secure Storage Core Test Assertions (SS-CA)

- **SS-CA-01:** The tool shall encrypt the PDE.  
**Justification:** This ensures confidentiality and circumvents acts of unauthorised access to the PDE.
- **SS-CA-02:** The tool shall hash the PDE before and after the encryption process.  
**Justification:** This ensures the integrity of the PDE before and after the encryption process.
- **SS-CA-03:** The tool shall log any user actions as well as internal processes that the tool is performing.  
**Justification:** This ensures reliability as well as verifiability in an incident where the authenticity of the analysis is required.
- **SS-CA-04:** The tool shall sanitise the data ingested.  
**Justification:** This ensures the stability of the system and reduces the attack vectors from a system compromise through injection attacks like SQL and XSS injection.

### 9.2.2.2 Secure Storage Test Cases (SC-CA)

- **SS-TC-01:** Send a POST request with the relevant data and verify that a successful entry was added.
- **SS-TC-02:** Send XSS strings with SQL and JS injection and see if they are sanitised.
- **SS-TC-03:** Send an incorrect API token and check if the POST request is denied.
- **SS-TC-04:** Send invalid data to see if the validation process works.
- **SS-TC-05:** Perform hashing on the stored PDE and check if it matches the stored hash digest.
- **SS-TC-06:** Download the PDE and see if it matches the database hash digest.
- **SS-TC-07:** Verify the timestamp of the database and the file timestamp.
- **SS-TC-08:** Perform URL manipulation to attempt to download PDE.
- **SS-TC-09:** Try to download PDE without 2FA authentication enabled.
- **SS-TC-10:** Verify if 2FA works as expected.

### 9.2.2.3 Secure Storage Compliance Matrix (SSCM)

The compliance matrix for secure storage is structured in the same way as the compliance matrix for secure collection. (Recall that the requirements for Secure Storage were defined in Section 6.3.) Table 9-2 presents the compliance matrix for

secure storage, which was also compliant (like the matrix for secure collection), because all the results of the test assertions have been satisfied. This implies that all the requirements defined for the secure storage have been met.

**Table 9-2.** W3RS SSCM

<b>Number</b>	<b>Requirement</b>	<b>Test Case</b>	<b>Result of Test Assertion</b>
1	SS-CR-01 - The tool shall ingest data from an API endpoint. SS-CR-03 - The tool must ingest data concurrently.	SS-TC-01	SS-CA-04
2	SS-CR-02 - The tool shall perform logging at every action/process that occurs.	SS-TC-01, SS-TC-02	SS-CA-03
3	SS-CR-04 - The tool must show consistency in data storage. SS-CR-05 - The tool must hash the ingested data.	SS-TC-05	SS-CA-02
4	SS-CR-06 - The tool must sanitise data ingested.	SS-TC-02	SS-CA-04
5	SS-CR-07 - The tool shall perform hashing on the collected data. SS-CR-08 - The tool must show the hash digest and metadata. SS-CR-09 - The tool must provide digests of the encrypted PDE to ensure integrity. SS-CR-10 - The tool must distinguish between different PDE.	SS-TC-01, SS-TC-05	SS-CA-01, SS-CA-02
6	SS-CR-11 - The tool must list the information collected.	--check--	--check--
7	SS-CR-12 - The tool must validate the data ingested. SS-CR-13 - The tool must verify user authentication details.	SS-TC-03, SS-TC-04, SS-TC-07, SS-TC-08	SS-CA-02

Number	Requirement	Test Case	Result of Test Assertion
	SS-CR-14 - The tool must securely download PDE.		
8	SS-OR-01 - The tool must encrypt all metadata.	SS-TC-01	SS-CA-01
9	SS-OR-02 - The tool must decrypt PDE on access.	SS-TC-06	SS-CA-02
10	SS-OR-03 - The tool must list all the stored PDE. SS-OR-04 - The tool must clearly show detected malicious PDE.	--check--	--check--
11	SS-OR-05 - The tool must perform 2FA authentication for PDE download.	SS-TC-09, SS-TC-10	SS-CA-03, --check--

Now that the prototype system has been validated from a software perspective, it is necessary to get experts in the field to review the system and pass judgment. As part of this evaluation process, expert reviews were conducted to gauge how industry-leading professionals would react to the prototype system and the notion in general. The next section discusses how the expert review process was conducted.

### 9.3 Expert review process

Expert review or opinion has been widely used in proof-of-concept tool evaluation, particularly in the forensic domain where actual application knowledge and usability processes are essential. This logic is also supported by studies that observed that a smaller group of experts in a given discipline provides a more insightful evaluation than do larger groups of randomly sampled participants [180]. In this context, the expert review can be compared to the classical process of content validity. The expert review provides a means to evaluate the usefulness and significance of the proposed framework and the proof-of-concept tools developed in this study. The current study therefore adopted expert review as one of the approaches for evaluating the developed proof-of-concept tools. The observation in [181] asserts that the evaluation process comprises essentially of two principles: ‘understanding the problem environment’ (that the tool must address) and ‘understanding the tool’. The developed prototype tools attempt to address the problem of volatile potential digital evidence in a ransomware investigation (which generally falls within the digital

forensic environment). Therefore, the first principle can be used to identify relevant experts who can participate in the evaluation process. These participants will provide an evaluation of the relevance and potential usefulness of the developed tool, which addresses the second principle of “understanding the tool”.

In order to develop a systematic approach towards the expert review process, a system science approach to tool evaluation was adopted [183]. The latter is a theory-based evaluation process that considers the holistic composition and formative design of a given proof-of-concept tool within the context of the discipline. Therefore, a system science approach is a systematic evaluation model [183] [184] that considers each component of a system and how each component interacts to achieve the overall intended goal of the tool. The detailed process used in the evaluation of the tool is presented in the rest of this section. First is the process of identifying appropriate respondents for the evaluation, followed by the process of adapting the measurement instrument for expert review. Data collection and result analysis constitute the last subsection of this section.

### **9.3.1 Respondent identification**

Respondents considered for this study include digital forensics experts and security experts in digital forensics and security organisations who are based in South Africa. The purposive convenient sampling technique [185] that was used in this study, is based on choosing a participant that possesses all the necessary qualities, knowledge or experience. Given the limited expertise in the field of digital forensics and security in South Africa, the expected sample size for the study was projected to range from 5 to 20. This projection considers the logic that some organisation may not provide support for such evaluation as a result of time constraints, potential legal and policy concerns, as well as non-response. However, an expert review size of five experts or more were considered in this study – as in other studies [180] [184] – on the use of an expert review for tool evaluation.

In order to enhance the ease of use of the developed tool and to aid the evaluation process, an installation guideline was developed. An automated approach towards tool utilisation was also considered to minimise the manual process required for the usage of the developed tool. Details of the installation are given in Appendix B. To simplify the setup involved for the expert reviewers to go through the W3RS, an analysis server was set up and hosted on a VPS. Therefore, only the W2RC tool needed to be installed, which was already preconfigured to connect to the relevant server. The tool and the measurements were sent to the organisation that had initially indicated their potential to participate in the evaluation process. The expert response was terminated after the fifth response was received and lasted for four weeks, between July and August 2019. The respondents included two cybersecurity specialists working with the Security Operation Centres (SOCs) of two banks in South Africa, one cybersecurity analyst at the SOC of a global cybersecurity

company with headquarters in Israel, a security software developer, and a security analyst at a digital security and forensics company.

The respondents were asked to provide an unbiased evaluation based on their experience of security and forensic tools such as classical end-point intrusion detection and response systems, and knowledge of SOCs. The current study assumes that such knowledge is leveraged by the respondents to evaluate the relevance and potential of the developed tools (W2RC and W3RS).

### 9.3.2 Measurement item development

In information system research, measurement items that comprise responding to questionnaires are rooted in theories and are often used to yield knowledge and perform evaluation. The measurement items considered in this study were based on two information system constructs: the constructs from the theory of PC utilisation [186] and computer self-efficacy theory [187]. These constructs were considered relevant and suitable for the evaluation of the tool. However, the instruments from each construct were further modified to reflect the context and content of the W2RC evaluation process. A summary of the original and adapted measurement instruments for the study is presented in Table 9-3.

**Table 9-3.** Measurement instrument for tool evaluation

S/N		Original item	Adaptation
Computer Self-Efficacy: I could complete the job using the software package (I could perform ransomware investigation using the investigation tools)			
Q1		if there was no one around to tell me what to do	if there was no one around to tell me what to do
Q2		if I had never used a package like it before.	if I had never used a package like it before.
Q7		if I had a lot of time to complete the job for which the software was provided.	if I had a lot of time to perform ransomware investigation on the infected system.
<b>Model of PC Utilisation (MPCU)</b>			
C1	Complexity	Using a PC takes too much time away from my normal duties.	The use of the proof-of-concept (POC) tools to perform digital investigation can help to reduce the time taken for an investigation.
C2		Working with PCs is so complicated, it is difficult to understand what is going on.	Working with the POC tools is complicated, as I don't seem to understand what is going on.
C3		Using a PC involves too much time doing mechanical operations (e.g. data input).	Using the POC tools consumes too much time (i.e. doing the installation and manual inputting).
JF1	Perceived job fit	Using a PC will have no effect on my performance of my job (reverse scored).	Using the tools will have no effect on the process of ransomware investigation.



S/N		Original item	Adaptation
JF2		Using a PC can decrease the time needed for my important job responsibilities.	The use of the tools can reduce the time required for me to conduct a ransomware investigation.
JF3		Using a PC can significantly increase the quality of output of my job.	The use of the tool can improve the efficiency of my investigation process.
JF4		Using a PC can increase the effectiveness of performing job tasks (e.g. analysis).	The use of the tool can increase the potential of recovery from a ransomware attack without paying the ransom.
JF5		A PC can increase the quantity of output for the same amount of effort.	The use of the W3RC tool can generate potential digital evidence that can be used to conduct a ransomware investigation.
JF6		Considering all tasks, the general extent to which the use of PC could assist on the job.	Considering all tasks, the general extent to which the use of the tool can assist in ransomware investigation.
LT1		Long-term consequence	Use of a PC will increase the level of challenge on my job
LT2	Use of a PC will increase the opportunity for preferred future job assignments.		The use of the tool could create an opportunity for the prevention of future ransomware attack.
LT4	Use of a PC will increase the opportunity for more meaningful work.		The use of the tool will increase the potential to gather more digital artefacts for ransomware investigation.
LT6	Use of a PC will increase the opportunity to gain job security.		The use of the tool will strengthen the security of the system against a ransomware attack.
<p>*The instructions to the respondents for these items were: "In this section, we wish to determine how useful you believe a personal computer could be for your current job responsibilities. Please tell us how much you agree or disagree with each of the following statements (1 = strongly disagree; 2= somewhat disagree; 3= neither agree nor disagree; 4= somewhat agree; 5= strongly agree)." (Note: The instructions and scale anchors differed for other constructs.)</p>			

Demographic variables that represent the type of organisation and technical competency of the respondent were also included in the measurement instrument.

### 9.3.3 Data analysis and presentation

A total of five expert reviews were received and used for the evaluation process. Descriptive statistics were used to analyse the results based on the construct of the PC utilisation model. Structural equation modelling could be considered for a sample

size smaller than 200 instances by using a bootstrapping technique. However, given the relatively smaller sample size and the nature of expertise of the respondents, a descriptive statistic of the response in relation to the constructs was considered. This suggested that the summary of the descriptive statistics, in accordance with the theoretical underpinning of the constructs, can express the response of the respondents. To this effect, the analysis process was further categorised based on two constructs: a model of PC utilisation, and a model of computer self-efficacy.

### 9.3.4 Results of the model of PC utilisation

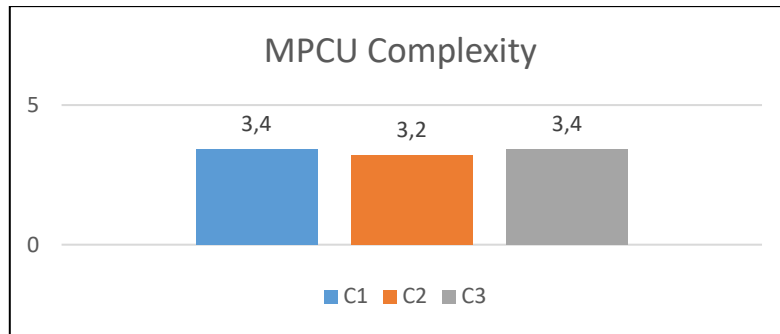
Three subconstructs from the model of PC utilisation (MPCU) theory were considered relevant to the evaluation process adopted in this study, based on the applicability of the measurement items to the tool and the context of the study. As highlighted in Table 9-4, Table 9-5 and Table 9-6, the MPCU subconstructs include complexity, job fit and long-term consequence of the tool. A brief description is given of the concept of each subconstruct, after which the statistical description of the response for each subconstruct will be discussed in the subsequent subsections.

#### 9.3.4.1 MPCU complexity

Complexity, in the context of this study, is conceptually defined as the degree of success with which the developed W2RC and W3RS tools are perceived. This ranges from relatively difficult to easy to understand and use the tools to perform digital forensic investigations. Intuitively, complexity is indirectly proportional to acceptance. This implies that the higher the perceived complexity of the W2RC and W3RS, the lower or more restricted their adoption and utilisation. As highlighted in Table 9-4, three measurement items were selected to evaluate the complexity of the proposed tools. The items measured the impact of the complexity of the proposed tools on conducting a ransomware investigation. A summarised version of the expert review is presented in Table 9-4. Items C2 and C3 are reverse scales. A reverse scale (R) implies that the observed response should be interpreted in the reverse order of magnitude as seen in Figure 9-2.

**Table 9-4.** Response statistics of MPCU complexity

Items	Description	Mean score
C1	Using the W2RC tools to perform digital investigation can help to reduce the time needed for the investigation	3.4
C2	Working with the W3RC tools is complicated, as I don't seem to understand what is going on (R)	1.8 (3.2)
C3	Using the tools consume too much time doing installation and coding (R)	1.6 (3.4)



**Figure 9-2.** MPCU complexity graph

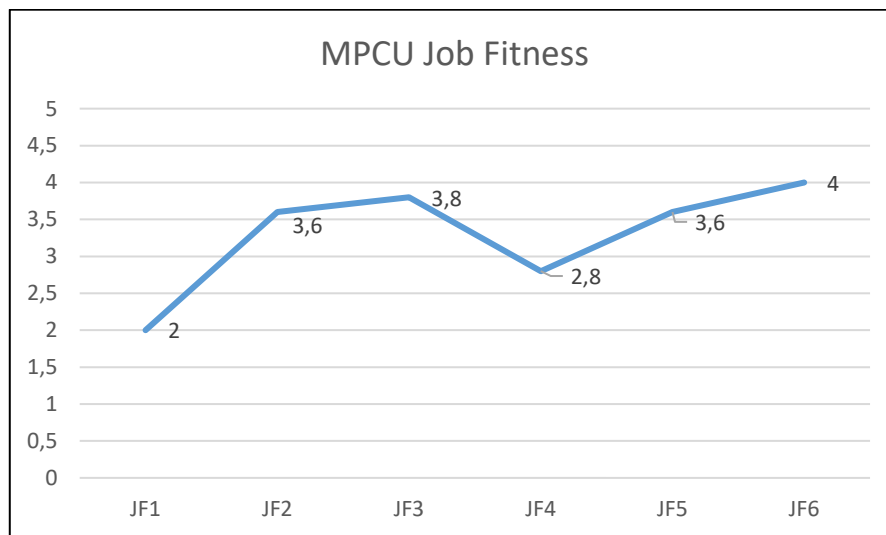
As shown in Table 9-4, the average mean score for each measurement item is greater than 3.0 out of 5. This implies that the experts agree that the use of the proposed tools can potentially (positively) influence the time required for a ransomware investigation. For instance, a mean score of 3.4 for C1 indicates that, on average, each expert agrees that the W2RC tool can reduce the investigation time required to perform ransomware forensics. Similar inferences can be made with the other two items. Thus, with respect to time complexity and investigation time, the proposed tool has the potential to improve the ransomware investigation process.

#### 9.3.4.2 MPCU job fitness

Six measurement items were used to form the job fitness subconstruct. A job fitness subconstruct is defined, in the context of this study, as a measure of the extent to which a forensic (and or security) expert believes that using the developed W2RC and W3RS tools can enhance their performance in conducting an investigation. A job fit subconstruct provides metrics for the evaluation of the appropriateness and effectiveness of the proposed tool. In the context of a ransomware investigation, job fit evaluates the potential of the proposed tool to prevent a ransom payment. This perception is particularly captured by the JF4 measurement item (see Table 9-5). There was consensus among the experts that the current state of the tool presents little probability of preventing ransom payment. This consensus is captured by the unanimous rating of “somewhat disagree” for item JF4 (average rating of 2, as shown in Table 9-5). However, the responses in respect of items JF5 and JF6 support the proposition that the integration of a readiness approach to ransomware forensic has the potential to enhance ransomware investigation. This logic is supported by all the experts with a rating of “agree” (averaged rating of 4). Item JF2 further extends the efficiency expectations of the proposed study. A graphical representation of this MPCU Job Fitness can be seen in Figure 9-3. The expert opinion supports the assertion that the proposed approach is fit for the job of a digital forensic investigator and can significantly improve the investigation process.

**Table 9-5.** Response statistics of MPCU job fitness

Items	Description	Mean score
JF1	Use of the tools will have no effect on the process of ransomware investigation (R)	2
JF2	Use of the tools can reduce the time required for me to conduct a ransomware investigation	3.6
JF3	Use of the tool can improve the efficiency of my investigation process	3.8
JF4	Use of the tool can increase the potential of recovering from ransomware without paying the ransom	2.8
JF5	Use of the W3RC tool can generate potential digital evidence that can be used to conduct a ransomware investigation	3.6
JF6	Considering all tasks, the general extent to which the use of the tool can assist in ransomware investigation	4



**Figure 9-3.** MPCU job fitness graph

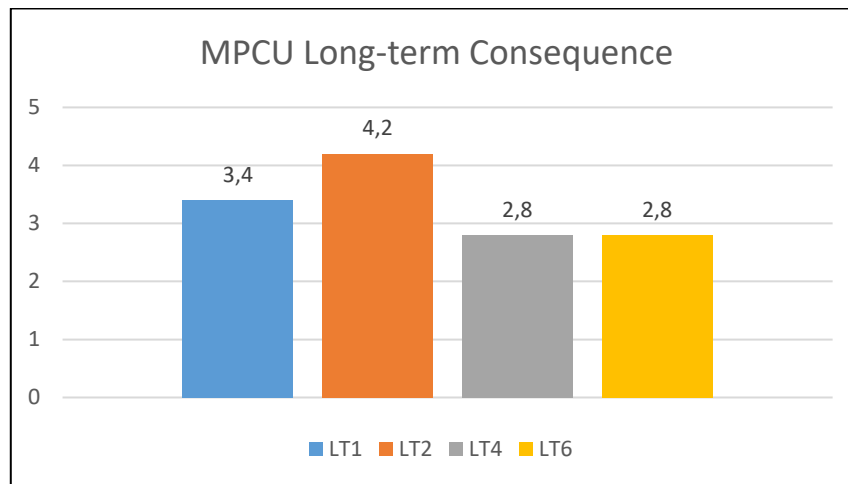
### 9.3.4.3 MPCU long-term consequence

One approach used in evaluating technological innovation is the long-term effect of a given technology. The long-term consequence subconstruct measures the potential of the W2RC and W3RS tools for ransomware investigation in the long term. In addition, this subconstruct attempts to evaluate the potential of the proposed tools to address future forensic challenges. Naturally, the long-term consequence is suggested to have a positive relationship with technology utilisation. Item LT2 (as shown in Table 9-6) captured the futuristic tendency of the proposed tool. The average response of the respondents shows that the proposed tool has the capacity to address future ransomware attacks. A graphical representation for this is shown in Figure 9-4. Consequently, the use of the proposed approach as part of ransomware forensics has the potential to reduce the risk of a ransomware attack.

**Table 9-6.** Response statistics of MPCU long-term consequence

Items	Description	Mean score
LT1	Use of this tool will complicate the investigation of a ransomware attack (R)	3.4
LT2	Use of the tool could create an opportunity for the prevention of future ransomware attacks	4.2
LT4	Use of the tool will increase the potential to gather more digital artefacts for ransomware investigation	2.8
LT6	Use of the tool will strengthen the security of the system against a ransomware attack	2.8

However, the experts assert that the use of the proposed tool could potentially introduce forensic analysis complexities. Data storage constitutes a major concern that emerged from the expert review. Furthermore, an increase in the rate of false positives and the tendency to store evidence with potentially no forensic value constitute the main reason for this assertion.



**Figure 9-4.** MPCU Long-term Consequence

The responses on items LT1 and LT4 reveal that the experts believe that the proposed tool could generate excessive data over a long period of time. Logically, the combination of the potential of a big data challenge and induced complexities will negatively affect the potential for strengthening data security in the long run. This problem is indicated by the response to item LT6. Whilst such complexities and the big data problem might not significantly affect the forensic process, they can potentially have a negative effect on security. Given that security strengthening constitutes one of the major applications of forensics, there is a need to address this as a potential future challenge of the proposed approach.

In summary, the developed W2RC and W3RS tools satisfied the evaluated criteria for forensic readiness solution for ransomware investigation.

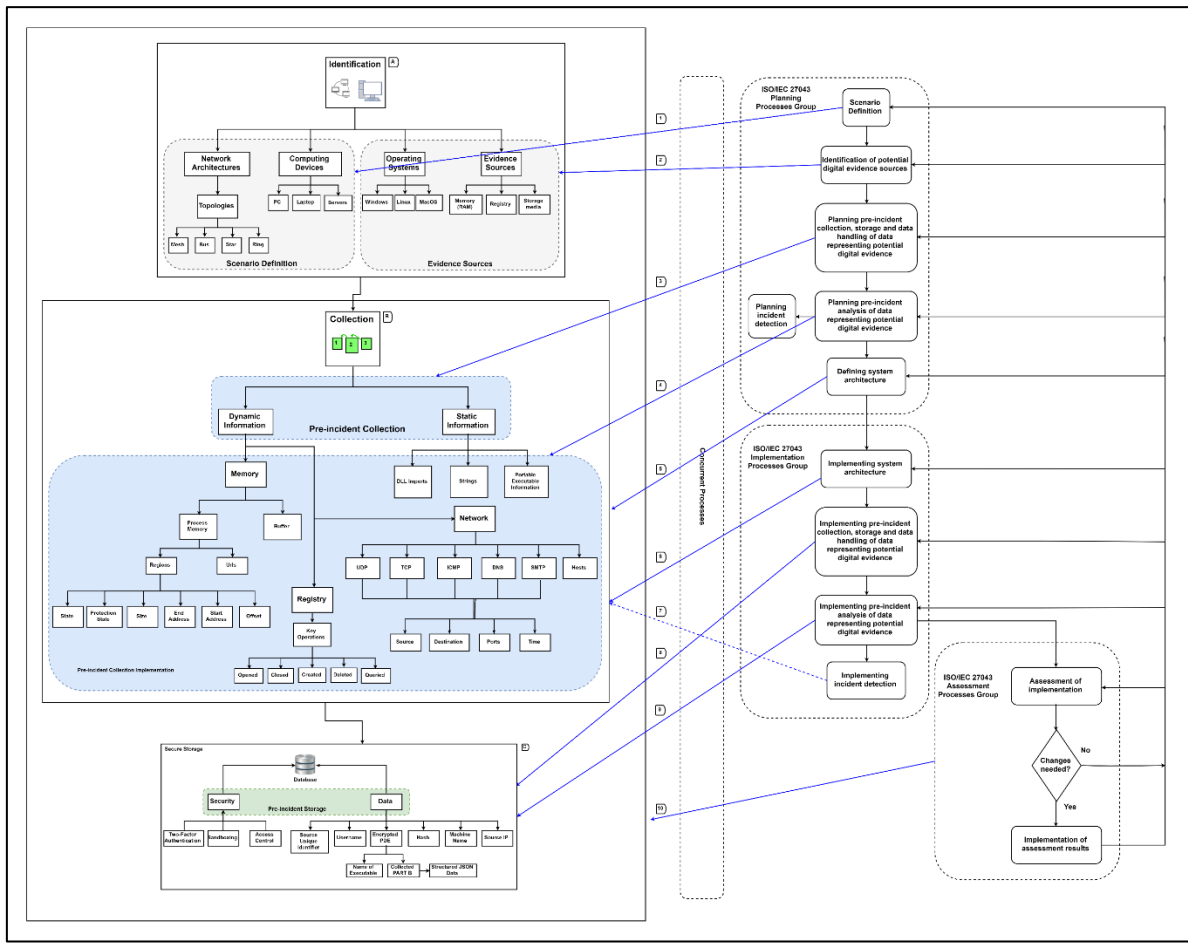
## **9.4 Mapping of the proposition to a Digital Forensic Standard**

A benchmarking approach towards framework evaluation is presented in this section. In order to benchmark the developed framework and tool, the ISO27043:2015 standard was considered [188]. This approach can be compared to external and criterion validity. The discussion begins with the benchmarking of the developed framework to the ISO/IEC 27043:2015 standard, followed by evaluation of the propositions made in this study, as well as of existing digital forensic readiness frameworks and process models.

### **9.4.1 Mapping of the proposed framework to ISO/IEC 27043:2015**

ISO/IEC 27043:2015 is an internationally standardised digital forensic readiness process model that suggests the process to be followed by an investigator (and or any security-related information technologist) to ensure the availability of potential digital evidence. Furthermore, the model simplifies the process of proactive forensic practice and the need for a readiness approach towards information acquisition and preservation, specifically in a volatile environment. A logical mapping of the proposed framework to the ISO/IEC 27043:2015 standard is presented in Figure 9-5. Recall that the readiness processes from the ISO/IEC 27043:2015 were discussed in Section 2.4, and that the relation to each process was mapped to the Ransomware Readiness Framework (RRF). A brief example of how this mapping is interpreted shows that the Scenario Definition of the ISO 27043 maps to the Network Architecture and Computing Devices phase of the RRF, represented as (1) Figure 9-5. Another example is that the Identification of Potential Digital Evidence Sources is mapped to the Operating Systems and Evidence Sources of the RRF framework, represented as (2). Further mappings of each process in the ISO 27043 can be interpreted in a similar fashion. The mapping shows that the proposed framework satisfied the requirement for a digital forensic readiness mechanism. Particularly, a mechanism is provided for the evaluation and potential extraction of a ransomware decryption key in a malware investigation.

This subsection further evaluates the suitability of the developed tool towards realising the proposed framework.



**Figure 9-5.** Mapping of the Ransomware Readiness Framework to the ISO/IEC 27043:2015 International Standard

The next section discusses works that are related to the proposed idea and framework. It serves as part of an evaluation to see how this research relates to existing literature and how it differs for the betterment of the body of knowledge.

## 9.5 Related literature

The aim of the research in hand was to provide a novel approach to ransomware detection as well as to collect forensically relevant information to aid digital forensic investigators in conducting timeous investigations. Since nothing specific to ransomware and digital forensic readiness had been proposed at the time, this section compares related literature to gauge the importance of this research and the gap that it fills in respect of ransomware detection.

Significant contributions have been made towards detecting ransomware by using Artificial Intelligence (AI), behavioural signatures, API call graphs and network activity [189]–[192]. Using artificial intelligence provides a competitive advantage as dynamic behaviour can be detected in real time. However, this is very limiting as these intelligence mechanisms can easily be fooled. For example, a well-known

anti-virus company Cylance [193] who boasts that they are the leading AI-driven anti-virus, was bypassed by researchers fairly easily by just adding random data at the end of the malicious executable file, thereby changing some of the detection characteristics [194] [195]. Another bypass was achieved when researchers masked malware by using video game code and the malware embedded within, and Cylance was not able to detect it [196]. Other works used deep learning as proposed by Tseng et al. [197], in which the authors used deep learning AI algorithms to detect malicious behaviour based on network traffic. This approach had merit, however, it did not cater for infections that happen off-network, for example if a user deliberately ran or was tricked into running ransomware through a USB device. The proposed Ransomware Readiness Framework (RRF) caters for both physical execution or network execution, as somewhere along the line the ransomware needs to be executed and that has to be done through an executable file. Therefore, by capturing all new processes, the current research caters for various ways of attack. A survey on deep learning and AI was conducted by Kwon et al. [198] where the authors looked at some of the techniques and algorithms used to detect anomalies and proposed a generic template of how anomaly classification can be achieved. This template consisted of various factors for intelligence which included neural networks, fuzzy logic, genetic algorithms, support vector machines and decision tree classifiers. Neural networks focused both on supervised and unsupervised learning for the neural networks.

Other methods of detection involved the work of Jung and Won [9], who proposed the idea of context-aware entropy – similar to the Entropy Monitoring (EM) context-based analysis proposed in this research. This method by Jung and Won calculates the entropy of each file before any operations (e.g. read/write) are performed on it, backs up the file in a safe file system and again performs the entropy of the file after any operation was carried out. If the entropy changed significantly, it would imply that the file underwent encryption. Thereafter, the headers of the file were analysed to determine whether abnormal or normal encryption occurred, where abnormal encryption is most likely to be ransomware. While this approach is good for detecting ransomware attacks using context-aware technology, it is very intensive in processing and data storage. This is because each file would need to be backed up. With the W2RC tool, this does not become a problem as analysis and storage are happening at a centralised place within the organisation with a specific focus on Digital Forensic Readiness (DFR). The work by Jung and Won furthermore does not provide any mechanism for automated incident response or collect any data that might have forensic value.

Related literature that used entropy [9] [63] [199] as a means of detection managed to achieve good accuracy for ransomware detection; however, they rely on backing up files. Therefore, entropy is by far the best method that can be used to detect encryption, which is the major objective of ransomware attacks. The proposed research also asserts that entropy is a good indication of encryption; however, when



combined with other metrics like registry and API calls, it provides more insight into performing accurate detection.

The next section provides an overview of what works have used DFR to enable timeous investigations and to provide investigators with a data repository to corroborate the evidence that might have not been present post-incident.

### **9.5.1 Digital Forensic Readiness**

Digital Forensic Readiness (DFR) aims to minimise the cost of an investigation while maximising the amount of Potential Digital Evidence (PDE) that is collected [60]. Many frameworks for DFR have been used to address challenges faced during investigation and litigation, for instance behavioural biometrics [200], public key infrastructure systems [201], cloud [202] and IoT [203]. Unfortunately, at the time this research was conducted, there were no DFR frameworks relating to ransomware. Due to the fact that it is the most dangerous form of malware, several detection methods exist for ransomware; however, nothing has been done to collect PDE that can be used during litigation. Related literature on DFR frameworks can be compared to make the idea behind each framework and the usefulness of DFR clear. For example, the work by Adeyemi and Venter [35] designed a DFR framework to incorporate behavioural biometrics within an organisation. This framework consisted of four phases: data acquisition, preservation, user authentication, and user pattern attribution phase, and each phase was subsequently evaluated with the ISO/IEC 27043 international standard.

Another work – by Valjarevic and Venter [42] – created a framework for Public Key Infrastructure (PKI) to aid organisations with integrating PKI systems. This framework provided guidelines on how PDE can be maximised to preserve and improve the level of information security within the PKI. Wireless devices have become an integral part of our daily lives, and the work by Mouton and Venter [204] presented a prototype to achieve DFR for wireless sensor networks. Their prototype also presented requirement specifications for its implementation so that the prototype tool could be reproduced. These requirements were drafted from real-world demonstrations of sensor networks. In a similar fashion, W2RC and W3RS in the current study also presented requirements specification and architectural requirements, based on real-world requirements of tools. All the DFR frameworks reviewed were evaluated in line with the ISO 27043 international standard. Therefore, each framework reviewed from related literature provides a cornerstone guide to successfully implement DFR within an organisation.

The next section focuses on ransomware investigation and on some of the key findings that support the fact that ransomware investigation is a fairly complex procedure.

## 9.5.2 Ransomware investigation

Several dissections have been made of different ransomware samples over the years, such as Locky, WannaCry, CryptoLocker and others [16] [62]. The work by MacRae and Franqueria [205] provided a detailed analysis of Locky ransomware to understand how it works. MacRae and Franqueria [205] also agreed that performing ransomware investigation is a difficult feat as the information left behind does not leave any traces of the origin of the attack or of ransomware payment tracking. Their work also proposed methods that can be used to track ransomware payment, such as tracking the Bitcoin [206] transactions for the suspected ransomware wallets. This is generally how law enforcement tracks the flow of ransomware payments, and somewhere down the line, it may lead them to the perpetrators when the Bitcoin is exchanged back to fiat currency.

WannaCry hit 2017 by storm leaving many organisations a victim of this attack. The work by Kumar et al. [207] found upon investigation that a key characteristic of ransomware is the ransom note that is always left behind. Secondly, the encryption process always checks a C&C server to determine if there are any updates or if the kill switch is active. This is usually done so that ransomware authors can have control of where the malware should spread. Another commonality is to ensure anonymity of the attacker's host services on the TOR [16] [207] network. This makes it more difficult for law enforcement to shut the attacker down, due to the design of anonymity on the TOR network. Another fundamental characteristic of ransomware is logging or tracing the network activity to determine what external resources the ransomware is contacting. This is why the proposed RRF collects network information. Kumar et al. [207] conducted an investigation by extracting 'strings' from the ransomware executable and determined traces of WannaCry ransomware in the executable itself. Their work [207] therefore proves that the information collected in the RRF provides an investigator with the necessary evidence, as it might not have been there at the time the first responders were present. More comparisons and motivations for each item collected in the RRF were presented in Section 4.3, outlining why the collected information was chosen and why it was relevant.

The next section presents a summary of the related literature that was available by late November 2019 to provide the necessary statistics of how much research has been done. It also indicates the gaps where more research still needs to be conducted.

## 9.5.3 Summary of the findings from related literature

This section presents a summary of the related literature and indicates the amount of literature based on keywords and various literature repositories. Repositories include IEEE Xplore, Springer Link, Taylor and Francis, Science Direct and Scopus. The results of these findings are presented in Table 9-7. The values in brackets represent an exact match, for example for the keyword "Digital Forensic Readiness",

the IEEE Xplore repository found 59 results and for the keyword's exact match, only 30 results were found.

The results in Table 9-7 reveal that little to no research has been conducted on DFR and ransomware detection/investigation. Some of the results in the table come from published papers that emanated from this dissertation. Some research has been done on digital forensics; however, not many of the studies proposed any digital forensics frameworks – as shown by the keyword “Digital Forensic Readiness Framework” with the exact match in brackets. Ransomware investigation and detection is a research area that is not new. However, DFR has not been applied to these areas in detail. This could be a major drawback for the forensic community when trying to find perpetrators.

**Table 9-7.** Summary of related literature findings (2019/11/28)

<b>Keywords</b>	<b>Articles found in repository</b>				
	IEEE Xplore	Springer Link	Taylor and Francis	Science Direct	Scopus
Digital Forensic Readiness	59 (30)	491 (43)	957 (5)	433 (65)	158 (87)
Ransomware Investigation	7 (0)	420 (4)	97 (0)	443 (1)	29 (3)
Ransomware Detection	101 (35)	737 (43)	86 (0)	432 (27)	238 (62)
Digital Forensic Readiness + Ransomware Detection	0	16 (1)	11 (0)	15 (0)	3 (1)
Digital Forensic Readiness + Ransomware Investigation	0	14 (1)	13 (0)	16 (0)	5 (3)
Digital Forensic Readiness Framework	17 (4)	364 (6)	477 (1)	266 (2)	44 (10)
Digital Forensic Readiness Framework + Ransomware Investigation	0	13 (1)	10 (0)	14 (0)	3 (2)
Digital Forensic Readiness Framework + Ransomware Detection	0	15 (1)	9 (0)	13 (0)	1 (0)

The next section concludes this chapter and makes a final comment with regard to what was achieved by means of the evaluation processes.

## **9.6 Conclusion**

The proposed framework and developed tools were critically evaluated in this chapter. Two approaches to evaluation were considered: expert review and benchmarking to international standards. Cybersecurity experts from cybersecurity-related organisations were identified as respondents for the study. Constructs from information systems theory, which criticises technology competency and utilisation, were used to develop measurement items for the evaluation process. The result

supports the underlying findings from the previous chapters, which all attest to the effectiveness of the proposed framework. Furthermore, the benchmarking process reveals that the proposed framework aligns with international best practice, specifically the ISO/IEC 27043:2015 international standard. Therefore, the evaluation process supports the external validity of the developed tools and the proposed framework.

The current research was also compared to the related literature to determine its usefulness and relevance. The summary of related literature clearly indicates that DFR has not been incorporated in organisations when it comes to ransomware. This is an area that needs further research to contribute to the existing body of knowledge and reduce the number of cybercrime attacks that occur every day.

The next chapter concludes the current study by providing an overview of the research performed. It also presents a holistic view of what has been achieved through this research and how it has expanded the body of knowledge to aid the forensic community.

# PART V

## CONCLUSION

## **10. CHAPTER 10: CONCLUSION**

This chapter serves as the conclusion to the dissertation and makes general remarks on how this research has improved the body of knowledge that is relevant to the forensic community. First, a summary of all the chapters is presented and next the problem statement is revisited to show the extent to which it has been addressed. After that follow a discussion of the limitations of this study, the future research work suggested in this field, and finally, a number of final remarks about this research.

### **10.1 Summary of chapters**

Chapter 1 served as the introduction to this research and presented the problem that it intended to solve, as well as the methodology used to solve the research problems. Chapter 2 presented the necessary background literature about digital forensics, the science involved, as well as the traditional investigative process of conducting a digital forensic investigation. Next followed digital forensic readiness and all the processes that it entails. Chapter 3 discussed malware forensics and malware analysis by presenting background to ransomware and the information that is typically needed to perform ransomware forensics successfully. Together these two chapters made up PART II of this dissertation.

Chapters 4, 5, and 6 constituted PART III of the dissertation. Chapter 4 introduced the research framework that was created during the course of this research and that provided the necessary structure and details for collection of relevant information to aid digital forensic investigators to perform timeous investigations. Chapter 5 introduced the W2RC part of the proposed prototype system, which focused on collecting and analysing every newly created process. This part introduced the main model for ransomware forensics and proposed the metrics that were used to actively detect a ransomware attack through Context-Based Analysis (CBA). Furthermore, architectural design and system implementation were also presented in Chapter 5. Chapter 6 introduced the second part of the prototype system (W3RS), which revolved around secure storage of the collected information received from the collection tool. This chapter also presented a process model for secure storage that aided digital forensic readiness storage issues by ensuring forensic soundness of the collected information as well as abiding by information security services.

Chapters 7, 8 and 9 constituted the evaluation part (PART IV) of the dissertation. Chapter 7 provided the results and interpretation of this research in which a series of tests was conducted to obtain an average thresholding value for each CBA and to determine a good range to actively distinguish between abnormal and normal behaviour. Chapter 8 proposed real-world case studies where the scenario of each study was replicated with the proposed prototype installed to determine the real-world use and implications of this research. Chapter 9 comprised a thorough process to evaluate this research through expert review, software verification and validation,

as well as to map the proposed framework to the ISO/IEC 27043 international standard.

Chapter 10 now concludes this report and suggests the future work that should stem from the current research. The limitations of this research are also presented in this chapter.

The section below revisits the problem statement to determine to what extent the current research has addressed the problems presented in Chapter 1.

## **10.2 Addressing the problem statement**

This research aimed to solve the complexities of detecting a ransomware attack and providing digital forensics investigators with the necessary information to conduct timeous investigations. As a result, investigators were provided a data repository to work with, thus making investigation easier and simplifying the corroboration of evidence. The extent to which the proposed subproblems are addressed is further discussed:

- **Q1)** To what degree can a framework/model be created to aid digital forensic investigators to perform a ransomware investigation?
  - This problem has been addressed because the proposed framework was evaluated and tested. A model for ransomware forensics was derived that provides the detail to collect relevant information before, during and after a ransomware attack. This aid digital forensic investigators by providing them with a data repository to perform an investigation.
- **Q2)** What potential digital evidence can be collected from a ransomware attack using Digital Forensic Readiness?
  - Based on the background literature presented in Chapter 3, the necessary information was extracted from the various malware analysis phases and processes used to conduct an investigation. This information was then formulated and built into the proposed framework. It formed part of the groundwork by indicating what information is relevant for collection and what information can be seen as potential digital evidence.
- **Q3)** Can such a framework reduce costs and improve incident response?
  - The extensive evaluation of the proposed framework and prototype shows that this problem has been addressed. Expert reviews confirmed that the implementation of such a framework will allow for better incident detection and response.
- **Q4)** To what degree can ransomware be detected before it causes any permanent damage?
  - Although partly addressed, this problem still requires more research because newer ransomware has an increasingly sophisticated ability to avoid detection. This research was nevertheless able to accurately detect

ransomware attacks and prevent any permanent damage from being done to the system.

- **Q5)** Is there a way to automate the digital forensic process for investigating ransomware?
  - o The proposed framework provided support to automated incident response and investigation by detecting when an incident takes place and alerting a system admin that an incident took place. It prompted an investigator to investigate the matter and provided the necessary data and information about the incident at hand.

The next section discusses the limitations of this research and their implications.

### 10.3 Contributions made by the current research

A summary of the contributions that stemmed from this research follows below:

1. Developed a framework for ransomware forensics.
2. Developed a model for ransomware forensics.
3. Developed a process model for secure readiness storage.
4. Developed a proof-of-concept tool (W2RC) for the collection of Potential Digital Evidence (PDE).
5. Developed a proof-of-concept tool (W3RS) for secure storage of PDE.
6. Implemented an automated incident detection and response framework.
7. Designed a data repository with PDE to aid investigators in conducting a timeous investigation.
8. Presented a digital forensically sound process model for integrity verification and assurance.
9. Developed a novel approach to ransomware detection through Context-Aware Trigger (CAT) technology.

### 10.4 Limitations of this research

This research does not have many limitations as far as the proposed framework is concerned. However, the developed prototype does have some limitations. These limitations are attributed to the scope of the work, since the objective was to provide a proof-of-concept prototype and not to offer a full-fledged solution. The limitations of the developed tools are listed in Table 10-1.

**Table 10-1.** Limitations of this research

Item	Reason
Speed and efficiency	Speed and efficiency were not major requirements of the tool. Since it is a proof of concept, time optimisation was not a core requirement, and there might be more efficient ways to do what the proposed tools perform. Despite the



Item	Reason
	approach not being the most efficient in terms of performance, it was able to accurately perform the task at hand.
Sandboxing	The reason for having an analysis server is that there is no true sandbox with an API that is open source and that provides the ability to be run alongside the machine with limited resources. At the time of starting this research, Windows 10 Sandbox was not released yet, and it is currently available only on Enterprise and Professional versions. However, the proposed Cuckoo sandbox environment can be replicated on the host machine, though it requires a lengthy installation process.
Accuracy	As the proposed tool and framework were developed to collect potential digital evidence (PDE), the accuracy of detection was not a major priority. A few complications arose with regard to certain applications that run only on specific versions of Windows. These were not catered for as they fall beyond the scope of the research and can be attributed to the incompatibility of the executable at hand. More metrics of ransomware protection may be useful to add in the near future.
Detection evasion	While the W2RC tool was able to detect ransomware with high accuracy, there are still limitations to what the tool can do. For example, if the ransomware employs delay tactics where it might know about the 15-second scan at the start of the process, it could just delay execution by 1 minute to prevent the ransomware from being flagged as malicious. However, with some of the static analysis performed, there is a possibility that it can still be detected. The chances of this happening are slim as ransomware already employs evasive techniques (as mentioned in the results).

The next section suggests the future work that may emanate from the current research so as to extend its scope and broaden its coverage.

## 10.5 Future work

Further analyses of ransomware samples are needed to find more unique characteristics and patterns that can be used to create better detection and collection mechanisms. More robust metrics are needed to provide better incident detection and response for PDE collection. The research conducted provides the groundwork for newer studies of ransomware investigation. More automation of digital evidence processing can be performed to further simplify the investigation process. Other research that should be conducted involves the dissection of malware propagation

and the attack vectors to deduce how the expansion of ransomware can be contained. This research explored these aspects on a high level but did not focus on them as they did not fall within the scope of this research. With the advances in malware, it has become more difficult to accurately detect ransomware. Therefore, further research needs to be conducted to improve detection rates, especially of malware evasion tactics. Context-aware technology can significantly enhance the detection of a ransomware attack and the collection of PDE, which closely relates to behavioural analysis and signatures. Instead of focusing on one thing, context-awareness technology focuses on a series of components to be more accurate, based on the current machine's working. While this is an ideal outcome, it is often very difficult to achieve, and further research needs to be conducted in this area.

Finally, some concluding remarks are made about this research and what it has achieved.

## **10.6 Final words**

This research proposed a digital forensic readiness framework that can be deployed for purposes of ransomware investigation. The implementation of such a framework within a system can significantly produce near real-time potential evidence, in contrast to pieces of post-mortem evidence (after the incident has occurred, when potential evidence may have been deleted or encrypted). Moreover, compared to post-mortem forensics, the proposed framework can potentially generate more evidential information during a ransomware incident. Based on this framework, a model was designed for automated ransomware incident detection and response for ransomware forensics. The results presented in Chapter 7 confirmed the usefulness of such a framework.

From a forensic perspective, the Ransomware Readiness Framework presents a mechanism to minimise the cost of legal prosecution and offers protection against ransom payment. Furthermore, this framework also provides a mechanism to better understand how ransomware works and propagates on a granular level.

***“Ransomware is not only about weaponizing encryption, it’s more about bridging the fractures in the mind with a weaponized message that demands a response from the victim.”***

**James Scott**

# **PART VI**

## **APPENDICES**

# Appendix A

## Derived Publications and Conference Papers

The following is a list of all the publications and conference papers that were derived from this research.

- ✓ Singh, H. S. Venter, and A. R. Ikuesan, “Windows registry harnesser for incident response and digital forensic analysis,” *Australian Journal of Forensic Science*, vol. 00, no. 00, pp. 629–638, 2018.
- ✓ Singh, A. R. Ikuesan, and H. S. Venter, “Digital Forensic Readiness Framework for Ransomware Investigation,” in *Digital Forensics and Cyber Crime*, 2019, pp. 91–105.
- ✓ A. Singh, A. Ikuesan, and H. Venter, “A context-aware trigger mechanism for ransomware forensics,” *14th International Conference on Cyber Warfare Security, ICCWS 2019*, pp. 629–638, 2019.

There are two more papers that will be submitted from the content of this dissertation.

# Appendix B

## W2RC & W3RS installation guide 2019

### A. Setting up the environment

#### Requirements

- *Python 3.7* (`sudo apt-get install python3.7 python3.7-venv`)
- *Virtual Box (Windows VM)* (<https://www.virtualbox.org/wiki/Downloads>)
- *MongoDB* (<https://docs.mongodb.com/manual/installation/>)
- *Cuckoo Framework* (<https://cuckoosandbox.org/>) – Guided in Step 3

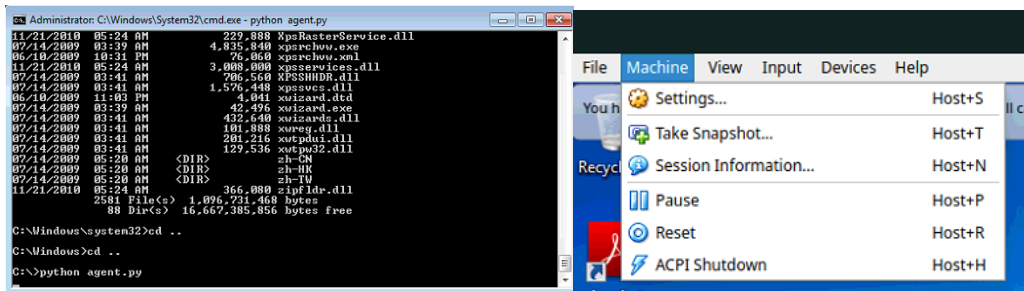
It is recommended that a Linux system should be used for the analysis server in order to ensure better performance and added security. Ensure all system packages are up to date prior to installation. It is not recommended to install the systems in a Virtual machine due to degraded performance and having virtualization within virtualization which is not recommended and has many limitations. Installing python is essential for the system to work as well as the above-mentioned packages. Following this guide will give you the basics on how to setup cuckoo and prepare the system for the Ransomware Readiness Framework (RRF). A detailed guide on how to setup cuckoo is available on their doc's page (<https://cuckoo.readthedocs.io/en/latest/installation/>). However, step 3 of this guide will help you install and configure the necessary requirements to get the system working. The next step will focus on creating a VM and configuring it.

### B. Setting up Virtual Box

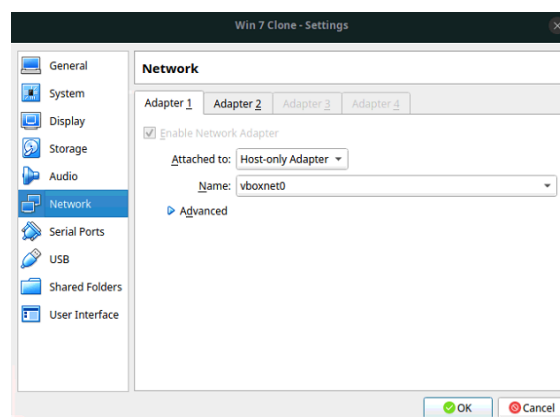
#### Create a Windows VM for automated sandbox analysis

In order to analyse executables, a Windows VM needs to be created. Other Virtual Box other platforms like VMWare, QEMU and KVM are supported. For the sake of simplicity and open-source nature of these prototypes, Virtual Box will be used. Due to Window Licensing, you will need to create this VM on your own using an ISO or preconfigured VM obtainable from Microsoft. The system specifications are up to you to decide. It is recommended to use at least 2GB of memory however with an increase in memory there is also an increase in the time it takes to analyse a sample. In order to allow the VM to communicate to the analysis engine, the agent needs to be run in the VM. Download and install python on the VM and ensure you install the pillow python package `$> pip install pillow` this allows the agent the ability to take screenshots of the VM which help an investigator to see what is happening in the background. After python is installed download the agent (<https://github.com/cuckoosandbox/cuckoo/blob/master/cuckoo/data/agent/agent.p>

y) and run it with using Administrator privileges with `$> python agent.py` no output will be shown once the agent is running.



Take a snapshot of the machine while the agent is running. This is important as this snapshot will be used for performing analysis. More information on how to correctly configure the agent is available here (<https://cuckoo.readthedocs.io/en/latest/installation/guest/agent/>). Ensure that your VM uses the Host-Only adapter and that the adapter is configured on the same subnet as the physical machine.

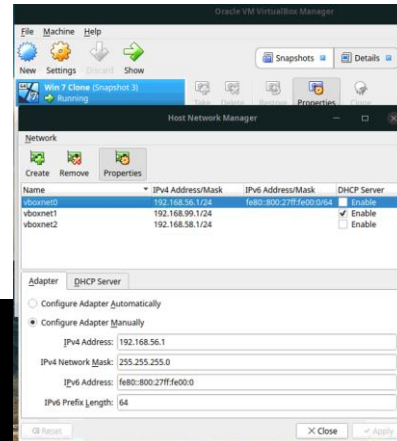


Setting the IP address of the host can be found on the setting of the VM under the network tab. Furthermore, setting the VirtualBox Host manager can be done from the main **Virtual Box Menu -> File -> Host Network Manager**. The screenshot below shows how to correctly set up the manager. Ensure that the gateway IP address matches that of cuckoo (note down the IP address so it can be added to the cuckoo configuration in step 3) and it matches vboxnet0 system, network manager. Please make sure that the DHCP Server is unchecked, this prevents Virtual Box from trying to assign an IP address using the NAT adapter from the system. This is because the cuckoo framework does not support DHCP.

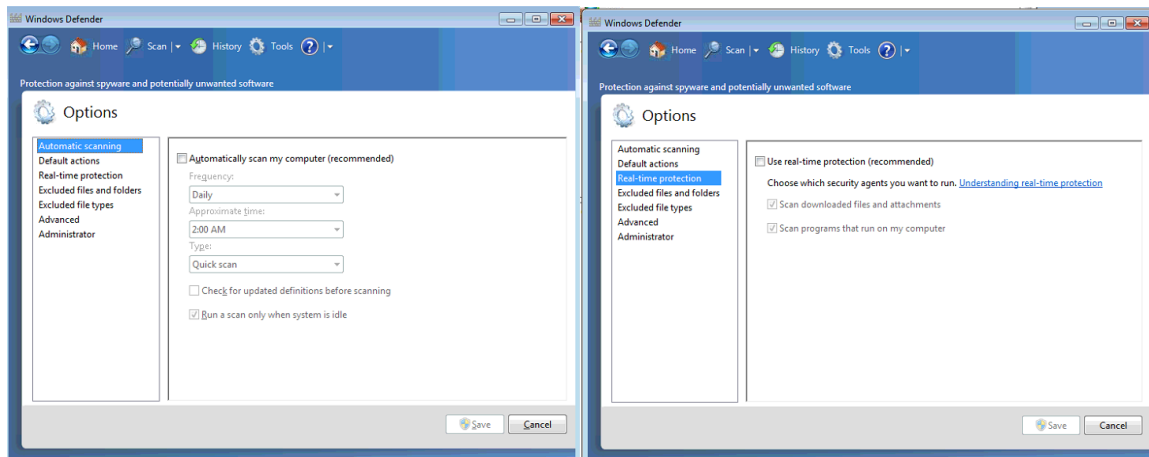
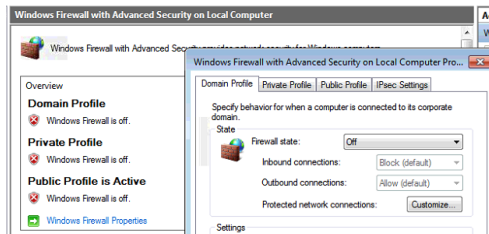
```

~/..c> conf -> ifconfig vboxnet0
vboxnet0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.56.1 netmask 255.255.255.0 broadcast 192.168.56.255
inet6 fe80::800:27ff:fe00:0 prefixlen 64 scopeid 0x20<link>
ether 0a:00:27:00:00:00 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1077 bytes 145385 (141.9 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

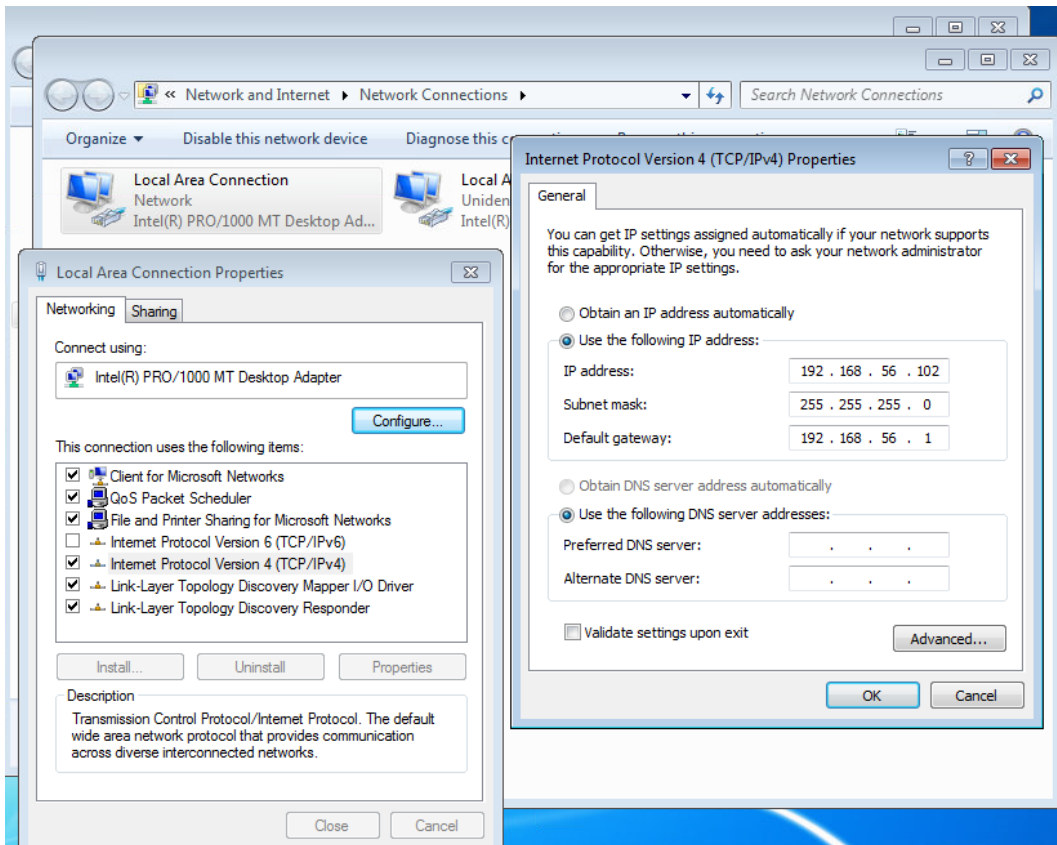
```



It is recommended that other general software is also installed and configured appropriately such as Adobe Reader, Microsoft Office, etc. Be sure to disable auto-updates of this software. Disable the firewall and anti-virus in the VM as well as any auto-updates.



This is vital otherwise this may hinder analysis and block the agent from communicating with the server. In order to correctly allow your VM to communicate with the cuckoo server, the network adapter inside the VM in Windows needs to change to be a static IP instead of DHCP as this is not supported by cuckoo and will break the entire analysis process. Navigate to **Control Panel\Network and Internet\Network Connections** right-click on the network adapter -> properties -> navigate to the IPv4 -> properties and change from "Obtain an IP address automatically" to "Use the following IP address" as indicated in the screenshot below. Assign the machine the IP address you want and note it down as this will be the same IP address you provide to the cuckoo configuration in step 3



### C. Setting up Cuckoo Framework

**i** Cuckoo framework is a bit tedious to setup but once everything is configured correctly it provides a powerful platform. Please ensure you follow these instructions carefully

It is recommended that you run cuckoo under its own user and therefore you must create a user called cuckoo. You can find the installation process for that here (<https://cuckoo.readthedocs.io/en/latest/installation/host/installation/>). Create a python virtual environment for cuckoo to run in it. This is to prevent harm to the system. It also ensures that cuckoo remains compatible even if system packages are updated. To sum it up run the following commands:

#### #Creating a cuckoo user

```
$ sudo adduser cuckoo
$ sudo usermod -a -G vboxusers cuckoo
$ sudo usermod -a -G libvirtid cuckoo
```

#### #Installing cuckoo in a venv

```
$ virtualenv venv # This creates a virtualized environment
$ source venv/bin/activate # This enters into python virtual environment
(venv)$ pip install -U pip setuptools
(venv)$ pip install -U cuckoo
(venv)$ cuckoo -d # This will now create cuckoo configuration files
```



After cuckoo configurations have been created they can be found in the CWD of cuckoo which is `~/cuckoo`

Open the file `~/cuckoo/config/cuckoo.conf` and ensure the information there is correct and matches the config below:

```
machinery = virtualbox
ip = 192.168.56.1 # IP address of Windows VM gateway (same as system adapter vboxnet0)
```

For larger processing of analysis, you can use your own database and can configure it in the [database] section, otherwise, a default SQLite DB will be created.

Open the file `~/cuckoo/config/auxiliary.conf` and ensure the information there is correct and matches the config below:

```
[sniffer]
enabled = yes # Ensure that tcpdump is installed
tcpdump = /usr/sbin/tcpdump # Path to tcpdump installation binary
```

This is to monitor network connections and any outgoing requests are logged for trackability as well as investigative processes.

Open the file `~/cuckoo/config/memory.conf` and ensure the information there is correct and matches the config below:

NB: ensure that volatility is installed (`(venv)$ pip install volatility`) or

<https://github.com/volatilityfoundation/volatility/wiki/Installation>

Ensure that all the plugins are enabled in this configuration as this analysis's memory information one of the vital parts for the framework and potential recovery from a ransomware attack. Set the "guest\_profile" variable to the profile that matches your Windows VM to prevent wasting time identifying the profile. A list of profiles can be found here (<https://github.com/volatilityfoundation/volatility/wiki/2.6-Win-Profiles>).

Open the file `~/cuckoo/config/processing.conf` and ensure the information there is correct and matches the config below:

```
[analysisinfo]
enabled = yes
[behavior]
enabled = yes
[buffer]
enabled = yes
[strings]
enabled = yes
[static]
enabled = yes
[procmemory]
enabled = yes
```

Open the file `~/.cuckoo/config/reporting.conf` and ensure the information there is correct and matches the config below:

```
[jsondump]
  enabled = yes
  indent = 4
  calls = yes
[mongodb]
  enabled = yes
  host = 127.0.0.1
  port = 27017
  db = cuckoo
  store_memdump = yes
  paginate = 100
  # MongoDB authentication (optional).
  username = root
  password = password
```

Ensure that a database is created in MongoDB and that the login details are correct.

Open the file `~/.cuckoo/config/virtualbox.conf` and ensure the information there is correct and matches the config below:

```
[virtualbox]
  mode = headless
  path = /usr/bin/VBoxManage      # Path to the local installation of the VBoxManage utility.
  interface = vboxnet0
  machines = cuckoo1
[cuckoo1]
  label = Win 7                   # Specify the label name of the current machine as specified
  in your
  platform = windows
  ip = 192.168.56.102            # Specify the IP address of the current virtual machine. Make
                                # sure that the IP address is valid and that the host machine is
                                # able to reach it. If not, the analysis will fail.
  snapshot = Snapshot 1         # Name of the snapshot taken in step 2
```

More details on how to configure other services and configurations can be found on the cuckoo config docs page

[\(https://cuckoo.readthedocs.io/en/latest/installation/host/configuration/\)](https://cuckoo.readthedocs.io/en/latest/installation/host/configuration/).

To ensure that everything is correctly setup within the (venv) run the cuckoo command “(venv)\$ cuckoo” you should see something similar to the screenshot below:

```

~/ .c/ conf cuckoo
sSSs .S S. sSSs .S S. sSSs_sSSs sSSs_sSSs
d%%SP .SS SS. d%%SP .SS SS. d%%SP-YS%%b d%%SP-YS%%b
d%S' S%S S%S d%S' S%S S&S d%S' S%b d%S' S%b
S%S S%S S%S S%S S%S d*S S%S S%S S%S S%S
S&S S&S S&S S&S S&S .S*S S&S S&S S&S S&S
S&S S&S S&S S&S S&S_sdSSS S&S S&S S&S S&S
S&S S&S S&S S&S S&S S&S `S% S&S S&S S&S S&S
S*b S*b d*S S*b S*S S% S*b d*S S*b d*S
S*S. S*S. .S*S S*S. S*S S& S*S. .S*S S*S. .S*S
SSSbs SSSbs_sdSSS SSSbs S*S S& SSSbs_sdSSS SSSbs_sdSSS
YSSP YSSP-YSSY YSSP S*S SS YSSP-YSSY YSSP-YSSY
SP
Y

Cuckoo Sandbox 2.0.6
www.cuckoosandbox.org
Copyright (c) 2010-2018

2019-07-17 23:11:51,977 [cuckoo.core.scheduler] INFO: Using "virtualbox" as machine manager
2019-07-17 23:11:52,910 [cuckoo.core.scheduler] INFO: Loaded 1 machine/s
2019-07-17 23:11:52,919 [cuckoo.core.scheduler] INFO: Waiting for analysis tasks.

```

The cuckoo server is now ready to receive analysis tasks. In order to make cuckoo accessible to other clients (W2RC), the cuckoo API needs to be executed in a different terminal tab with the command `(venv)$ cuckoo api -H 0.0.0.0 -p 8080`

```

- cuckoo api -H 0.0.0.0 -p 8080
2019-07-17 23:21:08,809 [werkzeug] INFO: * Running on http://0.0.0.0:8080/

```

**NB.** Note that it serves on 0.0.0.0 this means that is accessible by the systems IP assigned through the network. To test that it works check your IP from “ifconfig” and perform a curl request on the path “/cuckoo/status” like in the screenshot below:

```

- curl http://0.0.0.0:8080/cuckoo/status
{
  "cpu_count": 8,
  "cpuload": [
    0.19,
    0.31,
    0.26
  ],
  "diskspace": {
    "analyses": {
      "free": 31185977344,
      "total": 148774080512,
      "used": 117588103168
    },
    "binaries": {
      "free": 31185977344,
      "total": 148774080512,
      "used": 117588103168
    },
    "hostname": "ironman",
    "machines": {
      "available": 1,
      "total": 1
    },
    "memavail": 12083904,
    "memory": 26.162004440091636,
    "memtotal": 16365428,
    "processes": {
      "cuckoo": 32700
    },
    "tasks": {
      "completed": 0,
      "pending": 0,
      "reported": 178,
      "running": 0,
      "total": 178
    },
    "version": "2.0.6"
  }
}

```

This provides the status of the machine as well as some statistics if you get output similar to the screenshots above that means the API is running successfully. The next step is to set up a secure storage engine W3RS.

**D. Setting up Storage (W3RS)**

**i** Ensure that the relevant python packages are installed

If not done already clone or download the repository <https://github.com/AvinashSingh786/W3RS>. It is recommended to create a separate python virtual environment and run the W3RS server it in the directory `/srv/W3RS/venv` using the same commands listed in Step 3. You will then also have to install any python packages from the requirements.txt file. Since this application uses Django, it can also be configured using uwsgi. However, for simplicity, we will use Django's inbuilt server functionality. In the virtual environment run the following commands:

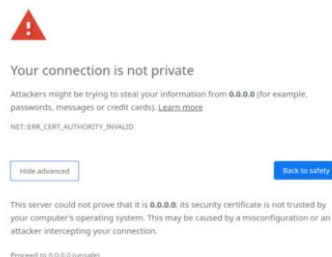
#### #Installing W3RS in a venv

```
$ virtualenv venv -p python3          # This creates a virtualized environment
$ source venv/bin/activate           # This enters into python virtual environment
(venv)$ pip install -r requirements.txt
(venv)$ python manage.py makemigrations # This sets up the storage engine and databases
(venv)$ python manage.py makemigrations pde # This sets up the storage engine and databases
(venv)$ python manage.py migrate      # This creates the databases and interfaces
(venv)$ python manage.py createsuperuser # Create a super user that you will use as the admin
(venv)$ python manage.py runsslserver 0.0.0.0:8082
```

The last command will run a secure SSL server on port 8000, note that this will use a self-signed certificate and will not be shown as safe in the browser. In order to prevent that you will need to get an SSL certificate from a Certificate Company. If you don't want the unsafe https to be displayed in the browser, you can obtain an SSL certificate from Let's Encrypt CA (<https://letsencrypt.org/>). You will get crt or pem files and you can just supply the path to these files with the following flags `--certificate` and `--key` when running the SSL server. Once the server is running to ensure that the setup has completed correctly navigate to the link provided from the command and check if you get a screenshot similar to the one below. Having an SSL server protects the traffic being transferred from clear text attacks and network sniffing. Navigate to (<https://0.0.0.0:8000>) got to advanced and proceed to site.

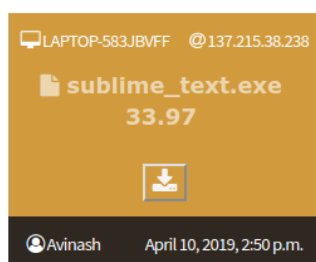
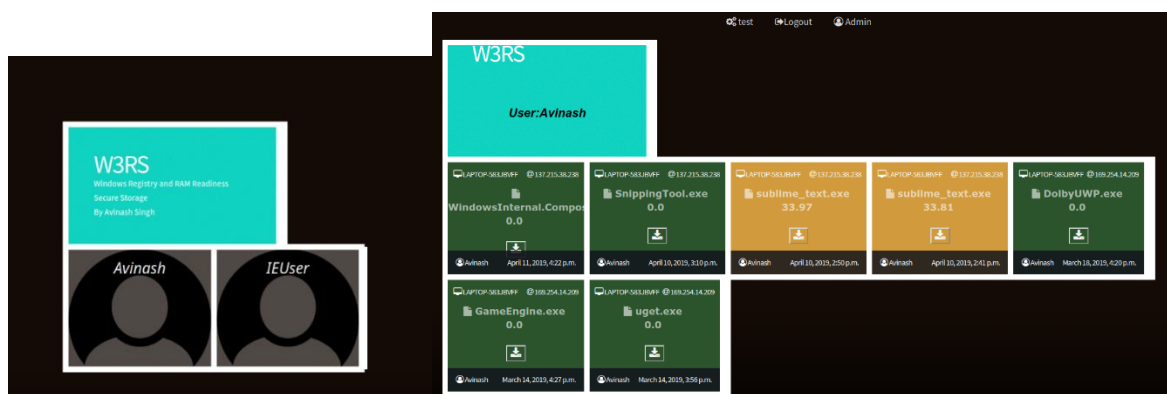
#### Problems:

- If you get the error of host is not allowed, please add the hostname to the [W3RS/settings.py](#) file at the ALLOWED\_HOSTS.
- If you get [TypeError: get\\_available\\_name\(\) got an unexpected keyword argument 'max\\_length'](#) please navigate to `/srv/W3RS/venv/lib/python3.6/site-packages/django/core/files/storage.py` in save, line 48 and remove the max\_length parameter.
- [Reverse for " not found. " is not a valid view function or pattern name.](#) navigate to `/srv/W3RS/venv/lib/python3.6/site-packages/django_encrypted_filefield/fields.py` and insert the following "return FETCH\_URL\_NAME" line 42.



In order to download PDE, you will need to enable 2-factor authentication. In order to do that you will need to be logged in. After you have logged in with the superuser

details you can choose your method of 2FA for simplicity, take the default option. Use the mobile app Authy or Google Authenticator to scan the QR code presented on the next screen and enter in the 6-digit token from the app and 2FA will be enabled. You can visit your profile page and create backup tokens however this is not advised. In order to enable email notifications, navigate to **W3RS -> settings.py** and replace the EMAIL details section with the relevant information. Once you are done setting up you will see a blank screen but once the W2RC is set up you will see something similar to the screenshot below and once the desired user is clicked the collection of analysed samples will be shown:



When analysing the information about the executable we see the machine name (“LAPTOP-533JBVFF”) and the IP address of the machine on the top right as well as the executable name in the center. The CAT value is shown in the middle with the download PDE button that will allow the JSON encoded PDE to be downloaded and further analysed. The user of the machine and the date and time appear at the bottom in case the machine has multiple users and the date and time it for reliability purposes.

After the storage engine has been setup the next step will be to setup the client on the user machines.

### E. Setting up Collection (W2RC)

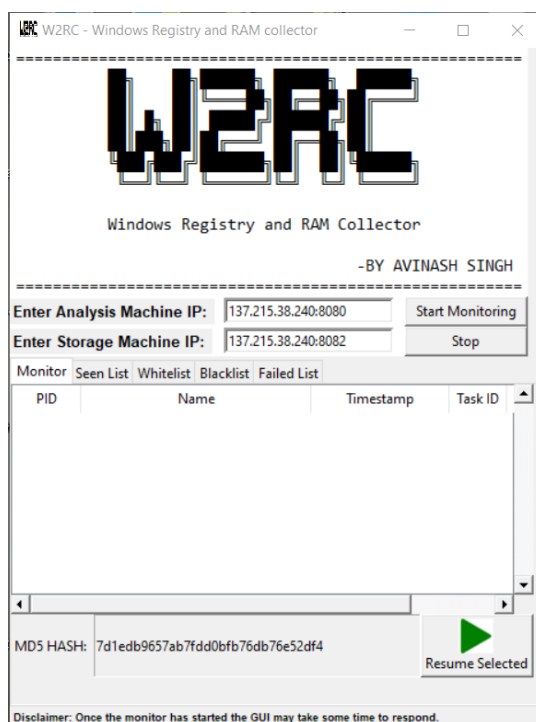
#### **i** Requirement:

- OpenSSL – this is used for the https connection to the storage server W3RS

*This tool is packaged in an MSI file to ease the installation process on the client-side and removes the need to install additional libraries, etc.*

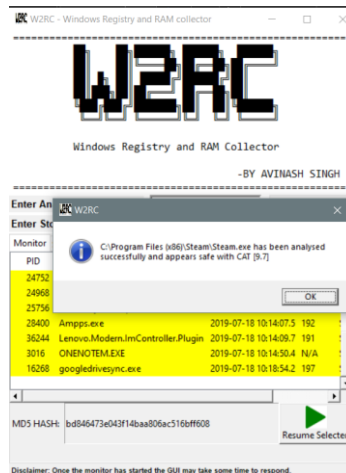
W2RC can be downloaded from the releases page (<https://github.com/AvinashSingh786/W2RC/releases>) and installed on the user pc that wishes to be monitored using this framework. Run the executable and follow the instructions. Once the tool is installed an icon will be displayed on the desktop, edit this shortcut to “Run as Administrator”. The tool needs to run as an

administrator because it will be collecting information and sending executables for analysis. Run the tool and you should get a GUI interface like the screenshot below:



In order to start the monitoring and collection tool W2RC, you will need to enter in the correct IP address or domain name of the server. The analysis machine IP is for the cuckoo API that was set up in step 3 and the storage machine IP is for the W3RS system. Ensure the ports are correct otherwise monitoring will not start and an error message will be shown. Logs are collected for the tool and can be found in the installation directory in program files. Once the tool detects that the 2 servers are online it will begin monitoring all processes on the system. Since this is just a prototype too there are some inefficiencies and bugs. For start, once the monitoring has started the GUI may become a bit unresponsive this is because of python not truly having concurrency but rather simulated. This, therefore, relies on the main thread to switch between child threads which takes some time due to the constant monitoring of the processes. In order to ensure the integrity of the databases for seen processes, whitelisted and blacklisted the MD5 sum of each database is shown at the bottom as well as in the logs further ensuring integrity when questioned. Once the monitor has started it will find processes that have not been seen before and send the executable to the analysis machine.

The tool relies on both static and dynamic analysis, therefore if the behaviour of the executable cannot be determined at this stage the tool will report that it failed to determine the behaviour of the executable and will perform calculations and analysis using static methods. Once a process has been analysed and the CAT value determined it will be alerted to the user as in the screenshot below:



All unseen processes will be quickly suspended, and the process will undergo analysis to determine if it is benign or malicious once the result comes back the process is resumed or killed.

THANK YOU ☺

---

**W2RC** – <https://github.com/AvinashSingh786/W2RC>

**W3RS** – <https://github.com/AvinashSingh786/W3RS>

Installation guide created by **Avinash Singh**

**Date:** 2019/07/18

**Contact:** [asingh@cs.up.ac.za](mailto:asingh@cs.up.ac.za)

# Appendix C

Since PDE for ransomware samples can be fairly long sample snippets are presented in this Appendix. For a full PDE sample please visit (<https://github.com/AvinashSingh786/W2RC/tree/master/sample>)

## A. Sample Snippets of PDE information

```
"markcount": 1282,
"families": [],
"description": "Drops 1282 unknown file mime types indicative of ransomware writing encrypted files back to disk",
"severity": 6,
"marks": [
  {
    "category": "file",
    "ioc": "C:\\Program Files\\Microsoft Office\\root\\Office16\\LogoImages\\WinWordLogo.scale-80.png.id-B42407E1.[bitpandacom@qq.com].combo",
    "type": "ioc",
    "description": null
  },
  {
    "category": "file",
    "ioc": "C:\\Program Files\\Microsoft Office\\root\\Templates\\1033\\Access\\Part\\1 Right.accdt.id-B42407E1.[bitpandacom@qq.com].combo",
    "type": "ioc",
    "description": null
  },
  {
    "category": "file",
    "ioc": "C:\\Program Files\\Microsoft Office\\root\\Licenses16\\VisioStdC0365R_SubTest-ppd.xrm-ms.id-B42407E1.[bitpandacom@qq.com].combo",
    "type": "ioc",
    "description": null
  },
  {
    "category": "file",
    "ioc": "C:\\Program Files\\Microsoft Office\\root\\Office16\\ADDINS\\Microsoft Power Query for Excel Integrated\\bin\\zh-HANS\\Microsoft.Mashup.OAuth.resources.dll.id-B42407E1.[bitpandacom@qq.com].combo",
    "type": "ioc",
    "description": null
  },
  {
    "category": "file",
    "ioc": "C:\\Program Files\\Microsoft Office\\root\\Licenses16\\ProPlusR_Retail-pl.xrm-ms.id-B42407E1.[bitpandacom@qq.com].combo",
    "type": "ioc",
    "description": null
  }
],
}
```

Figure 0-1. PDE snippet showing detected signature

```
{
  "markcount": 1,
  "families": [],
  "description": "A process attempted to delay the analysis task.",
  "severity": 2,
  "marks": [
    {
      "type": "generic",
      "description": "dharma.exe tried to sleep 378 seconds, actually delayed analysis time by 378 seconds"
    }
  ],
  "references": [],
  "name": "antisandbox_sleep"
},
{
  "markcount": 1,
  "families": [],
  "description": "Creates a suspicious process",
  "severity": 2,
  "marks": [
    {
      "category": "cmdline",
      "ioc": "C:\\Windows\\System32\\cmd.exe",
      "type": "ioc",
      "description": null
    }
  ],
  "references": [],
  "name": "suspicious_process"
},
{
  "markcount": 58,
  "families": [],
  "description": "Searches running processes potentially to identify processes for sandbox evasion, code injection or memory dumping",
  "severity": 2,
  "marks": [
    {
      "call": {

```

Figure 0-2. PDE snippet showing delay operations



```

"pe_sections": [
  {
    "size_of_data": "0x00009e00",
    "virtual_address": "0x00001000",
    "entropy": 5.965306229244429,
    "name": ".text",
    "virtual_size": "0x00009c25"
  },
  {
    "size_of_data": "0x00002800",
    "virtual_address": "0x0000b000",
    "entropy": 7.78504061338529,
    "name": ".rdata",
    "virtual_size": "0x00002636"
  },
  {
    "size_of_data": "0x0000a800",
    "virtual_address": "0x0000e000",
    "entropy": 7.9813944978194655,
    "name": ".data",
    "virtual_size": "0x0000aad5"
  }
]

```

**Figure 0-3.** PDE snippet showing PE sections and entropy

```

"procmemory": [
  {
    "regions": [
      {
        "protect": "rw",
        "end": "0x00020000",
        "addr": "0x00010000",
        "state": 4096,
        "offset": 24,
        "type": 262144,
        "size": 65536
      },
      {
        "protect": "rw",
        "end": "0x00030000",
        "addr": "0x00020000",
        "state": 4096,
        "offset": 65584,
        "type": 262144,
        "size": 65536
      },
      {
        "protect": "r",
        "end": "0x00036000",
        "addr": "0x00030000",
        "state": 4096,
        "offset": 131144,
        "type": 262144,
        "size": 24576
      }
    ]
  }
]

```

**Figure 0-4.** PDE snippet showing process memory

```

"buffer": [
  {
    "yara": [],
    "sha1": "e8ddc8fe39290a424e967aea62a7f3d0822da27e",
    "name": "e8ddc8fe39290a424e967aea62a7f3d0822da27e",
    "type": "Python script, ASCII text executable",
    "sha256": "9ab11c145f32b75db58e08fb8a183095572bcc22f800f0c1550f5dc397501e4a",
    "urls": [],
    "crc32": "02893E92",
    "path": "/home/avinash/.cuckoo/storage/analyses/206/buffer/e8ddc8fe39290a424e967aea62a7f3d0822da27e",
    "ssdeep": null,
    "size": 18043,
    "sha512": "dd41842d124190f50d00983e4d4b17dad182745b8772fde07df4f387e60a7b1e3b46fe8e9d19256cea586a672021bb776e41b4a1c59a530f433527c2dd13cc1f",
    "md5": "868f6d79a3de7ac63c28e1ec4d3743b8"
  },
  {
    "yara": [],
    "sha1": "23678df9986bfe11334elb41d4f1924395e6f76f",
    "name": "23678df9986bfe11334elb41d4f1924395e6f76f",
    "type": "data",
    "sha256": "f3c19fac909bf0687fc953671ac0dcf59f6392a1e66dd132bc94609df45b998c0",
    "urls": [],
    "crc32": "5E2D8BC0",
    "path": "/home/avinash/.cuckoo/storage/analyses/206/buffer/23678df9986bfe11334elb41d4f1924395e6f76f",
    "ssdeep": null,
    "size": 4941,
    "sha512": "6d8798affa28248c1a65cca976d81d6bb30e09824e1f0ab41b58b744dd2c525cf6e90f9269275cle53dfb12904d0f95475d30f8e5160b36304ac72cf6cc78db",
    "md5": "99312558b5385d7ed5d159f7bdf1a0"
  }
],

```

Figure 0-5. PDE snippet showing buffer information location

```

"network": {
  "tls": [],
  "udp": [
    {
      "src": "192.168.56.1",
      "dst": "192.168.56.101",
      "offset": 1242,
      "time": 45.145857095718384,
      "dport": 137,
      "sport": 137
    },
    {
      "src": "192.168.56.1",
      "dst": "192.168.56.101",
      "offset": 1482,
      "time": 42.39110708236694,
      "dport": 138,
      "sport": 138
    },
    {
      "src": "192.168.56.101",
      "dst": "178.33.158.0",
      "offset": 1962,
      "time": 9.085948944091797,
      "dport": 6893,
      "sport": 59965
    },
    {
      "src": "192.168.56.101",
      "dst": "178.33.158.0",
      "offset": 2045,
      "time": 25.95373511314392,
      "dport": 6893,
      "sport": 64716
    }
  ],

```

Figure 0-6. PDE snippet showing network activity

```

"dll_loaded": [
  "gdiplus.dll",
  "urlmon.dll",
  "gdi32.dll",
  "kernel32.dll",
  "oleaut32.dll",
  "netapi32.dll",
  "dwmapi.dll",
  "ntdll.dll",
  "shlwapi.dll",
  "mpr.dll",
  "PROPSYS.dll",
  "crypt32.dll",
  "API-MS-Win-Core-LocalRegistry-L1-1-0.dll",
  "NETUTILS",
  "advapi32.dll",
  "ole32.dll",
  "CRYPTSP.dll",
  "API-MS-Win-Security-SDDL-L1-1-0.dll",
  "version.dll",
  "ADVAPI32.dll",
  "OLEAUT32.dll",
  "profapi.dll",
  "SAMCLI",
  "SHELL32.dll",
  "comctl32.dll",
  "ws2_32.dll",
  "NTDLL",
  "powerprof.dll",
  "shell32.dll",
  "SETUPAPI.dll",
  "user32.dll"
],

```

**Figure 0-7.** PDE snippet showing loaded DLLs

```

{
  "call": {
    "category": "crypto",
    "status": 1,
    "stacktrace": [],
    "api": "CryptExportKey",
    "return_value": 1,
    "arguments": {
      "crypto_handle": "0x008179d0",
      "crypto_export_handle": "0x00000000",
      "buffer": "\u0006\u0002\u0000\u0000\u0000\u0000\u00a4\u0000\u0000RSA1\u0000\b\u0000\u0000\u0001\u0000\u0001\u0000\u0000",
      "blob_type": 6,
      "flags": 0
    },
    "time": 1573637827.016,
    "tid": 2216,
    "flags": {}
  },
  "pid": 1396,
  "type": "call",
  "cid": 921
},

```

**Figure 0-8.** PDE snippet showing cryptographic key information

```

"calls": [
  {
    "category": "registry",
    "status": 1,
    "stacktrace": [],
    "api": "RegOpenKeyExW",
    "return_value": 0,
    "arguments": {
      "access": "0x00020019",
      "base_handle": "0x80000000",
      "key_handle": "0x00000086",
      "regkey": "HKEY_CLASSES_ROOT\\interface\\{3050f55f-98b5-11cf-bb82-00aa00bdce0b}",
      "regkey_r": "interface\\{3050f55f-98b5-11cf-bb82-00aa00bdce0b}",
      "options": 0
    },
    "time": 1573638297.250375,
    "tid": 2148,
    "flags": {}
  },
  {
    "category": "resource",
    "status": 0,
    "stacktrace": [],
    "last_error": 1814,
    "nt_status": -1073741685,
    "api": "FindResourceExW",
    "return_value": 0,
    "arguments": {
      "name": "#12",
      "type": "#5122",
      "module_handle": "0x76e10000",
      "language_identifier": 0
    },
    "time": 1573638297.250375,
    "tid": 2148,
    "flags": {}
  },
  {
    "category": "resource",
    "status": 0,
    "stacktrace": [],
    "last_error": 1813,
    "nt_status": -1073741686,
    "api": "FindResourceExW",
    "return_value": 0,
    "arguments": {

```

Figure 0-9. PDE snippet showing API calls

```

"signatures": [
  {
    "markcount": 1,
    "families": [],
    "description": "Queries for the computername",
    "severity": 1,
    "marks": [
      {
        "call": {
          "category": "misc",
          "status": 1,
          "stacktrace": [],
          "api": "GetComputerNameW",
          "return_value": 1,
          "arguments": {
            "computer_name": "USER-PC"
          },
          "time": 1573637822.328,
          "tid": 2216,
          "flags": {}
        },
        "pid": 1396,
        "type": "call",
        "cid": 0
      }
    ],
    "references": [],
    "name": "antivm_queries_computername"
  },
  {
    "markcount": 1,
    "families": [],
    "description": "Command line console output was observed",
    "severity": 1,
    "marks": [
      {
        "call": {
          "category": "misc",
          "status": 1,
          "stacktrace": [],
          "api": "WriteConsoleW",
          "return_value": 1,
          "arguments": {
            "buffer": "Successfully processed 515 files; Failed processing 0 files\r\n",
            "console_handle": "0x00000007"
          },
          "time": 1573637847.96625,
          "tid": 3144,
          "flags": {}
        },
        "pid": 3140,
        "type": "call",
        "cid": 13495
      }
    ],
    "references": [],
    "name": "console_output"
  },
  {
    "markcount": 4,

```

**Figure 0-10.** PDE snippet showing various signatures

# Bibliography

- [1] L. A. Gordon, M. P. Loeb, and W. Lucyshyn, "Sharing information on computer systems security: An economic analysis," *Journal of Accounting and Public Policy*, vol. 22, no. 6, pp. 461–485, 2003.
- [2] M. F. Vilardo, "Online impersonation in securities scams," *IEEE Security & Privacy Magazine*, vol. 2, no. 3, pp. 82–85, May 2004.
- [3] A. Idika, Nwokedi and Mathur and N. Idika, "A Survey of Malware Detection Techniques," *Purdue University*, pp. 1–48, 2007.
- [4] J. P. Tailor and A. D. Patel, "A Comprehensive Survey: Ransomware Attacks Prevention, Monitoring and Damage Control," *International Journal of Research and Scientific Innovation*, vol. IV, pp. 2321–2705, 2017.
- [5] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," *Journal of Computer and System Sciences*, vol. 80, no. 5, pp. 973–993, 2014.
- [6] W. Han, J. Xue, Y. Wang, Z. Liu, and Z. Kong, "Mallnsight: A systematic profiling based malware detection framework," *Journal of Network and Computer Applications*, vol. 125, pp. 236–250, Jan. 2019.
- [7] S. J. Vaughan-Nichols, "Today's most popular operating systems," 2017. [Online]. Available: <http://www.zdnet.com/article/todays-most-popular-operating-systems/>. [Accessed: 12-Apr-2018].
- [8] K. Cabaj *et al.*, "Experimental Analysis of Ransomware on Windows and Android Platforms: Evolution and Characterization," *IEEE Network*, vol. 94, no. 6, pp. 465–472, 2016.
- [9] S. Jung and Y. Won, "Ransomware detection method based on context-aware entropy analysis," *Soft Computing*, 2018.
- [10] R. A. Ikuesan, "Online psychographic model for insider identification," *Universiti Teknologi Malaysia*, 2015.
- [11] S. Logen, H. Höfken, and M. Schuba, "Simplifying RAM forensics: A GUI and extensions for the volatility framework," *Proceedings - 2012 7th International Conference on Availability, Reliability and Security, ARES 2012*, pp. 620–624, 2012.
- [12] C. Hargreaves and H. Chivers, "Recovery of encryption keys from memory using a linear scan," *ARES 2008 - 3rd International Conference on Availability, Security, and Reliability, Proceedings*, no. March, pp. 1369–1376, 2008.
- [13] Statista, "Global market share held by the leading mobile operating systems from 2010 to 2015," 2015. [Online]. Available: <https://www.statista.com/statistics/218089/global-market-share-of-windows-7/>. [Accessed: 12-Apr-2018].
- [14] P. Faruki, V. Laxmi, M. S. Gaur, and P. Vinod, "Behavioural detection with API call-grams to identify malicious PE files," *ACM International Conference Proceeding Series*, no. August, pp. 85–91, 2012.
- [15] A. Zimba, Z. Wang, and H. Chen, "Multi-stage crypto ransomware attacks: A new emerging cyber threat to critical infrastructure and industrial control systems," *ICT Express*, vol. 4, no. 1, pp. 14–18, 2018.
- [16] V. Kotov and mantej S. Rajpal, "Understanding Crypto-Ransomware," *Bromium.com*, pp. 1–35, 2014.
- [17] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: A survey and research

- directions,” *Computers and Security*, vol. 74, pp. 144–166, 2018.
- [18] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, “Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection,” 2016.
- [19] K. Hausknecht, D. Foit, and J. Buric, “RAM data significance in digital forensics,” in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 1372–1375.
- [20] Monika, P. Zavorsky, and D. Lindskog, “Experimental Analysis of Ransomware on Windows and Android Platforms: Evolution and Characterization,” *Procedia Computer Science*, vol. 94, pp. 465–472, 2016.
- [21] D. Morato, E. Berrueta, E. Magaña, and M. Izal, “Ransomware early detection by the analysis of file sharing traffic,” *Journal of Network and Computer Applications*, vol. 124, pp. 14–32, 2018.
- [22] A. Singh, A. R. Ikuesan, and H. S. Venter, “Digital Forensic Readiness Framework for Ransomware Investigation,” in *Digital Forensics and Cyber Crime*, vol. 259, F. Breitingger and I. Baggili, Eds. Cham: Springer International Publishing, 2019, pp. 91–105.
- [23] A. Singh, H. S. Venter, and A. R. Ikuesan, “Windows registry harnesser for incident response and digital forensic analysis,” *Australian Journal of Forensic Sciences*, vol. 00, no. 00, pp. 1–17, Dec. 2018.
- [24] A. Singh, A. Ikuesan, and H. Venter, “A context-aware trigger mechanism for ransomware forensics,” *14th International Conference on Cyber Warfare and Security, ICCWS 2019*, pp. 629–638, 2019.
- [25] V. R. Kebande and H. S. Venter, “A Cloud Forensic Readiness Model Using a Botnet as a Service,” in *The International Conference on Digital Security and Forensics (DigitalSec2014)*, 2014, pp. 23–32.
- [26] P. M. Trenwith and H. S. Venter, “Digital forensic readiness in the cloud,” in *2013 Information Security for South Africa*, 2013, pp. 1–5.
- [27] A. Adamov and A. Carlsson, “The state of ransomware. Trends and mitigation techniques,” in *Proceedings of 2017 IEEE East-West Design and Test Symposium, EWDTs 2017*, 2017.
- [28] K. Savage, P. Coogan, and H. Lau, “Information Resources,” *Research-Technology Management*, vol. 54, no. 5, pp. 59–63, 2011.
- [29] S. Mansfield-Devine, “Ransomware: the most popular form of attack,” *Computer Fraud and Security*, vol. 2017, no. 10, pp. 15–20, 2017.
- [30] ISO 27043, “INTERNATIONAL STANDARD ISO / IEC 27043: Information technology — Security techniques — Incident investigation principles and processes,” vol. 2015, 2015.
- [31] A. R. Ikuesan and H. S. Venter, “Digital behavioral-fingerprint for user attribution in digital forensics: Are we there yet?,” *Digital Investigation*, 2019.
- [32] E. Casey, *Digital Evidence and Computer Crime*, 3rd ed. Academic Press, 2011.
- [33] S. L. Garfinkel, *Digital Forensics.*, vol. 101, no. 5. American Scientist, 2013.
- [34] DFRWS, “A Road Map for Digital Forensic Research,” 2001.
- [35] A. R. Ikuesan and H. S. Venter, “Digital forensic readiness framework based on behavioral-biometrics for user attribution,” in *2017 IEEE Conference on Application, Information and Network Security (AINS)*, 2017, no. 1, pp. 54–59.
- [36] R. Von Solms and J. Van Niekerk, “From information security to cyber security,” *Computers and Security*, 2013.

- [37] A. Shabtai, Y. Elovici, and L. Rokach, "Introduction to Information Security," in *SpringerBriefs in Computer Science*, 2012.
- [38] A. Refsdal, B. Solhaug, and K. Stølen, "Cybersecurity," in *SpringerBriefs in Computer Science*, 2015.
- [39] G. J. Simmons, "Symmetric and Asymmetric Encryption," *ACM Computing Surveys (CSUR)*, vol. 11, no. 4, pp. 305–330, 1979.
- [40] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," *Journal of Cryptology*, 2013.
- [41] Microsoft, "Description of Symmetric and Asymmetric Encryption," Microsoft, 2018. [Online]. Available: <https://support.microsoft.com/en-us/help/246071/description-of-symmetric-and-asymmetric-encryption>. [Accessed: 12-Aug-2019].
- [42] A. Valjarevic and H. S. Venter, "Towards a Digital Forensic Readiness Framework for Public Key Infrastructure systems," in *2011 Information Security for South Africa*, 2011, pp. 1–10.
- [43] R. Bhanot and R. Hans, "A review and comparative analysis of various encryption algorithms," *International Journal of Security and its Applications*, 2015.
- [44] K. K. Sindhu and B. B. Meshram, "Digital Forensics and Cyber Crime Datamining," *Journal of Information Security*, vol. 03, no. 03, pp. 196–201, 2012.
- [45] B. Kaplan, "RAM is Key: Extracting Disk Encryption Keys From Volatile Memory," Carnegie Mellon University, 2007.
- [46] J. A. Lewis, D. E. Zheng, and W. A. Carter, *The Effect of Encryption on Lawful Access to Communications and Data*, vol. 2. 2017.
- [47] L. Chi and X. Zhu, "Hashing Techniques: A Survey and Taxonomy," *ACM Computing Surveys (CSUR)*, 2017.
- [48] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in *High Performance Mass Storage and Parallel I/O: Technologies and Applications*, 2001, pp. 3–14.
- [49] I. Agrafiotis, "Biggest-ever DDoS attack takes down high-profile web services," *Computer Fraud & Security*, vol. 2016, no. 11, pp. 1–3, 2016.
- [50] J. L. Hernandez-Ardieta, A. I. Gonzalez-Tablas, J. M. De Fuentes, and B. Ramos, "A taxonomy and survey of attacks on digital signatures," *Computers and Security*, 2013.
- [51] J. Zhou and D. Gollmann, "Evidence and non-repudiation," *Journal of Network and Computer Applications*, 1997.
- [52] B. Schneier, "Two-factor authentication: Too little, too late," *Communications of the ACM*, vol. 48, no. 4, p. 136, 2005.
- [53] P. Stephenson, "A comprehensive approach to digital incident investigation," *Information Security Technical Report*. 2003.
- [54] M. Kohn, M. S. Olivier, and J. H. P. Eloff, "Framework for a Digital Forensic Investigation.," *Communications*, 2006.
- [55] S. R. Selamat, R. Yusof, and S. Sahib, "Mapping Process of Digital Forensic Investigation Framework," *Journal of Computer Science*, vol. 8, no. 10, pp. 163–169, 2008.
- [56] E. Casey, *Handbook of Digital Forensics and Investigation*. Elsevier, 2010.
- [57] B. Carrier and E. H. E. H. Spafford, "An event-based digital forensic investigation framework," *Proceedings of the 4th Digital Forensic Research Workshop (DFRWS)*, 2004.



- [58] E. Casey, "Error, Uncertainty, and Loss in Digital Evidence," *International Journal of Digital Evidence*, 2002.
- [59] I. Xie and K. K. Matusiak, "Digital preservation," in *Discover Digital Libraries*, Elsevier, 2016, pp. 255–279.
- [60] J. Tan, "Forensic readiness," *Cambridge*, pp. 1–23, 2001.
- [61] Robert Rowlingson Ph, "A Ten Step Process for Forensic Readiness," *International Journal*, vol. 2, no. 3, pp. 1–28, 2004.
- [62] J. M. Ehrenfeld, "WannaCry, Cybersecurity and Health Information Technology: A Time to Act," *Journal of Medical Systems*, vol. 41, no. 7, p. 104, 2017.
- [63] M. Wojnowicz, G. Chisholm, B. Wallace, M. Wolff, X. Zhao, and J. Luan, "SUSPEND: Determining software suspiciousness by non-stationary time series modeling of entropy signals," *Expert Systems with Applications*, vol. 71, pp. 301–318, 2017.
- [64] D. O'Brien, Symantec, and D. O'Brien, "2017 Internet Security Threat Report," *Symantec*, vol. 20, no. April, p. 35, 2017.
- [65] Microsoft, "Microsoft Advanced Threat Analytics," 2015. [Online]. Available: <https://www.microsoft.com/en-us/cloud-platform/advanced-threat-analytics>. [Accessed: 27-Apr-2018].
- [66] Matt Mansfield, "Cyber Security Statistics," 2017. [Online]. Available: <https://smallbiztrends.com/2017/01/cyber-security-statistics-small-business.html>. [Accessed: 23-Apr-2018].
- [67] Av-test, "Malware," 2018. [Online]. Available: <https://www.av-test.org/en/statistics/malware/>. [Accessed: 25-Apr-2018].
- [68] M. I. Sobol, "Anti-Virus software," *Information Systems Security*, vol. 3, no. 2, pp. 24–29, 1994.
- [69] Q. Chen and R. A. Bridges, "Automated behavioral analysis of malware: A case study of wannacry ransomware," *Proceedings - 16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017*, vol. 2018-Janua, no. July, pp. 454–460, 2018.
- [70] Y. Ding, X. Xia, S. Chen, and Y. Li, "A malware detection method based on family behavior graph," *Computers and Security*, vol. 73, pp. 73–86, 2018.
- [71] A. Cohen, N. Nissim, L. Rokach, and Y. Elovici, "SFEM: Structural feature extraction methodology for the detection of malicious office documents using machine learning methods," *Expert Systems with Applications*, vol. 63, pp. 324–343, 2016.
- [72] J. A. Gómez-Hernández, L. Álvarez-González, and P. García-Teodoro, "R-Locker: Thwarting ransomware action through a honeyfile-based approach," *Computers and Security*, vol. 73, pp. 389–398, 2018.
- [73] M. Baykara and B. Sekin, "A novel approach to ransomware: Designing a safe zone system," *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pp. 1–5, 2018.
- [74] M. Damshenas, A. Dehghantanha, and R. Mahmoud, "A Survey on Malware propagation, analysis and detection," *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, vol. 2, no. 4, pp. 10–29, 2013.
- [75] M. Bishop, "Worm targets children," *Computer Fraud & Security*, vol. 2000, no. 10, pp. 4–5, Oct. 2000.
- [76] P. Denning, "The Internet Worm," in *Computers Under Attack: Intruders, Worms and Viruses*, 1990, pp. 193–200.
- [77] H. Orman, "The Morris worm: A fifteen-year perspective," *IEEE Security and*

- Privacy*. 2003.
- [78] E. Gandotra, D. Bansal, and S. Sofat, "Malware Analysis and Classification: A Survey," *Journal of Information Security*, vol. 05, no. 02, pp. 56–64, 2014.
- [79] kaspersky, "A Brief History of Computer Viruses & What the Future Holds." [Online]. Available: <https://www.kaspersky.co.za/resource-center/threats/a-brief-history-of-computer-viruses-and-what-the-future-holds>. [Accessed: 10-May-2019].
- [80] kaspersky, "Pegasus: The ultimate spyware for iOS and Android," 2017. [Online]. Available: <https://www.kaspersky.co.za/blog/pegasus-spyware/14604/>. [Accessed: 11-May-2019].
- [81] M. Dimitrova, "CVE-2019-3568 in WhatsApp Exploited Using Pegasus Spyware," 2019. [Online]. Available: <https://sensorstechforum.com/cve-2019-3568-whatsapp-pegasus/>. [Accessed: 18-May-2019].
- [82] Malwarebytes, "Adware." [Online]. Available: <https://www.malwarebytes.com/adware/>. [Accessed: 12-May-2019].
- [83] B. Stone-Gross, M. Cova, B. Gilbert, R. Kemmerer, C. Kruegel, and G. Vigna, "Analysis of a botnet takeover," *IEEE Security and Privacy*, vol. 9, no. 1, pp. 64–72, Jan. 2011.
- [84] Cloudflare, "What is the Mirai Botnet?" [Online]. Available: <https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/>. [Accessed: 14-May-2019].
- [85] Malwarebytes, "Rootkit." [Online]. Available: <https://blog.malwarebytes.com/detections/rootkit/>. [Accessed: 17-May-2019].
- [86] F. Li *et al.*, "Remedying Web Hijacking," in *Proceedings of the 25th International Conference on World Wide Web - WWW '16*, 2016, pp. 1009–1019.
- [87] N. Nikiforakis, W. Meert, Y. Younan, M. Johns, and W. Joosen, "SessionShield: Lightweight protection against session hijacking," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011.
- [88] D. O'Brien, "Internet Security Threat Report - Ransomware 2017," *Symantec*, p. 35, 2017.
- [89] B. Stone-Gross, M. Cova, B. Gilbert, R. Kemmerer, C. Kruegel, and G. Vigna, "Analysis of a Botnet Takeover," *IEEE Security Privacy*, vol. 9, no. 1, pp. 64–72, Jan. 2011.
- [90] S. Yu, G. Gu, A. Barnawi, S. Guo, and I. Stojmenovic, "Malware propagation in large-scale networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 1, pp. 170–179, 2015.
- [91] United States Government, "How to Protecting Your Networks from Ransomware," pp. 2–8, 2016.
- [92] E. H. Spafford, "The internet worm incident," in *ESEC '89*, 1989, pp. 446–468.
- [93] B. Rad, M. Masrom, and S. Ibrahim, "Camouflage in Malware: from Encryption to Metamorphism," *International Journal of Computer Science and Network Security*, vol. 12, no. 8, pp. 74–83, 2012.
- [94] P. Okane, S. Sezer, and K. McLaughlin, "Obfuscation: The hidden malware," *IEEE Security and Privacy*, vol. 9, no. 5, pp. 41–47, 2011.
- [95] B. Bashari Rad, M. Masrom, and S. Ibrahim, "Camouflage In Malware: From Encryption To Metamorphism," *International Journal Of Computer Science And Network Security (IJCSNS)*, vol. 12, no. 8, pp. 74–83, 2012.
- [96] S. Campbell, S. Chan, and J. R. Lee, "Detection of fast flux service networks,"

- Conferences in Research and Practice in Information Technology Series*, vol. 116, pp. 57–66, 2011.
- [97] E. Eilam, *Reversing: secrets of reverse engineering*. Wiley, 2013.
- [98] Q. Le, O. Boydell, B. Mac, and M. Scanlon, “Deep Learning at the Shallow End: Malware Classification for Non-Domain Experts,” *Digital Investigation*, 2018.
- [99] A. Gazet, “Comparative analysis of various ransomware virii,” *Journal in Computer Virology*, vol. 6, no. 1, pp. 77–90, 2010.
- [100] N. Hampton, Z. Baig, and S. Zeadally, “Ransomware behavioural analysis on windows platforms,” *Journal of Information Security and Applications*, vol. 40, pp. 44–51, 2018.
- [101] A. Pektaş and T. Acarman, “Classification of malware families based on runtime behaviors,” *Journal of Information Security and Applications*, vol. 37, pp. 91–100, 2017.
- [102] “VirusTotal,” 2019. [Online]. Available: <https://www.virustotal.com/>. [Accessed: 20-Jun-2019].
- [103] “Hybrid-Analysis,” 2019. [Online]. Available: <https://www.hybrid-analysis.com/>. [Accessed: 20-Jun-2019].
- [104] C. Guarnieri, “Cuckoo Sandbox,” 2014. [Online]. Available: <https://cuckoosandbox.org/>. [Accessed: 04-Oct-2018].
- [105] A. Patcha and J. M. Park, “An overview of anomaly detection techniques: Existing solutions and latest technological trends,” *Computer Networks*, 2007.
- [106] S. Sibi Chakkaravarthy, D. Sangeetha, and V. Vaidehi, “A Survey on malware analysis and mitigation techniques,” *Computer Science Review*, vol. 32, pp. 1–23, 2019.
- [107] J. Bergeron *et al.*, “Static Detection of Malicious Code in Executable Programs,” *Int. J. of Req. Eng*, pp. 184–189, 2001.
- [108] A. Cohen and N. Nissim, “Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory,” *Expert Systems with Applications*, vol. 102, pp. 158–178, 2018.
- [109] R. Kaur and M. Singh, “A survey on zero-day polymorphic worm detection techniques,” *IEEE Communications Surveys and Tutorials*, vol. 16, no. 3, pp. 1520–1549, 2014.
- [110] M. Mimoso, “Leaked NSA Exploit Spreading Ransomware Worldwide,” *Threatpost*, 2017. .
- [111] L. H. Newman, “How Leaked NSA Spy Tool ‘EternalBlue’ Became a Hacker Favorite | WIRED,” *Wired*, 2018. .
- [112] D. E. Denning and W. E. Baugh, “Hiding Crimes in Cyberspace,” *Information, Communication & Society*, vol. 2, no. 3, pp. 251–276, 1999.
- [113] W. Cui, J. Kannan, and H. J. Wang, “Discoverer: Automatic Protocol Reverse Engineering from Network Traces,” *USENIX Security*, no. 2, pp. 199–212, 2007.
- [114] K. Cabaj, M. Gregorczyk, and W. Mazurczyk, “Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics,” *Computers and Electrical Engineering*, vol. 66, pp. 353–368, 2017.
- [115] E. Gandotra, D. Bansal, and S. Sofat, “Malware Threat Assessment Using Fuzzy Logic Paradigm,” *Cybernetics and Systems*, vol. 48, no. 1, pp. 29–48, 2017.
- [116] J. Urrea, “An Analysis of Linux RAM Forensics,” pp. 1–89, 2006.
- [117] “Volatility,” 2019. [Online]. Available: <https://www.volatilityfoundation.org>.

- [Accessed: 26-Jun-2019].
- [118] K. Savage, P. Coogan, and H. Lau, "Information Resources," *Research-Technology Management*, vol. 54, no. 5, pp. 59–63, 2011.
- [119] United States Government, "How to Protecting Your Networks from Ransomware," 2016. [Online]. Available: <https://www.justice.gov/criminal-ccips/file/872771/download>. [Accessed: 26-Apr-2018].
- [120] J. M. Ehrenfeld, "WannaCry, Cybersecurity and Health Information Technology: A Time to Act," *Journal of Medical Systems*, vol. 41, no. 7, p. 104, 2017.
- [121] V. Kotov and M. S. Rajpal, "Understanding Crypto-Ransomware," *Bromium.com*, p. 35, 2015.
- [122] Sophos, "Stopping Fake Antivirus: How to Keep Scareware Off Your Network," 2011.
- [123] Symantec, "SpySherrif." [Online]. Available: <https://www.symantec.com/security-center/writeup/2005-122910-4625-99>. [Accessed: 17-May-2019].
- [124] Kaspersky, "Overall Statistics for 2017," Kaspersky, 2017. [Online]. Available: [https://kasperskycontenthub.com/securelist/files/2017/12/KSB\\_statistics\\_2017\\_EN\\_final.pdf](https://kasperskycontenthub.com/securelist/files/2017/12/KSB_statistics_2017_EN_final.pdf). [Accessed: 04-Apr-2018].
- [125] T. Halpin and T. Morgan, "Information Levels and Frameworks," in *Information Modeling and Relational Databases*, 2008.
- [126] T. Halpin and T. Morgan, *Information Modeling and Relational Databases*. 2008.
- [127] R. Finkel, "Operating systems," in *Computers, Software Engineering, and Digital Devices*, 2005, pp. 18-1-18–18.
- [128] P. C. Monali Chim, Vasundhara Rathod, "Linux & Windows Operating Systems," *Journal of Engineering, Computers & Applied Sciences (JEC&AS)*, 2013.
- [129] B. Kaplan, "RAM is Key: Extracting Disk Encryption Keys From Volatile Memory," p. 20, 2007.
- [130] A. Basu, J. Gandhi, J. Chang, M. D. Hill, and M. M. Swift, "Efficient virtual memory for big memory servers," in *Proceedings - International Symposium on Computer Architecture*, 2013, pp. 237–248.
- [131] H. Pomeranz, "Detecting Malware With Memory Forensics Why Memory Forensics ? Everything in the OS traverses RAM," pp. 1–27, 2012.
- [132] F. Olajide and N. Savage, "On the Extraction of Forensically Relevant Information from Physical Memory," *IEEE World Congress on Internet Security (WorldCIS)*, pp. 248–252, 2011.
- [133] K. A. Alghafli, A. Jones, and T. A. Martin, "Forensic Analysis of the Windows 7 Registry," *JDFSL*, vol. 5, no. 4, pp. 5–30, 2010.
- [134] L. W. Wong, "Forensic Analysis of the Windows Registry | Forensic Focus - Articles," *Digital Forensics Articles and Research Papers*, 2011.
- [135] M. Gruhn and F. C. Freiling, "Evaluating atomicity, and integrity of correct memory acquisition methods," *Digital Investigation*, vol. 16, pp. S1–S10, 2016.
- [136] R. B. van Baar, W. Alink, and A. R. van Ballegooij, "Forensic memory analysis: Files mapped in memory," *Digital Investigation*, vol. 5, no. SUPPL., pp. 52–57, 2008.
- [137] G. Andriana Mutiara and R. Wijaya, "Digital Forensics Random Access Memory Using Live Technique Based On Network Attacked," vol. 0, no. c,

- 2017.
- [138] S. Vömel and F. C. Freiling, "A survey of main memory acquisition and analysis techniques for the windows operating system," *Digital Investigation*, vol. 8, no. 1, pp. 3–22, 2011.
  - [139] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender Systems Handbook, Second Edition*, 2015, pp. 191–226.
  - [140] K. Hausknecht, D. Foit, and J. Burić, "RAM data significance in digital forensics," *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015 - Proceedings*, no. May, pp. 1372–1375, 2015.
  - [141] D. N. Patil and B. B. Meshram, "Extraction of forensic evidences from windows volatile memory," in *2017 2nd International Conference for Convergence in Technology (I2CT)*, 2017, pp. 421–425.
  - [142] R. C. Dodge, C. Carver, and A. J. Ferguson, "Phishing for user security awareness," *Computers and Security*, vol. 26, no. 1, pp. 73–80, 2007.
  - [143] S. Abraham and I. S. Chengalur-Smith, "An overview of social engineering malware: Trends, tactics, and implications," *Technology in Society*, vol. 32, no. 3, pp. 183–196, 2010.
  - [144] "Authy." [Online]. Available: <https://authy.com/>. [Accessed: 19-Aug-2019].
  - [145] M. Alaeiyan, S. Parsa, and M. Conti, "Analysis and classification of context-based malware behavior," *Computer Communications*, vol. 136, no. January 2018, pp. 76–90, 2019.
  - [146] X. Wang, D. Feng, and P. Su, "Reconstructing a packed dll binary for static analysis," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009.
  - [147] Microsoft, "ListDLLs," 2016. [Online]. Available: <https://docs.microsoft.com/en-us/sysinternals/downloads/listdlls>. [Accessed: 05-Oct-2018].
  - [148] Y. Ki, E. Kim, and H. K. Kim, "A novel approach to detect malware based on API call sequence analysis," *International Journal of Distributed Sensor Networks*, vol. 2015, 2015.
  - [149] B. Dolan-Gavitt, "Forensic analysis of the Windows registry in memory," *Digital Investigation*, vol. 5, no. SUPPL., pp. 26–32, 2008.
  - [150] NIST, "Computer Forensics Tool Testing Program," 2014. [Online]. Available: [http://www.cftt.nist.gov/disk\\_imaging.htm](http://www.cftt.nist.gov/disk_imaging.htm). [Accessed: 12-Oct-2017].
  - [151] NIST, "NIST," 2019. [Online]. Available: <https://www.nist.gov/>. [Accessed: 12-Oct-2019].
  - [152] V. R. Kebande, N. K. Menza, and H. S. Venter, "Functional Requirements for Adding Digital Forensic Readiness as a Security Component in IoT Environments," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, no. 2, p. 342, 2018.
  - [153] P. S. Foundation, "Python," 2019. [Online]. Available: <https://www.python.org/>. [Accessed: 05-Nov-2019].
  - [154] "TkInter." [Online]. Available: <https://wiki.python.org/moin/TkInter>. [Accessed: 08-Oct-2019].
  - [155] "OpenSSL." [Online]. Available: <https://wiki.openssl.org/index.php/Binaries>. [Accessed: 17-Jun-2019].
  - [156] A. Kiezun, P. J. Guo, K. Jayaraman, and M. D. Ernst, "Automatic creation of SQL injection and cross-site scripting attacks," in *Proceedings - International Conference on Software Engineering*, 2009.

- [157] M. L. Das and N. Samdaria, "On the security of SSL/TLS-enabled applications," *Applied Computing and Informatics*, 2014.
- [158] S. Omeleze and H. S. Venter, "Proof of concept of the online neighbourhood watch system," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 2016, vol. 171, pp. 78–93.
- [159] S. Pandey and M. Farik, "Best Symmetric Key Encryption - A Review," *International Journal of Scientific & Technology Research*, vol. 6, no. 6, pp. 310–312, 2017.
- [160] D. S. Abd Elminaam, H. M. A. Kader, and M. M. Hadhoud, "Evaluating the performance of symmetric encryption algorithms," *International Journal of Network Security*, vol. 10, no. 3, pp. 213–219, 2010.
- [161] R. Sears, C. Van Ingen, and J. Gray, "To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?," *Microsoft Research*, pp. 1–11, Jan. 2007.
- [162] T. D. Morgan *et al.*, "On the Extraction of Forensically Relevant Information from Physical Memory," *Digital Investigation*, vol. 2, no. 12, pp. 1–28, 2008.
- [163] "Docker," 2019. [Online]. Available: <https://www.docker.com/>. [Accessed: 14-Sep-2019].
- [164] J. R. D. M'Raihi, S. Machani, M. Pei, "TOTP: Time-Based One-Time Password Algorithm," *Journal of Chemical Information and Modeling*, 2013.
- [165] Google, "Google Authenticator," 2019. [Online]. Available: <https://github.com/google/google-authenticator>. [Accessed: 02-Nov-2019].
- [166] Yubico, "YubiKey." [Online]. Available: <https://www.yubico.com/>. [Accessed: 18-Oct-2019].
- [167] DjangoProject, "Django Auth." [Online]. Available: <https://docs.djangoproject.com/en/2.2/topics/auth/passwords/>. [Accessed: 04-Oct-2019].
- [168] Q. X. Miao, "Research and analysis on Encryption Principle of TrueCrypt software system," in *2nd International Conference on Information Science and Engineering, ICISE2010 - Proceedings*, 2010, pp. 1409–1412.
- [169] E. Spero, M. Stojmenović, and R. Biddle, "Helping Users Secure Their Data by Supporting Mental Models of VeraCrypt," in *International Conference on Human-Computer Interaction*, 2019, pp. 211–218.
- [170] L. L. Yuval Nativ, "theZoo," 2015. [Online]. Available: <https://github.com/ytisf/theZoo>. [Accessed: 12-Sep-2018].
- [171] S. Vajapeyam, "Understanding Shannon's Entropy metric for Information," 24-Mar-2014. [Online]. Available: <http://arxiv.org/abs/1405.2061>. [Accessed: 10-Feb-2019].
- [172] Whitewood Encryption Systems, "Understanding And Managing Entropy," *Whitewood Encryption Systems*, pp. 1–7, 2015.
- [173] K. A. Alghafli, A. Jones, and T. A. Martin, "Forensic Analysis of the Windows 7 Registry," *JDFSL*, vol. 5, no. 4, pp. 5–30, 2010.
- [174] C. Framework, "Cuckoo Community." [Online]. Available: <https://github.com/cuckoosandbox/community>. [Accessed: 02-Nov-2019].
- [175] Comptroller and Auditor General, "Investigation: WannaCry cyber attack on the NHS," *Hc*, vol. 414, no. April 2018, p. 6, 2017.
- [176] C. Sienko, "InfosecInstitute." [Online]. Available: <https://resources.infosecinstitute.com/category/healthcare-information-security/healthcare-attack-statistics-and-case-studies/ransomware-case->

- [studies-hollywood-presbyterian-and-the-ottawa-hospital/](#).
- [177] 24by7security, "Ransomware Statistics," 2018. [Online]. Available: <https://24by7security.com/have-you-scheduled-your-first-cybersecurity-task-in-2018-here-are-some-interesting-2017-statistics/>. [Accessed: 01-May-2018].
- [178] C. Osborne, "Texas hospital becomes victim of Dharma ransomware," 2018. [Online]. Available: <https://www.zdnet.com/article/texas-hospital-becomes-victim-of-ransomware-patient-data-potentially-leaked/>. [Accessed: 15-Aug-2019].
- [179] Emily Stewart, "Hackers have been holding the city of Baltimore's computers hostage for 2 weeks," 2019. [Online]. Available: <https://www.vox.com/recode/2019/5/21/18634505/baltimore-ransom-robbinhood-mayor-jack-young-hackers>. [Accessed: 14-Aug-2019].
- [180] M. Tory and T. Möller, "Evaluating Visualizations: Do Expert Reviews Work?," *IEEE computer graphics and applications*, vol. 25, no. 5, pp. 8–11, 2005.
- [181] A. Powell, A. Vickers, E. Williams, and B. Cooke, "A practical strategy for the evaluation of software tools," in *Method Engineering*, Springer Science+Business Media Dordrecht 1996, 1996, pp. 165–185.
- [182] M. S. Zareen, B. Aslam, and M. Akhlaq, "Criteria for validating secure wiping tools," in *IFIP Advances in Information and Communication Technology*, 2015.
- [183] P. Jokela, P. Karlsudd, and M. Östlund, "Theory, method and tools for evaluation using a systems-based approach," *Electronic Journal of Information Systems Evaluation*, vol. 11, no. 3, pp. 197–212, 2008.
- [184] H. Chang fu, L. Rong kwei, K. He yau, and L. Amy HI, "A systematic evaluation model for solar cell technologies," *Mathematical Problems in Engineering*, vol. 2014, pp. 1–16, 2014.
- [185] I. Etikan, "Comparison of Convenience Sampling and Purposive Sampling," *American Journal of Theoretical and Applied Statistics*, vol. 5, no. 1, p. 1, 2016.
- [186] R. L. Thompson, C. A. Higgins, and J. M. Howell, "Personal Computing: Toward a Conceptual Model of Utilization," *MIS Quarterly*, 2006.
- [187] C. A. Compeneau, D. R. Higgins, "Development of a measure and initial test," *Management Information Systems*, vol. 19, no. 2, pp. 189–211, 2012.
- [188] A. Valjarevic, H. Venter, and R. Petrovic, "ISO/IEC 27043:2015 - Role and application," *24th Telecommunications Forum, TELFOR 2016*, pp. 1–4, 2017.
- [189] D. Nieuwenhuizen, "A behavioural-based approach to ransomware detection," *MWR Labs Whitepaper*, 2017.
- [190] P. Raunak and P. Krishnan, "Network detection of ransomware delivered by exploit kit," *ARNP Journal of Engineering and Applied Sciences*, vol. 12, no. 12, pp. 3885–3889, 2017.
- [191] R. Brewer, "Ransomware attacks: detection, prevention and cure," *Network Security*, vol. 2016, no. 9, pp. 5–9, 2016.
- [192] G. Cusack, O. Michel, and E. Keller, "Machine learning-based detection of ransomware using SDN," in *SDN-NFVSec 2018 - Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks and Network Function Virtualization, Co-located with CODASPY 2018*, 2018, pp. 1–6.
- [193] BlackBerry, "Cylance." [Online]. Available: <https://www.cylance.com/>. [Accessed: 05-Oct-2019].
- [194] S. Boulevard, "A universal bypass tricks Cylance AI antivirus." [Online].

- Available: <https://securityboulevard.com/2019/07/a-universal-bypass-tricks-cylance-ai-antivirus-into-accepting-all-top-10-malware-revealing-a-new-attack-surface-for-machine-learning-based-security/>. [Accessed: 05-Oct-2019].
- [195] Skylight, “Cylance, I Kill You!,” 2019. [Online]. Available: <https://skylightcyber.com/2019/07/18/cylance-i-kill-you/>. [Accessed: 05-Oct-2019].
- [196] SCMagazine, “Researchers bypass Cylance’s AI-based AV solution by masking malware with video game code,” 2019. [Online]. Available: <https://www.scmagazine.com/home/security-news/researchers-bypass-cylances-ai-based-av-solution-by-masking-malware-with-video-game-code/>. [Accessed: 05-Oct-2019].
- [197] A. Tseng *et al.*, “Deep Learning for Ransomware Detection,” *IEICE Technical Report; IEICE Tech. Rep.*, vol. 116, no. 282, pp. 87–92, 2016.
- [198] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, “A survey of deep learning-based network anomaly detection,” *Cluster Computing*, pp. 1–13, 2017.
- [199] A. Continella *et al.*, “ShieldFS: A Self-healing , Ransomware-aware Filesystem,” *Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC ’16*, 2016.
- [200] M. Mohlala, I. R. Adeyemi, and H. S. Venter, “User Attribution based on Keystroke Dynamics in Digital Forensic Readiness Process,” in *IEEE Conference on Applications, Information and Network Security (AINS)*, 2017, pp. 124–129.
- [201] A. Valjarevic and H. S. Venter, “Towards a Digital Forensic Readiness Framework for Public Key Infrastructure systems,” *2011 Information Security for South Africa*, pp. 1–10, 2011.
- [202] V. R. Kebande and H. S. Venter, “On digital forensic readiness in the cloud using a distributed agent-based solution: issues and challenges,” *Australian Journal of Forensic Sciences*, vol. 50, no. 2, pp. 209–238, 2018.
- [203] V. R. Kebande, N. M. Karie, and H. S. Venter, “Functional requirements for adding digital forensic readiness as a security component in IoT environments,” *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, no. 2, pp. 342–349, 2018.
- [204] F. Mouton and H. S. Venter, “A prototype for achieving digital forensic readiness on wireless sensor networks,” *IEEE AFRICON Conference*, no. September, pp. 1–6, 2011.
- [205] J. MacRae and V. N. L. Franqueira, “On locky ransomware, Al Capone and Brexit,” in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 2018.
- [206] T. Haigh, F. Breitingger, and I. Baggili, “If I Had a Million Cryptos: Cryptowallet Application Analysis and a Trojan Proof-of-Concept,” in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 2019, vol. 259, pp. 45–65.
- [207] M. Satheesh Kumar, J. Ben-Othman, and K. G. Srinivasagan, “An Investigation on Wannacry Ransomware and its Detection,” *Proceedings - IEEE Symposium on Computers and Communications*, vol. 2018-June, pp. 1199–1204, 2018.