DOCTOR OF PHILOSOPHY

Network-based advanced malware detection using multi-classifier machine learning

Almashhadani, Ahmad

Award date:
2021

Awarding institution:
Queen's University Belfast

Link to publication

Terms of use
All those accessing thesis content in Queen’s University Belfast Research Portal are subject to the following terms and conditions of use

• Copyright is subject to the Copyright, Designs and Patent Act 1988, or as modified by any successor legislation
• Copyright and moral rights for thesis content are retained by the author and/or other copyright owners
• A copy of a thesis may be downloaded for personal non-commercial research/study without the need for permission or charge
• Distribution or reproduction of thesis content in any format is not permitted without the permission of the copyright holder
• When citing this work, full bibliographic details should be supplied, including the author, title, awarding institution and date of thesis

Take down policy
A thesis can be removed from the Research Portal if there has been a breach of copyright, or a similarly robust reason.
If you believe this document breaches copyright, or there is sufficient cause to take down, please contact us, citing details. Email: openaccess@qub.ac.uk

Supplementary materials
Where possible, we endeavour to provide supplementary materials to theses. This may include video, audio and other types of files. We endeavour to capture all content and upload as part of the Pure record for each thesis.
Note, it may not be possible in all instances to convert analogue formats to usable digital formats for some supplementary materials. We exercise best efforts on our behalf and, in such instances, encourage the individual to consult the physical thesis for further information.
Network-based Advanced Malware Detection Using Multi-Classifier Machine Learning

Ahmad Obaid Salman Almashhadani, MSc

School of Electronics, Electrical Engineering and Computer Science
Queen’s University Belfast, UK

A thesis submitted for the degree of

Doctor of Philosophy

December 2020
Abstract

Over the past decade, cyber threats have significantly evolved in persistence and sophistication. Malware has been the primary choice of weapon to carry out various cyberattacks. Host-based malware detection, as the primary line of defence, evolved into the “Achilles Heel”. In particular, the increase of security-aware targeted attacks, comprises of reconnaissance and delivery phases, are capable of identifying deployed security tools and disabling these without being detected. Hence, the deployment of advanced, network-based Intrusion Detection System (IDS) has become an inevitable line-of-defence assisting host-based malware detection.

Ransomware is a kind of advanced malware that has spread rapidly in recent years, causing massive financial losses for a broad range of victims, such as healthcare facilities, companies, and individuals. Modern host-based detection methods require the host to be infected first to be able to identify anomalies and detect the malware. By the time of infection, it may be too late as some of the system’s assets would have been already encrypted or exfiltrated by the malware. Conversely, the network-based approach can be an effective detection method as most families of ransomware attempt to contact with command and control (C&C) servers before their harmful payloads are
executed. Also, some recent ransomware families have evolved and combined the propagation properties of computer worms to be able to spread across the networks.

A network-based ransomware detection approach, which complements well-established host-based ransomware detection methods, can be one of the essential means for detecting ransomware attack effectively. It can overcome the limitations of current ransomware defence while enabling early detection and timely deployment of countermeasures. State-of-the-art presents little research work that focuses on network-based approaches for ransomware detection.

This thesis investigates the use of machine learning techniques for detecting crypto ransomware network activities. A thorough dynamic analysis of crypto ransomware network traffic is carried out using a dedicated malware testbed. A set of 18 network-based features are extracted from several network protocols of Locky, one of the well-established ransomware families. A new classification scheme is introduced to classify the features into four types. A multi-feature and multi-classifier intrusion detection system is proposed and implemented for detecting the communications between ransomware and its C&C server. This new approach employs two independent classifiers working in parallel on two levels: packet and flow. The experimental evaluation of the presented detection system demonstrates that the system offers high detection accuracy for each level: 97.92% and 97.08% respectively.
Second, machine learning techniques are used to detect covert C&C channels established using Domain Generation Algorithm (DGA). DGA is one of the main techniques deployed by ransomware and botnet to connect with attackers by generating many pseudorandom domain names. A malicious domain name detection system, called MaldomDetector, is introduced. Prototyped MaldomDetector can detect the DGA-based communications before the malware is able to establish a successful connection with the C&C server, basing only on the used characters for the domain name. MaldomDetector deploys a deterministic algorithm and easy to compute features extracted out of the domain name characters. It is not based on any probabilistic language model, i.e., a language-independent system, and does not utilise any data from an external site or wait for a DNS response packet; hence, significantly reducing the time and computation required to classify the domain names. The evaluation results demonstrate that MaldomDetector provides high accuracy of 98% in detecting different types of DGA-based domains. MaldomDetector can be employed as an early warning system to raise early alarms about potential malicious DNS communications.

Finally, a multi-feature and multi-classifier network-based system (MFM-CNS) is presented for detecting ransomware propagation activities. A comprehensive analysis of ransomware traffic is performed, and two sets of features are extracted based on two independent flow levels: session-based and time-based. Also, two individual classifiers are built employing the two different feature sets. The experimental
results demonstrate a high detection accuracy for the session-based and time-based classifiers: 99.88% and 99.66% respectively validating the effectiveness of the extracted features. MFMCNS employs these classifiers in parallel on different levels where the classifiers’ decisions are combined using a fusion rule. Experimental results validate that the overall MFMCNS detection accuracy and reliability have been enhanced.
Acknowledgements

All praise and thanks are due to my Almighty Allah for helping me complete my PhD; his favours on me are countless and unlimited.

It is my pleasure to acknowledge the people without whom this thesis would not have been possible. Primarily, I would like to express my sincere gratitude to my supervisors, Prof Sakir Sezer, Dr Mustafa Kaiiali and Dr Philip O’Kane for their supervision of my PhD, intellectual guidance, encouragement, and endless support.

I owe a debt of gratitude to Dr Domhnall Carlin for the invaluable proofreading, and constructive feedback. I would like to offer special thanks to Prof Máire O’Neill and Dr Kieran McLaughlin for their very helpful comments and recommendations.

I wish to thank the University of Baghdad for granting me a scholarship to pursue my PhD degree in the UK. I would also like to thank my colleagues Dr Mohammed Alzaylaee, Dr Harsha Kumara Kalutarage, Dr Fahad Siddiqui, Dr Rafiullah Khan, Dr Mathew Hagan, and Dr Abdullah Al Balushi for the assistance they have provided me whilst completing my thesis.

Finally, thanks to all my friends and colleagues at Queen’s University Belfast, EEECS and ECIT, who have provided help at various times.
To my dear parents, Maryam and Obaid, for nursing me with affection and love, and their dedicated partnership for success in my life.

To my beloved wife, Samah, without her love and support, this work would not have been possible.

To my amazing children, Maryam, Mohammed, Retaj, and Mustafa, for their constant encouragement and great love.

To my lovely brothers and sisters for their support and offering me help whenever I need it.

To my lovely relatives and friends.

To my beloved country Iraq.
Table of Contents

Table of Contents vii
List of Tables xii
List of Figures xv
List of Acronyms xx

1 Introduction 1
1.1 Research context ................................................. 1
1.2 Research aim .................................................... 6
1.3 Thesis objectives ................................................ 7
1.4 Thesis structure ................................................... 9

2 Background 11
2.1 Introduction ...................................................... 11
2.2 Cyber threats .................................................... 11
  2.2.1 Network-based threats ..................................... 15
  2.2.2 Domain Generation Algorithm (DGA) ................... 18
2.3 Malware threats .................................................. 22
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1</td>
<td>Taxonomy of malware</td>
<td>25</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Malware analysis techniques</td>
<td>28</td>
</tr>
<tr>
<td>2.4</td>
<td>Ransomware</td>
<td>30</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Ransomware taxonomy</td>
<td>32</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Network-based ransomware activities</td>
<td>36</td>
</tr>
<tr>
<td>2.5</td>
<td>Network-based malware detection techniques</td>
<td>42</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Signature-based detection</td>
<td>43</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Heuristic-based detection technique</td>
<td>44</td>
</tr>
<tr>
<td>2.5.3</td>
<td>Anomaly-based detection</td>
<td>45</td>
</tr>
<tr>
<td>2.5.4</td>
<td>Specification-based detection</td>
<td>45</td>
</tr>
<tr>
<td>2.6</td>
<td>Machine learning-based detection</td>
<td>46</td>
</tr>
<tr>
<td>2.6.1</td>
<td>Need for machine learning</td>
<td>49</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Machine learning methods</td>
<td>51</td>
</tr>
<tr>
<td>2.6.3</td>
<td>Classification algorithms</td>
<td>53</td>
</tr>
<tr>
<td>2.7</td>
<td>Summary</td>
<td>59</td>
</tr>
</tbody>
</table>

3 A Multi-Classifier ML Approach for Network-Based Crypto Ransomware Detection

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>60</td>
</tr>
<tr>
<td>3.2</td>
<td>Related work</td>
<td>61</td>
</tr>
<tr>
<td>3.3</td>
<td>Setting up the testbed</td>
<td>65</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Malware testbed environment</td>
<td>65</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Ransomware dataset</td>
<td>66</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Locky ransomware samples collection</td>
<td>68</td>
</tr>
<tr>
<td>3.4</td>
<td>Locky network traffic analysis and features extraction</td>
<td>69</td>
</tr>
</tbody>
</table>
## TABLE OF CONTENTS

3.4.1 TCP traffic analysis ................. 72
3.4.2 DNS traffic analysis .................. 78
3.4.3 NetBIOS name service (NBNS) ............ 85
3.5 Building a multi-classifier intrusion detection system .......... 87
  3.5.1 Data preparation .................. 89
  3.5.2 Models building .................. 90
  3.5.3 Models evaluation .................. 91
3.6 Summary ................................ 94

4 An ML based Approach for Detecting Algorithmically Generated Domain Names 95
  4.1 Introduction ........................ 95
  4.2 Related work .................. 96
  4.3 MaldomDetector high-level architecture .................. 101
  4.4 MaldomDetector implementation .................. 101
    4.4.1 Raw dataset collection .................. 101
    4.4.2 Domain name analysis .................. 103
  4.5 Randomness Measuring Algorithm (RMA) ............. 106
  4.6 Building and evaluating the domain name classifier (MaldomDe-
tector) .................. 111
    4.6.1 Feature extraction and selection .................. 111
    4.6.2 Building a labelled dataset .................. 113
    4.6.3 Model training .................. 114
    4.6.4 Evaluating the classifier .................. 115
  4.7 Discussion ........................ 119
## Table of Contents

4.8 Summary .................................................. 121

5 Multi-Feature and Multi-Classifier ML Approach for Network-based Ransomworm Detection ........................................... 122

5.1 Introduction .................................................. 122
5.2 Related work .................................................. 123
5.3 How a ransomworm works ..................................... 126
  5.3.1 WannaCry network activities ............................... 127
  5.3.2 NotPetya network activities ............................... 129
5.4 Dataset description ............................................ 129
5.5 Network traffic analysis and feature extraction .................. 130
  5.5.1 Kill switch property ...................................... 132
  5.5.2 The TCP connection ...................................... 132
  5.5.3 Feature extraction ...................................... 142
5.6 Building the network-based classifiers ......................... 144
  5.6.1 Building the dataset ...................................... 144
  5.6.2 Feature selection ...................................... 145
  5.6.3 Building the classifiers .................................. 149
  5.6.4 Model evaluation ...................................... 149
5.7 A Multi-Feature and Multi-Classifier Network-based System (MFM-CNS) for ransomworm detection ................................. 151
  5.7.1 System topology ...................................... 155
  5.7.2 Classifier dependencies .................................. 156
  5.7.3 Fusion unit ...................................... 157
  5.7.4 Further evaluation .................................. 159
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.8</td>
<td>Discussion</td>
<td>162</td>
</tr>
<tr>
<td>5.9</td>
<td>Summary</td>
<td>164</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions and Future Work</td>
<td>165</td>
</tr>
<tr>
<td>6.1</td>
<td>Conclusions</td>
<td>165</td>
</tr>
<tr>
<td>6.2</td>
<td>Main Contributions</td>
<td>166</td>
</tr>
<tr>
<td>6.3</td>
<td>Future work</td>
<td>170</td>
</tr>
<tr>
<td>A</td>
<td>Author’s Publications</td>
<td>174</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>175</td>
</tr>
</tbody>
</table>
List of Tables

2.1 DGA classes .................................................. 22

3.1 Brief information about Locky and benign PCAP files that were collected from MCFP datasets ........................................ 69
3.2 Network protocols’ statistics .................................... 71
3.3 Some examples of the meaningless domain names ................. 81
3.4 Examples of the entropy, max. no. of sequential consonants and max. no. of sequential vowels computed for some domain names . 82
3.5 The ranked attributes by Weka ................................ 85
3.6 Summary of the extracted features classified into: Behavioral vs Non-behavioral & Detectable vs Non-detectable .............. 87
3.7 Summary of the extracted features classified into: Packet-level vs Flow-level & Simple vs Complex ............................... 88
3.8 The selected packet-level and flow-level features ................. 90
3.9 Percentages of the training and testing datasets .................. 91
3.10 Confusion matrix ................................................. 91
3.11 Packet-based classifier evaluation .............................. 93
3.12 Flow-based classifier evaluation ................................ 93
4.1 Malware families employing DGA that were collected from the
   DGArchive and Bambenek feeds .................................. 104
4.2 Some samples of DGA-based domain names ..................... 105
4.3 RMA evaluation results of DGA families and Alexa .............. 109
4.4 The basic and derived features ................................ 111
4.5 The selected features ........................................... 113
4.6 The summary of the dataset used to build MaldomDetector .... 114
4.7 The evaluation results of the models ............................ 116
4.8 The dataset used for further testing ............................ 118
4.9 Testing results of the selected models .......................... 118
4.10 The properties of the DGA-based domains detection systems  .. 119

5.1 Brief information of PCAP files from MCFP dataset used in this
   chapter .............................................................. 131
5.2 Session-based extracted features ................................. 143
5.3 Time-based extracted features ................................. 144
5.4 The datasets of session-based and time-based ...................... 145
5.5 The selected session-based features ............................. 148
5.6 The selected time-based features ............................... 148
5.7 Session-based classifier evaluation .............................. 150
5.8 Time-based classifier evaluation ................................ 151
5.9 Alarms types of the fusion unit ................................ 159
5.10 Brief information of other PCAP files used for extra evaluation. 160
5.11 New datasets for extra evaluation ................................ 161
5.12 Further evaluation results of MFMCNS .......................... 161
5.13 Some properties of the ransomworm detection methods. . . . . . . 162
List of Figures

2.1 Cybersecurity overview [1] .............................................. 13
2.2 Variety of attacks [2] .................................................. 15
2.3 Examples of possible NIDS sensor locations [3] .................. 17
2.4 Ramdo DGA Python code [4] ........................................ 20
2.5 The execution flow of DGA [4] ....................................... 21
2.6 Traditional vs. advanced malwares ................................. 23
2.7 Malware taxonomy ..................................................... 26
2.8 Ransomware taxonomy based on various factors [5] .......... 33
2.9 Crypto ransomware lifecycle [6] ..................................... 33
2.10 Crypto ransomware network communications [7] (a) Symmetrical

    encryption, (b) Asymmetrical encryption ............................ 35
2.11 The communications between Crypto Locker and C&C server [8] . 36
2.12 Ransomware’s attack landscape [9] ............................... 38
2.13 View of an SDN architecture [10] .................................. 41
2.14 Taxonomy of malware detection techniques ....................... 43
2.16 Understanding ML [12] ............................................... 47
2.17 The workflow implementation of the ML system [13].................................. 48
2.18 Components of the ML model ........................................................................ 48
2.19 A typical workflow (a) conventional programming paradigm (b) ML paradigm ................................................................. 49
2.20 The importance of ML [14] ............................................................................... 50
2.21 Supervised learning method [15] ................................................................... 53
2.22 Decision tree formulation [16] ........................................................................ 54
2.23 SVM hyperplane [13] ..................................................................................... 59
3.1 The malware testbed architecture .................................................................... 67
3.2 The reset connection ratio ................................................................................ 72
3.3 The frequency of the TCP reset packets (a) within the 1st hour (b) within the 2nd hour (c) within the 3rd hour ................................. 74
3.4 Samples of the IP-wise reset connections ratio (a) malicious samples (b) benign samples ................................................................. 75
3.5 Samples of the IP-wise reset connections ratio in a subsequent time-frame (a) malicious samples (b) benign samples ............... 75
3.6 Locky’s malicious channels ............................................................................. 76
3.7 HTTP POST method ratio: Malicious Vs Benign ............................................ 77
3.8 HTTP POST method frequencies that were taken over subsequent time-frames ................................................................. 78
3.9 Samples of the name errors within the Locky’s traffic .................................... 79
3.10 Number of the DNS name errors in the malicious vs the benign traffic ................................................................. 80
LIST OF FIGURES

3.11 DNS name error frequencies that were taken over subsequent time-frames .......................................................... 80
3.12 The top frequencies of meaningless DNS domain names ........ 82
3.13 The flow chart of the domain name randomness measuring algorithm 84
3.14 Attributes selection using Weka .................................................. 85
3.15 The DNS labels’ count: Malicious Vs Benign ......................... 86
3.16 The top frequencies of NBNS queries within the malicious traffic .... 86
3.17 The architecture of the multi-classifier detection system ........ 89
3.18 Building labelled datasets out of the raw PCAP files ............. 90
3.19 Performance metrics of each algorithm for the packet-based classifier 93
3.20 Performance metrics of each algorithm for the flow-based classifier 93

4.1 The architecture of MaldomDetector system ......................... 102
4.2 The frequency of the sequential consonant letters: Benign vs DGA-based ...................................................... 105
4.3 The normal distribution curve for the Malicious and benign domains106
4.4 The flow chart of Randomness Measuring Algorithm (RMA) .... 107
4.5 Ratio of the entropy values in DGA-based and benign domains .. 108
4.6 The plot of the feature importance .............................................. 113
4.7 Training of the machine learning models using the classification learner application of MATLAB ......................... 115
4.8 Performance metric of the models .............................................. 117
4.9 ROC curve and AUC of the selected models: (a) Fine Tree (b) Boosted Tree (c) Gaussian Naïve Bayes (d) Linear SVM (e) Coarse KNN ......................................................... 117
5.1 Meaningless domain name queried by WannaCry. ............... 133
5.2 The ratio of communications with external IP addresses (ext-ip-addr) without using DNS service: Malicious & Benign. .......... 134
5.3 The ratio of the communications used SMB port: Malicious & Benign. ......................................................... 134
5.4 The ratio of the SYN-ACK (established TCP connections): Malicious & Benign .................................................. 135
5.5 The normal distribution curves for the number of established TCP connections (SYN-ACK packets) every 10 seconds: Malicious & benign ................................................................. 136
5.6 Sending (RST, ACK) directly after receiving SYN packet. ...... 137
5.7 The ratio of the sending SYN packets that received (RST, ACK) packet: Malicious & Benign) every 10 seconds: Malicious & benign. 137
5.8 Frequency of the SYN-RST packets (a) within 1st three minutes (b) within 2nd three minutes (c) within 3rd three minutes. .... 138
5.9 ICMP packet types received by ransomworm after attempting to establish TCP connections. ................................. 139
5.10 The ratio of the SYN-ICMP packets: Malicious & Benign. .... 140
5.11 The ratio of the SYN packets that received no response (syn-noresp): Malicious & Benign. ................................. 140
5.12 Frequency of the SYN-No Response packets (a) within 1st three minutes (b) within 2nd three minutes. ....................... 141
5.13 WannaCry traffic that indicates the short-duration connection and reconnection packets. ................................. 142
5.14 The ratio of the short-duration sessions: Malicious & Benign. 142
5.15 Building labelled datasets out of the raw PCAP files. . . . . . . . 145
5.16 The chart of the importance between each session-based feature
and the label. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 146
5.17 The chart of the importance between each time-based feature and
the label. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 147
5.18 ROC curve and AUC of the session-based selected models: (a) Fine
    Tree (b) Cubic KNN (c) Quadratic SVM (d) Linear discriminant . 152
5.19 ROC curve and AUC of the time-based selected models: (a) Bagged
    Tree (b) Cubic SVM (c) Cubic KNN (d) Linear discriminant . . 153
5.20 The overview architecture of MFMCNS. . . . . . . . . . . . . . . . 154
5.21 The architecture of MFMCNS. . . . . . . . . . . . . . . . . . . . . 155
# List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACR</td>
<td>Asymmetric Cryptosystem Ransomware.</td>
</tr>
<tr>
<td>AES</td>
<td>Advanced Encryption Standard.</td>
</tr>
<tr>
<td>AGDs</td>
<td>Algorithmically Generated Domains.</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence.</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface.</td>
</tr>
<tr>
<td>APT</td>
<td>Advanced Persistent Threats.</td>
</tr>
<tr>
<td>AUC</td>
<td>Area Under the Curve.</td>
</tr>
<tr>
<td>AWS</td>
<td>Amazon Web Services.</td>
</tr>
<tr>
<td>C&amp;C</td>
<td>Command and Control.</td>
</tr>
<tr>
<td>CGR</td>
<td>Cryptographic Ransomwares.</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit.</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma-Separated Values.</td>
</tr>
<tr>
<td>DDoS</td>
<td>Distributed Denial of Service.</td>
</tr>
<tr>
<td>DGA</td>
<td>Domain Generation Algorithm.</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic-link library.</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System.</td>
</tr>
<tr>
<td>DT</td>
<td>Decision Trees.</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>EK</td>
<td>Exploit Kits.</td>
</tr>
<tr>
<td>FPGAs</td>
<td>Field Programmable Gate Arrays.</td>
</tr>
<tr>
<td>FPR</td>
<td>False Positive Rate.</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol.</td>
</tr>
<tr>
<td>GCS</td>
<td>Google Cloud Storage.</td>
</tr>
<tr>
<td>HCR</td>
<td>Hybrid Cryptosystem Ransomware.</td>
</tr>
<tr>
<td>HIDS</td>
<td>Host-based Intrusion Detection System.</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov model.</td>
</tr>
<tr>
<td>HPCs</td>
<td>Hardware Performance Counters.</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol.</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol.</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System.</td>
</tr>
<tr>
<td>IOC</td>
<td>Indicators of Compromise.</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol.</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider.</td>
</tr>
<tr>
<td>KNN</td>
<td>K-Nearest Neighbour.</td>
</tr>
<tr>
<td>Malware</td>
<td>Malicious software.</td>
</tr>
<tr>
<td>MCFP</td>
<td>Malware Capture Facility Project.</td>
</tr>
<tr>
<td>MCS</td>
<td>Multi Classifier System.</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning.</td>
</tr>
<tr>
<td>NB</td>
<td>Naive Bayes.</td>
</tr>
<tr>
<td>NBNS</td>
<td>NetBIOS Name Service.</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>NCR</td>
<td>non-Cryptographic Ransomwares.</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card.</td>
</tr>
<tr>
<td>NIDS</td>
<td>Network-based Intrusion Detection System.</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System.</td>
</tr>
<tr>
<td>PCAP</td>
<td>packet capture.</td>
</tr>
<tr>
<td>PCC</td>
<td>Pearson’s correlation coefficient.</td>
</tr>
<tr>
<td>PFEs</td>
<td>Programmable Forwarding Engines.</td>
</tr>
<tr>
<td>PrCR</td>
<td>Private-key Cryptosystem Ransomware.</td>
</tr>
<tr>
<td>PuCR</td>
<td>Public-key Cryptosystem Ransomware.</td>
</tr>
<tr>
<td>RaaS</td>
<td>Ransomware-as-a-Service.</td>
</tr>
<tr>
<td>RAT</td>
<td>Remote Access Control.</td>
</tr>
<tr>
<td>RMA</td>
<td>Randomness Measuring Algorithm.</td>
</tr>
<tr>
<td>ROC</td>
<td>Receiver Operating Characteristic.</td>
</tr>
<tr>
<td>RSA</td>
<td>Rivest–Shamir–Adleman.</td>
</tr>
<tr>
<td>SDN</td>
<td>Software-Defined Networking.</td>
</tr>
<tr>
<td>SMB</td>
<td>Server Message Block.</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine.</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol.</td>
</tr>
<tr>
<td>TLDs</td>
<td>Top-Level Domains.</td>
</tr>
<tr>
<td>Tor</td>
<td>The Onion Router.</td>
</tr>
<tr>
<td>TTL</td>
<td>Time To Live.</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Research context

The 1960s to 1980s saw a merger of the fields of informatics and data communication, defining the foundation and the birth of the Internet in 1990. Over the past three decades the Internet has evolved and emerged as a critical infrastructure and the underpinning technology for enabling e-commerce, e-government, online education, social media, entertainment etc. Nowadays, most modern information technologies depend on the use of computer networks and have a significant impact on all aspects of life in developed societies [17].

With the increased dependence upon computer systems and the growth of the Internet, communication related cyber threats and targeted cyberattacks have also increased exponentially. According to Symantec more than one million attacks are carried out every day. As stated by the Symantec cybercrime report issued in April 2012, cyberattacks cost US$114 billion each year [18]. Symantec data has shown that 4,818 unique websites were compromised with formjacking
1.1 Research context

code every month in 2018. Formjacking is a malicious JavaScript code used to steal credit card details and collect sensitive information. According to Symantec, the overall web attacks on endpoint machines increased by 56% in 2018 compared to 2017, where the average web attacks blocked per day was about 953,800 [19]. Cyberattacks are becoming more prevalent, diverse and more sophisticated [18].

In general, Cybercrime is a term used to describe criminal activities that use specifically computer networks or the Internet as a tool, target, or place for malicious activities [3]. A cyberattack can be defined as a malicious and deliberate attempt to breach an individual or organisation’s information system [20]. The cyberattack aims to attempt to compromise the confidentiality, integrity, and availability of data. Cybercriminals typically aim to gain access to victim’s data by disrupting victim’s network or computer system [21].

Several types of cyberattacks can be launched by the attackers, such as (a) DDoS (Distributed Denial of Service) type attacks which compromises the availability of service; (b) man-in-the-middle type attacks, which compromises the confidentiality and integrity of data; and (c) malware attacks, which compromises the confidentiality, integrity, and availability of data. Malware, a contraction for malicious software, is a generic term pointing to any software intentionally designed to harm a computer system or network without the user’s knowledge. The word “software” implies many types of malicious code that are used by malware, such as executable code, scripts, and DLL [22]. Some common types of malware are viruses, worms, trojans, botnet, and ransomware [21].

Many cybersecurity experts have considered malware as a significant weapon to run various malicious activities to breach cybersecurity efforts in the cyberspace. Also, it helps in increasing the number of cyberattacks [18]. Malware
has become a dangerous threat that can cause many problems and launch various cyberattacks [23].

Nowadays, ransomware has emerged as one of the most serious malware types that threatens individuals and businesses alike [24]. Ransomware is a kind of malware that disables the functionality of a computer in some way or restricts access to the user’s files. It displays a message which demands payment of an amount of money to restore the functionality or files [25]. The amount of the ransom usually ranges between $300 and $700 for individuals and $10000 to $17000 for enterprises [5].

Ransomware has spread rapidly in recent years, causing significant financial losses for a wide range of victims like industries, healthcare companies, governments, and individuals. A rise in the number of undetected ransomware families was seen between 2015 and 2016. Symantec’s researchers observed about 4,000 ransomware attacks per day in the first quarter of 2016, with an increase of 300% in the attacks observed in 2015 [26].

SonicWall’s annual threat report in 2017 shows a tenfold increase in the number of ransomware attack attempts during 2016, up to 266.5 million attempts by the end of Q4. Ransomware remained to be one of the most dangerous cyber-attacks in 2017 and 2018 [27] [28]. GandCrab-ransomware infected more than 50,000 systems during only the first three weeks of 2018 [29].

The FBI estimates that the attackers behind ransomware collected $209 million in the first three months of 2016 [30]. Influential ransomware families have earned big revenue like CryptoWall version 3.0 gained an estimated $325 Million as ransoms in the USA alone [31]. The global ransomware financial loss is predicted to hit $20 billion in 2021, up from just $325 million in 2015, which is about
Unlike traditional malware, ransomware’s effect is irreversible and hard to mitigate against without the help of the ransomware’s author most of the time, especially for crypto ransomware that employs encryption. Besides the money and downtime costs that business entities and individuals could pay as a ransom, the victims could incur other harm such as loss of data and reputation [5].

Most ransomware attacks are targeted attacks in many cases tailored to victim’s (organisation) known vulnerabilities. The objective is to cause maximum disruption and to justify maximum ransom from the victims. Most of these targeted attacks use sophisticated tools and employ tactics widely used for cyberespionage [32].

The process of detecting activities that attempt to compromise the confidentiality, integrity or availability of a system or network is called intrusion detection. An Intrusion Detection System (IDS) is a hardware or software application that is designed to monitor the activities of a network or a system and analyse them to detect any intrusions or suspicious actions [33]. IDS complement the firewall security to achieve a defence-in-depth strategy and can be used for detecting the attacks originated from external or internal intruders. Intruders can get access to a system using several ways such as exploiting a vulnerability (a software weakness) or taking advantage of any misconfiguration errors [34]. IDS can be classified based on the data source into network-based IDS (NIDS) and host-based IDS (HIDS). A NIDS captures the incoming packets at a specific network segment using sensors, then it inspects these packets and triggers an alarm if an intrusion has been found. While HIDS monitors and collects the events of a host and analyses them to recognise any suspicious incidents [35].
Malware detection methods can be divided into host-based type and network-based type. Modern host-based detection approaches require the host to be infected first in order to identify anomalies and detect the malware. By the time of infection, it might be too late as some of the system’s assets would have been already encrypted or exfiltrated by the malware [36].

The network-based security system monitors network traffic over a network segment and is independent of the operating system type. This type of security provides more options for companies that run specialised software. Network-based system can be installed easily on a network segment and can capture the network packets with little overhead. Also, it is useful when network topology has been changed where the network-based systems can be moved and used as needed [37]. The network-based security system provides enhanced protection against the growing number of unknown threats that are difficult to block. Besides, analysing network traffic can yield significant results even in the presence of encryption [38].

Network-based methods are effective to detect sophisticated attacks during different stages. First, before downloading the payload on the victim’s system, this stage is called pre-breach. Second, after installing the payload, this stage is called post-breach. For example, most of the advanced malwares like ransomware attempt to connect to a C&C server after the target is infected. Finally, when malware tries to propagate across networks to infect other machines or networks [3]. Therefore, analysing network traffic can be one of the necessary means of detecting advanced attacks such as a ransomware attack [39].

Although many security systems have been presented to prevent or mitigate cybercrimes, cyberattacks keep evolving in terms of the number of attacks and the level of damage caused to their victims. The fight between cybercriminals
and security defenders has become more complicated [40] as malware has been
developed rapidly by malware authors.

Robust and effective network security systems are essential to avoid the big
extortion of ransomware and to protect the data of individuals and companies
from loss. This topic is an emerging field of study in academic research [41]. The
available works are not enough as new ransomware families employ a combination
of several sophisticated techniques to evade detection and expand their damage
[42]. The existing proposed methods for ransomware detection predominantly
focus on host-level monitoring, e.g., exploring the I/O requests to the file system
and tracking the file system activities, such as memory usage and processor usage.
A small amount of academic research has only focused on detecting ransomware
activities at the network-level [43].

Due to the emergence of Ransomware-as-a-Service (RaaS) platforms, ran-
somware will keep its leading position as a major threat in the future [5]. Since
the attack of ransomware evolves rapidly, the researches presented to mitigate
the ransomware’s effects become less effective after a short period [29]. Therefore,
more research work is required to stop the growing trend of ransomware
attacks [41] and develop network-based IDSs.

1.2 Research aim

This thesis aims to develop a network-based approach for the accurate and fast
detection of ransomware activities. A comprehensive network-based analysis is
performed that covers a carefully selected set of the most severe crypto ran-
sonware families and investigates the data obtained by analysing the packets
1.3 Thesis objectives

of various network protocols to extract informative and related features. Machine learning techniques are applied where the extracted features are fed into a machine learning engine to build a classification model acting as an IDS that can accurately and promptly discriminate between malicious and benign network packets. This model helps to detect the presence of ransomware early enough before it starts propagating over the network or communicating with its C&C server to execute its malicious payload.

Several IDSs have been implemented to detect different ways of C&C communication, including the DGA-based (Domain Generation Algorithm), e.g., Locky, and none-DGA based approaches, e.g., WannaCry. Furthermore, this thesis aims to use a machine learning technique to develop an accurate and fast detection system capable of detecting DGA-based domain names used by several types of advanced malware, such as ransomware and botnet.

1.3 Thesis objectives

The main overall challenges to be addressed in this thesis are:

- Configure a safe and controlled environment, capable of running ransomware samples, monitor and capture the network traffic, and provide a suitable environment for analysing the network activities.

- Perform a comprehensive behavioural analysis of crypto ransomware network communications when trying to connect with a C&C server.

- Extract two sets of informative features from flow and packet levels and classify them into different categories based on domain knowledge.
1.3 Thesis objectives

- Build a labelled dataset for each level out of the raw captured PCAP files.
- Use supervised learning techniques to build an independent classifier for each level.
- Present a multi-classifier network-based system for detecting the crypto ransomware communications with the C&C server. This system consists of two independent classifiers, operating in parallel on different levels.
- Build a detection system of algorithmically generated malicious domain names based on a deterministic algorithm and machine learning technique.
- Analyse the crypto ransomware network activities when trying to propagate over the network.
- Extract two sets of informative features from flow and time levels of ransomware propagation traffic based on domain knowledge.
- Build a labelled dataset for each level of the propagation traffic out of the raw captured PCAP files.
- Use supervised learning techniques to build two individual classifiers using the two selected feature sets. These classifiers are capable of detecting the propagation activities of crypto ransomware accurately.
- Present a multi-feature and multi-classifier network-based system for detecting ransomware propagation activities. This system consists of two independent classifiers, operating in parallel on different flow levels.
1.4 Thesis structure

The remainder of this thesis is presented as follows:


Chapter 3: Presents the proposed multi-classifier network-based crypto ransomware detection approach. Reviews related work and outlines lab setup and experimental environment, including the collected dataset and ransomware samples. Illustrates crypto ransomware network traffic analysis and feature extraction. Overall architecture of the proposed multi-classifier approach, including data preparation, model building, and model evaluation.

Chapter 4: Presents the proposed system for detecting algorithmically generated domain names. Outlines related work and the architecture of the proposed system. Details the implementation of the proposed system, including dataset collection, domain name analysis, and the randomness measuring algorithm. Describes the operation of the proposed detection system using machine learning technique, including feature extraction and selection, building a labelled dataset, model training, and evaluating the classifier.

Chapter 5: Presents a multi-feature and multi-classifier network-based system for ransomworm detection. Outlines related work, how a ransomworm works, and dataset description. Details the analysis of ransomworm network traffic and feature extraction. Outlines the construction of the individual network-based classifiers, including the construction of dataset, feature selection, training and evaluating the classifiers. Details the implementation of the proposed multi-
1.4 Thesis structure

feature and multi-classifier network-based system, including system topology, classifier dependencies, fusion method, and evaluating the system.

Chapter 6: Presents a concluding summary of this thesis, drawing together the conclusions made from this body of research and suggesting future research directions.
Chapter 2

Background

2.1 Introduction

This chapter provides related background information and context to explain the objectives and relevant area of research of this work. Host and network-based cyber-threats are reviewed, the concept of the Domain Generation Algorithm (DGA) is introduced, and various malware types and current malware detection techniques are discussed. Following this, the concept, types, and network activities of ransomware are discussed. Then, the concept of network-based malware detection is addressed. Finally, the concept of machine learning-based malware detection is introduced.

2.2 Cyber threats

Technological innovations of cyber systems are fuelled by the advancement of communication technology and the Internet [44]. The term cyber broadly de-
fines computers, computer networks, information technology, and the Internet. Cyberspace is the environment used to form a global computer network or the Internet to facilitate communications and data exchange activities [45].

Cyber threat is the possibility of a malicious action that intends to steal data or disrupt a computer system or network. It also refers to the possibility of a successful cyberattack that aims to compromise the confidentiality, integrity, or availability of the target. Some examples of cyber threats are credential thefts, data tampering, and account hijacking [46]. Recently, cyber threats have become more sophisticated, more widespread, and harder to detect [47].

Cybercriminals can be external or internal to the individual or organisation facing a cyberattack. An attack on the computer system or network performed by someone who has authorisation access is known as an insider attack. It is relatively easy for an insider to carry out a cyberattack because he has an awareness of the policies, IT architecture, and weakness of the running security system. Insider attacks can be prevented or mitigated by installing Internal IDS in the organisation’s network. The attacks that originate from outside the organisation and attempt to exploit security exposures in the attack surface that exist outside the firewall are known as external attacks. The organisation that is a victim of cyberattack faces not only financial loss but also reputational loss [46]. Fig. 2.1 shows the overview of cybersecurity.

Some top cyber threats are illustrated in the following [48] [45] [49]:

- **Malware**: It is any software that intentionally designed to cause any malicious tasks on a computer or network, such as stealing data or taking control of a system. Malware has remained the top cyber threat since 2014. More than four million samples of malware are detected by security com-
2.2 Cyber threats

panies every day in 2017. The increase in malware samples has escalated the malware attacks.

- **Denial of service (DoS) attack**: Is an attempt to compromise the availability of machine or network resources and make them unavailable to legitimate users. An example is flooding a server with many superfluous requests to prevent the server from responding to the valid requests promptly. A DoS attack can significantly harm the target that relies on a web presence. Distributed Denial of Service (DDoS) attack strikes a target from many sources and is harder to stop.

- **Ransomware**: Ransomware, which is a type of malware, restricts access to user’s files or a computer system until the victim pays a ransom. It can be considered a special kind of malware where it has been evaluated as a separate threat, even though it belongs to the malware category. The
2.2 Cyber threats

appearance of RaaS (Ransomware as a Service) allows cybercriminals to launch cyberattacks easily by providing a set of tools for less than $400 on the dark web.

- **Exploit Kits:** Exploit Kits are a collection of ready-made exploits designed to be modular and easy to use. The Kit gathers information on the victim machine, identifies vulnerabilities on programmes, such as web browsers or web applications, and determines the appropriate exploit to compromise them automatically. When an exploit is successful, the exploit kit can deliver a payload to infect the host. The payload can be a downloader which downloads another malware or the intended malware itself such as ransomware, botnet, and banking Trojan.

- **Cyber Espionage:** It is the act of obtaining confidential information without permission and knowledge from the holder of the information for political, military, economic, or personal objectives. It involves using the Internet or a computer network through using proxy servers, cracking techniques and malware including spyware and Trojan horses. The typical targets of this attack are government, political, and commercial organisations. Many spy attacks use Advanced Persistent Threats (APTs) as a method to gain unauthorised access to networks or systems covertly and remain there undetected for an extended period.

Cyberattacks on cyberspace are consistently evolving and growing through time. Cybercriminals benefit from the development of new tools and techniques to increase the number of attacks and the degree of harm caused to its victims.
2.2 Cyber threats

2.2.1 Network-based threats

Without appropriate security systems, any network or host can be susceptible to attacks or unauthorised activities. Switches, routers, and hosts can all be compromised by attackers or even internal employees. Network-based attacks can be varied according to the variety of penetration attempts, i.e., external and internal attacks, as shown in Fig. 2.2.

![Figure 2.2: Variety of attacks](image)

In general, there are four categories of threats to network security [2] [50]:

1. **Unstructured threats**: These threats consist of mostly inexperienced hackers using various available hacking tools, such as password crackers and shell scripts, to compromise a system. Although hackers in this type may have harmful intent, most of them are more interested in the intellectual challenge of cracking safeguards than maliciousness.

2. **Structured threats**: Structured threats: Hackers launch these threats with higher-level skills. These hackers can know system vulnerabilities,
2.2 Cyber threats

develop exploit codes and scripts, and use sophisticated hacking techniques to compromise systems. Sometimes, such hackers are hired by organised crime or industry competitors, and they are often involved in the large frauds that reported to law enforcement agencies.

3. **External threats:** These threats originate outside an enterprise or institution and can be structured and unstructured threats.

4. **Internal threats:** These threats originate inside an organisation and typically involve disgruntled former or current employees.

Increased and advanced connectivity defines the rapid growth of the Internet and subsequently the importance of network-based security. A large number of devices are now connected, which contributes to an increased need for security systems against such attackers [51].

Network security aims to provide confidentiality, integrity, availability, and non-repudiation for data that is either transmitted in networks or stored in networked machines. Network security is an essential part of information security [52]. Network-based IDS (NIDS) is installed on a network segment and operates as a stand-alone device. It can be used to detect computer attacks or illegitimate access and gives an alert about intrusion detection by monitoring the network traffic. IDSs use two main detection techniques: anomaly-based and signature-based intrusion detection [50].

NIDSs are typically installed in the perimeter of the security infrastructure of an organisation, and they focus on monitoring for external intrusion attempts. A typical NIDS includes several sensors to monitor packet traffic. The function of the sensor is to collect data from a system or a network segment and forward
it to the analyser part. The type of this data could be log files, network packets or system calls. There are several NIDS tools available such as Snort, Suricata, and Bro IDS [53]. Fig. 2.3 illustrates the possible placements of the sensors. Although network-based attacks have been around for decades, and there are many protective methods presented to overcome them, they are still causing numerous problems today [51].

![Figure 2.3: Examples of possible NIDS sensor locations [3]](image)

Some common network attacks are [51] [54]:

- **SYN attack:** It is a kind of DoS attack that exploits the incapability of the server to handle unfinished connection requests. This attack attempts to consume the server’s resources by flooding it with connection requests to make the system unresponsive to legitimate requests.

- **Eavesdropping:** It is an effective method to capture sensitive information. The purpose of eavesdropping is to make unauthorised interception of network communication and disclosure of exchanged information. It can
be carried out in different network layers using a network device and packet sniffer.

- **Buffer overflow**: Buffers are areas of memory used to store data temporarily. A buffer overflow occurs when the volume of data exceeds the buffer’s capacity, and as a result, the program overwrites adjacent memory locations. Attackers exploit such a situation to crash a system or insert specific code that allows them to get control of the system.

- **Password sniffing**: Password sniffers are small programs, used to listen to the traffic in the attached network and extract the login information such as usernames and passwords. Common network protocols such as Telnet and FTP often require users to enter their names and passwords for authentication, making them vulnerable for a sniffer attack to intercept the login information.

- **SQL injection**: It is a code injection technique used by attackers to attack websites via their backend database to steal data which was not intended to be displayed.

### 2.2.2 Domain Generation Algorithm (DGA)

Many cybercriminals depend on the C&C channels to carry out remotely several malicious activities, such as data exfiltration and sending instructions to malware. To protect these channels from being detected and blocked by security controls, attackers have started to use different communication techniques to build reliable C&C infrastructures to conduct their cyberattacks. One of the main C&C
communication techniques is DGA, which used by malware to generate numerous C&C server domains algorithmically [55]. DGA is a deterministic algorithm that can be used to generate many random domains so that the attacker should only register one or more of them in advance to enable the malware making a connection with its C&C server [56].

The generated domains of DGA can be utilised to allow the malware to locate its C&C server dynamically rather than relying on hardcoded IP addresses or domains. Also, these domains can be used as rendezvous points for collecting sensitive information from infected machines. The main advantage of using the DGA technique is that it allows for a large amount of redundancy in the C&C server. If the server is taken down, a new one can be available in a short time [57]. The DGA method enables the threat actors to change their C&C servers or locations periodically to make malware communications back to the C&C server in an automated fashion. Utilising a DGA would establish a game of cat and mouse for both security controls that mitigate the threat and the attackers that would change the location of the C&C server frequently [58].

DGA emerged as a stealth mechanism to communicate with a C&C server. After the malware installed successfully in the victim’s machine, the attackers can control the malware and instruct it to download sophisticated agents called remote access toolkit (RAT), which allows the attackers to have access to the system and do various harm activities stealthy [4].

DGAs generate a large set of pseudorandom domains dynamically using a precalculated seed value. The seed value is generated by embedded code in the malware binary. These pseudorandom domains are called algorithmically generated domains (AGDs). Fig. 2.4 shows the Python implementation code of Ramdo
DGA family, which authored using the logic extracted after reverse-engineering the Ramdo binary.

The attacker knows the seed value in advance and hence can generate the same set of AGDs. As a result, the malware author (attacker) can register one or more of these many AGDs to enable the malware forms a communication channel with its C&C server. When the malware is executed in the compromised machine, it runs the built-in DGA code to generate pseudorandom domains on the fly. After that, the malware queries these domains (AGDs) individually until one of them is resolved and use it in establishing a communication channel with the attacker-managed server. Fig. 2.5 illustrates the steps to carry out DGA.

### 2.2.2.1 Taxonomy of DGAs

The classification of DGAs depends on the seed value. The seed value is a set of parameters required for the execution of a DGA. It works as a shared secret key
2.2 Cyber threats

required for the calculation of generated domains. Typical parameters include numerical constants, such as the domain’s length, or strings. There are two important features of a seed that are used to characterise the DGAs [59] [60]:

- **Time dependence**: This feature indicates that a DGA incorporates a time source (e.g., the system time of the compromised host or the date) for calculation of AGDs. As a result, the generated domains will have a validity period only during the time of querying them by the infected system.

- **Determinism**: This feature addresses the availability of parameters. For most known DGAs, the parameters required to carry out DGA are known to the degree that all possible domains can be determined. Non-determinism property disallows arbitrary prediction of future AGDs by using unpredictable data for seeding.

According to the features above, four combinations of DGAs can be identified.
Table 2.1 shows the classes of DGAs.

<table>
<thead>
<tr>
<th>Type</th>
<th>Time dependence</th>
<th>Deterministic</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDD (time-dependent and deterministic)</td>
<td>✓</td>
<td>✓</td>
<td>Gameover Zeus, Conficker</td>
</tr>
<tr>
<td>TID (time-independent and deterministic)</td>
<td>❌</td>
<td>✓</td>
<td>TinyBanker, Kraken,</td>
</tr>
<tr>
<td>TDN (dependent and non-deterministic)</td>
<td>✓</td>
<td>❌</td>
<td>Bedep, Torpig,</td>
</tr>
<tr>
<td>TIN (time-independent and non-deterministic)</td>
<td>❌</td>
<td>❌</td>
<td></td>
</tr>
</tbody>
</table>

2.3 Malware threats

Malware is a program that gets into a victim’s system without user’s knowledge to compromise the confidentiality, integrity, or availability of the infected system, such as viruses, Trojans, botnets, worms, rootkits, and ransomwares. Sometimes malware may cause annoying activities to their victims without affecting the files or the operating system [61]. Nowadays, malware is considered as one of the most prevalent types of threats to cybersecurity [62], and many cybersecurity experts have considered malware as the key choice of weapon to run various malicious actions to breach cybersecurity [18]. The malware threat becomes enormous due to the increased sophistication of malicious software and growing the vulnerabilities that can be utilised by the cybercriminals. Nowadays, the protection of networks or computer systems from malware threats is one of the essential concerns of information security for individuals and organisations [62].

Although many research works have been presented on malware detection and mitigation technique, malware reigns at number one as the most widespread threat to businesses, organisations, and individuals. The unceasing war between malware authors and the security vendors is driving innovations on both sides. It is a war limited by economics rather than attrition [63].
Malware families are rapidly evolving and have become more sophisticated, dangerous, and targeted. Fig. 2.6 shows a comparison between traditional and advanced malwares [64].

![Figure 2.6: Traditional vs. advanced malwares](image)

In comparison to traditional malware, advanced malware attempt to contact a C&C server for further instructions. Formation and use of the C&C system is an integral part of cyberattacks carried out remotely. C&C servers are used to instruct compromised machines to perform malicious activities, such as infect other machines and spy, or the infrastructure of C&C can be also used as a covert channel over which data can be exfiltrated [65]. In the cybersecurity community, the development of more advanced and efficient malware protection mechanisms has been considered as an urgent necessity [18].

The main malware barrier has moved from host protection to network detection over the few last decades. Although the host-based defences provide the most advanced suite of anti-malware systems and clean up capabilities, they are diffi-
2.3 Malware threats
cult to manage and often easy to circumvent [63]. The network-based detection
system of malware is based upon network artefacts and communication features
generated by malware such as contact with the C&C server. Cybercriminals use
various ways to infect the victims with malware and get unauthorised access to
a network or computer system that grants remote access to a C&C server. The
typical infection vectors used by the attackers to infect the targets are [66] [67]:

- **Email:** It is an infection vector used by cybercriminals to distribute mal-
  ware. Email attacks can exploit vulnerabilities in the email software or
  in the libraries that the email software uses. When the email software
downloads a message and displays it, an embedded code can exploit a vul-
  nerability and execute the malicious code.

- **Phishing:** It is the practice of sending emails appearing to be from trusted
  sources to gain personal information or damage the victim’s system. This
deadly email aims to lure the user into opening an attachment that con-
tains malware or clicking a link to a malicious website. Today, phishing
is considered as one of the biggest cyber threats facing organisations and
individuals.

- **Drive-by Download:** A drive-by download means the unintentional down-
  load of malicious code (malware) onto the victim’s machine without a user’s
  knowledge that leaves the user open to a cyberattack. A drive-by download
usually exploits a browser, an application, or an operating system that has
a vulnerability due to unsuccessful updates or lack of updates. Unlike other
types of cyberattack, a drive-by download does not need activity from the
user to actively enable the attack.
2.3 Malware threats

2.3.1 Taxonomy of malware

Malware can be classified depending on several criteria, such as the propagation method and activities performed on the victim’s device. There are also some characteristics of malwares that can be used to classify them. In general, malware types can be classified according to the following characteristics [3] [68] [69]:

- **Propagation method**: It is how the malware spreads to reach the desired targets (pre-breach), or subsequently propagate to other machines or networks. While early malware used a single means of propagation to deliver a single payload, modern malware uses multiple methods of infection to maximise the speed of infection and the severity of the attack, which is called a blended attack. Many malwares depend on a particular medium to penetrate devices and hence based on this property, computer malware can be classified as either system-based or network-based.

- **Payload**: The variety of actions carried out by malware once it reaches a target is called a payload, such as a system corruption, stealing sensitive information, or making the machine as a part of a botnet to launch attacks on other victims.

- **Self-replicating**: It is the malware ability to create new instances of itself.

- **Parasitic (dependent)**: A dependent malware requires some other executable code to run.

Fig. 2.7 shows malware taxonomy basing on some common characteristics of malware.

Some types of malware are:
2.3 Malware threats

- **Virus:** A virus is a program or form of code that can infect executable files by inserting a malicious code within the original code of a benign executable file. Generally, the virus consists of two parts: insertion code and payload code. The virus is self-replicating malware and uses different methods to spread its malicious code to the victims, such as an email attachment and getting embedded into a removable media like USB sticks and optical disks.

- **Worm:** A malware that can run independently and replicate itself to infect other hosts on a network without the need for user intervention. Each infected machine works as an automated launching node to attack neighbour machines. Worms usually exploit software vulnerabilities in host programs to get access to the target system.

- **Trojan horse:** It is a program that appears useful, but it contains a hidden code which performs malicious actions when is executed. Trojan malware
can be used to perform some actions indirectly when the attacker could not run them directly, such as stealing sensitive or banking information from the victim and establishing remote access to enable the attacker accessing the victim’s device covertly. Modern Trojan horses use worms and viruses to get access to target machines.

- **Rootkit:** It is an automated set of tools that is installed on the victim’s system and can be employed for several objectives, such as getting administrative privileges on the victim’s device, concealing intrusion, and sniffing information about a computer or network. The rootkit can access all the system functions and modify them in a harmful and covert way. It is difficult for the user to discover the rootkit activities due to the several changes and modifications that happened on the system by the rootkit to hide its existence.

- **Botnet:** A bot, a shortcut of a robot, is a kind of advanced malware that is implanted into a compromised computer to be controlled by a botmaster remotely to execute malicious commands. A botmaster is a person or a group who creates the botnet and controls the bots. Botnets are responsible for launching various sophisticated attacks, such as the Distributed Denial-of-Service (DDoS) and email spams.

- **Spyware:** It infiltrates the device and steals sensitive data or spies on personal information without the user’s knowledge by monitoring screen data, keystrokes, and network traffic. Spyware runs in the background and sends the collected data to adversaries over the Internet.
2.3 Malware threats

- **Backdoor**: It is a type of Trojan horses that allows cybercriminals to get unauthorised access to compromised systems, networks, or applications covertly. Cybercriminals use backdoors to run various malicious actions, such as stealing personal and financial data and installing additional malware.

- **Ransomware**: It is a category of advanced malware that locks computer system or restricts access to the user’s files until a specific ransom has paid.

- **Logic bomb**: It is a piece of code that deliberately inserted into a software system. It triggers a malicious action when a specified condition is met.

2.3.2 Malware analysis techniques

Malware analysis is the action of extracting information from the malware by dissecting it using different tools and techniques [70]. Malware analysis is considered a major part in an incident response plan through which we can get the necessary information to be able to respond to an incident or intrusion. It is also used to develop host-based and network-based signatures. The purpose of the malware analysis should be set before performing the analysis and can be as:

- Know how the malware works.
- Know how to identify malware.
- Know how to detect the malware in a system or on a network.
- Know how to defeat or eliminate malware.
There are two essential methods of malware analysis, static and dynamic analysis. A combination of these two techniques can be used and is called a hybrid analysis. Each technique has specific tools and can provide different information for the malware analyst [71] [72].

### 2.3.2.1 Static analysis

It is the process of analysing the code or the structure of malware to extract the information without executing it. There are no special requirements to perform the analysis, so it is a quick method. Also, it is considered a safe (less risky) approach as it does not require the malware to run during the analysis and can be accomplished by several operating system native tools. Various information can be collected from the static analysis, such as file type, strings, and imported functions (DLLs). Static analysis may fail with advanced and sophisticated malware, which use sophisticated techniques such as obfuscation and packing, and may not provide vital information about malware behaviour. Different tools are used in this analysis, such as PEiD, PEview, Strings, and disassemblers [71].

### 2.3.2.2 Dynamic analysis

It is the process of analysing malware actions while it is being executed. Unlike the static analysis, the dynamic analysis provides a more in-depth analysis of malware behaviour and operation because the information is collected while the malware is running its functions and directives. Two essential things are required to perform the dynamic analysis:

- Malware testbed
2.4 Ransomware

Dynamic analysis tools

This approach is more complicated than the static analysis because it requires a safe environment and needs a connection to the Internet for some types of malware to operate and behave appropriately. Implementing the dynamic analysis puts the computer or the network at risk due to the possibility of malware spreading while it is running [73].

2.4 Ransomware

Ransomware is a kind of advanced malware that makes a victim’s machine or data unusable, and it is increasingly being used by attackers to achieve revenue through extortion of victims [74]. The first appearance of ransomware program was in 1989, which was created by Joseph Popp and called ‘AIDS’ (PC Cyborg) where was deployed as a Trojan. The AIDS Trojan was propagated using floppy discs. When inserting the floppy disc, the AIDS program, i.e., ransomware, encrypts the files on the hard disk and then demand payment of $189 to a post office box in Panama [9].

Depending on the study and analysis of the ransomware attack, five distinct phases can be recognised. Understanding the events and activities that can appear in each phase may help to build an efficient system to detect and prevent or mitigate the attack. The following points illustrate the anatomy of the ransomware attack [75] [76]:

- **Deployment** (Infection). In this phase, the ransomware is spread and delivered the malicious files to the victims. It uses common attack vectors that employed by other types of malware, such as:
2.4 Ransomware

- Phishing
- Drive-by- download
- Vulnerability Exploitation

**Installation.** The ransomware payload is executed on the infected system during this phase. A persistence technique is implemented here to ensure that the ransomware starts up after making a reboot for the computer system.

**Communications with C&C server.** After the deployment and installation phase, most of the ransomware families require establishing some forms of communications channels with their C&C servers. The ransomware can remain dormant in the victim’s system until receiving the instructions from the attacker server.

**Destruction.** The payload of the ransomware is executed in this stage. The sophisticated families of Crypto ransomware generate a symmetric key inside the victim’s system, which used to encrypt and decrypt the user’s files. These files could be specified by the ransomware binary using a hard-coded list of files extensions or identified by C&C server. Some ransomware variants encrypt the filenames in addition to their content to render the user unable to identify the encrypted files. Besides, some families of ransomware can lock the victim’s system and make the user unable to use the system until a specific ransom has been paid.

**Payment.** The demand instructions for extortion and payment method are presented in this stage. The victims usually receive a ransom message
that asked them to pay within a few days; otherwise, they will lose their files permanently. The attackers often ask the victims to pay in Bitcoin, which is a cryptocurrency used to conceal the attacker’s information.

2.4.1 Ransomware taxonomy

Ransomware can be classified into two main types based on the means of extortion, i.e., whether the encryption is employed against the user’s files or not. The first type is crypto ransomware, which is a cryptographic type that encrypts the victim’s files. The second one is locker ransomware, which is a non-cryptographic type that locks the victim’s computer and prevents users from accessing the system [77].

Several factors can be used to categorise the ransomware, such as severity, type of targeted victims, and type of system affected. Fig. 2.8 shows a generic ransomware taxonomy based on some perspectives.

The most common and aggressive type is crypto ransomware that is adopted in this research as a case study. Crypto ransomware is not only capable of encrypting users’ files, but it also attempts to encrypt any file located on both mapped and unmapped network drives [78], bringing a department or the entire organisation to a halt if one system is infected. Generally, the crypto ransomware follows several typical steps to achieve its goals, as shown in Fig. 2.9. Crypto ransomware can be divided into three types based on the type of cryptosystem used, as indicated in the subsections below.
2.4 Ransomware

Figure 2.8: Ransomware taxonomy based on various factors [5]

Figure 2.9: Crypto ransomware lifecycle [6]
2.4.1.1 Symmetric Cryptosystem Ransomware (SCR)

Symmetric Cryptosystem Ransomware (SCR) employs symmetric encryption algorithms, such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES), to encrypt the victim’s files. The same key, which is called a secret or shared key, is used for both encryption and decryption processes. Although SCR carries out the attack faster than other types, the victim may recover the secret key by applying reverse engineering or memory scanning techniques [79]. Trojan.Pgpcoder is an example of this type [5]. Fig. 2.10(a) illustrates the network activities involved in this type of ransomware.

2.4.1.2 Asymmetric Cryptosystem Ransomware (ACR)

Asymmetric Cryptosystem Ransomware (ACR) employs asymmetric cryptography algorithms such as RSA. In this type, a pair of keys are used such that a public key is used for encryption, i.e., encrypt the victim’s file, while a private key is used for decryption [80]. The public key is embedded within the ransomware file or downloaded during the communication with the C&C server. As the private key is kept only with the attacker, the victim cannot obtain it without paying the ransom. The private key might be exposed if one victim obtained and shared it with other victims. To prevent such a situation, ransomware authors generate a list of asymmetric keys, which makes sharing these with all victims very difficult. However, this technique consumes more resources while encrypting the files [5]. The corresponding network communications are depicted in Fig. 2.10(b).
2.4 Ransomware

Figure 2.10: Crypto ransomware network communications [7] (a) Symmetrical encryption, (b) Asymmetrical encryption

2.4.1.3 Hybrid Cryptosystem Ransomware (HCR)

The ransomware authors have integrated symmetric and asymmetric encryption techniques to generate a hybrid type called Hybrid Cryptosystem Ransomware (HCR). HCR is a crucial part for adversaries to carry out unbreakable attacks as it provides quick and robust encryption [5]. Most modern families of crypto ransomware use the HCR technique to take advantage of both types of encryption. HCR uses a dynamically generated symmetric key to encrypt the victim’s files, and a pre-loaded public key to encrypt this symmetric key itself after clearing it out of memory. Then, the victim will be asked through the display message to send the encrypted symmetric key, together with payment, back to the C&C server. When the victim pays the ransom, the cybercriminal extracts the symmetric key and sends it to the victim to enable the user to regain access to the files [81]. Crypto Locker, which generates an RSA 2048 bit public key with the
AES symmetric key, is an example of HCR. Crypto Locker uses AES to encrypt the user files, then the file that contains the encryption key of AES is encrypted by the public key of RSA, as shown in Fig. 2.11. Therefore, the user needs the private key of RSA, which is kept only at the attacker side, to decrypt the file containing the AES key and restore this key.

![Diagram of Crypto Locker and C&C server communications](image)

**Figure 2.11:** The communications between Crypto Locker and C&C server [8]

### 2.4.2 Network-based ransomware activities

Ransomware compromises Internet users by encrypting user’s most important files and then demanding payment in exchange for the decryption key to restore the files. Organisations can minimise the damage of malware if they can detect them early. Some security controls detect ransomware based on its activities in the host, such as file system and registry activities [82]. Some of the research works were interested in discovering the network behaviour of ransomware and building network-based detection systems [25].
The attacker has developed a complex infrastructure to support the ransomware attack as depicted in Fig. 2.12. The sequence numbers annotated in Fig. 2.12 is typical for many families of ransomware and attack campaigns. It is noticeable from Fig. 2.12 that ransomware generates several network activities during the campaign that can be divided into four stages according to the attack landscape:

- Deployment (delivering the ransomware binaries)
- The communication with C&C servers (exchanging the encryption keys and data)
- Propagation across the networks
- The method of payment (cyber currency)

Nowadays, with a massive increase in the number of Internet users, the attackers have chosen the Internet environment, which provides an efficient and cost-effective delivery method, to distribute their malware. The attack starts by deploying the ransomware files and attempting to trick the users and infect the computers. After delivering the malicious files for the victim’s computer, the infection begins, and the payload is carried out. The most common ransomware propagation methods are email (malicious attachments and embedded links), social engineering, and malicious web sites [31].

The attempts of ransomware to establish contact with a C&C server is considered a significant network activity to detect the attack. Ransomware communicates with the C&C server to retrieve necessary encryption keys to perform the data encryption or receive instructions through covert channels to execute
various malicious actions. The instructions can be anything, such as identifying the files’ type to be encrypted, how long should the ransomware wait before it begins the attack, and whether the ransomware should continue to spread over the networks [83].

The attackers use various techniques to perform their C&C communications with the victim’s computer. These communication channels can be ranged from simple (uncomplicated) systems, such as static (hardcoded) IP addresses and DNS domains, and HTTP protocol, to complex systems that are difficult to detect by security controls, such as DGA, and Tor services [84].

The Domain Name Service (DNS) is a hierarchical naming system for computers connected to the Internet. Attackers widely use DNS to establish and operate the C&C channels. DGA technique has been used by many families of
ransomware to periodically generate numerous domain names that can be employed as rendezvous points with the C&C servers. It is a resilient technique to evade the shutdown of attackers’ servers.

Tor (The Onion Router) is a service used to provide anonymity over the Internet. Tor works by relaying Internet traffic through several nodes and applies traffic encapsulation, which is multiple levels of encryption, to hide the source and destination nodes of the traffic [85]. Tor is a complex system that makes more difficult to trace the exact location of the attackers participating in the ransomware attack. Some variants of the ransomware install Tor clients on the victims’ systems to ensure they have secure communications. The introduction of the Tor system and the economic growth of the dark web led to the adoption of RaaS (Ransomware as a service). RaaS is a method that enables skilled hackers to offer their services to others who do not have enough skills or infrastructure to launch a ransomware attack [77]. Also, Tor has been used by the attackers in the ransom payment through a Tor-based Bitcoin to ensure improved anonymity [9].

2.4.2.1 Network-based ransomware detection techniques

The research community has presented different detection techniques in recent years. These techniques depend on differentiating ransomware action from benign software to detect the ransomware attack. As illustrated in previous sections, the ransomware attack consists of several stages and generates several network activities in some of them. The network activities of ransomware have been utilised to develop different detection techniques basing on performing a comprehensive analysis of network traffic for unique features, which is a vital step in designing an effective detection algorithm [86].
Malware network characteristics can be extracted from inspecting the ransomware binary code without running it, or from analysing the network traffic after running the binary code. These characteristics are used as input data to the proposed detection systems for the traffic classification as benign or malicious. The prevalent detection techniques used to identify the ransomware network activities are outlined below:

A- Machine learning

Machine learning (ML) involves learning the patterns in data to build a model. This model can predict the outcome of new data that have not been seen during the learning process. Many of the ML-based malware detection systems use the supervised learning technique that requires a labelled dataset [29]. The ML-based systems can accurately predict the outcome with an adequate and balanced training dataset. They are also less susceptible to obfuscation because machine learning involves learning the pattern within the data. However, finding the appropriate algorithm is not an easy task and may require some runs of trial and error method.

B- Honeypot

Honeypot involves setting up decoy files in decoy computers deployed by network administrators to detect ransomware activity. Once these files are accessed, the ransomware attack can be detected. The honeypot looks like a real target, with applications and data to fool cybercriminals. The intrusion attempts are used to gain information about the ransomware attack and to raise an alert of an attack [87]. Honeypot technique can attract attackers and does not require much maintenance or processing power from the system. However, there is no guarantee that the honeypot files will be attacked by the ransomware. Hence, it
2.4 Ransomware

is significant to know the characteristics of the files that the ransomware intends to target [75].

C- Software-Defined Networking

Software-Defined Networking (SDN) is an attractive and emerging method for networking management. SDN can overcome the limitations of the current network infrastructures through enabling dynamic and programmatically efficient network configuration to enhance network performance and monitoring. Fig. 2.13 shows the simplified view of an SDN architecture. SDN applications have been used to block the communications between ransomware and the attackers. The proposed systems were employed to identify the suspicious activities of ransomware through monitoring the network traffic. After that, the infected hosts are blocked by applying appropriate control rules in a real-time and affecting how network devices handle the traffic [7].

Figure 2.13: View of an SDN architecture [10]
2.5 Network-based malware detection techniques

Conventional malware detection methods are carried out at client computers that target malware activities at compromised machines. These approaches are referred to as host-based detection methods, and they usually rely on the signatures of malware. Host-based methods can also analyse various activities of the host, such as running processes and logon actions. Besides, they can monitor traffic that is originating and coming to the network interface of a particular host only without able to detect attacks in any other part of the network [88] [89].

Advanced malware depends on Internet-based communications to increase its severity and carry out various actions, such as spreading across networks and communicating with the attacker, i.e., C&C server [90] [91]. Network activities generated by malware are essential indicators, and they are considered as one of the essential means for malware detection. Network-based detection systems rely on analysing network traffic to detect malware attacks and identify compromised machines [92] [93].

Typically, network-based detection approaches can be classified into two main types [35] [94] [95]:

- Signature-based detection

- Anomaly (behaviour)-based detection

There are also two detection techniques derived from the main methods above [94] [96] [97]:

- Specification-based detection (stateful protocol analysis)
2.5 Network-based malware detection techniques

- Heuristic-based detection technique

These types use one of the three distinctive analysis methods: static, dynamic or hybrid to extract the necessary information (features) required to detect malware. Fig. 2.14 shows the classification of malware detection techniques.

![Figure 2.14: Taxonomy of malware detection techniques](image)

2.5.1 Signature-based detection

A signature is defined as a small unique sequence or a pattern of bytes which is used to identify a known malware with a high detection rate [11]. A signature-based detection uses these patterns or heuristics (rules) to detect malware and identify its type by comparing the program under inspection against known malware signatures. The mechanism of signature-based detection technique is illustrated in Fig. 2.15. To create a signature for unknown malware, the anti-malware vendors need to get a sample of this malware to analyse it and extract the new signature, which means that this technique is not efficient (i.e., has a low detection rate) for detecting new malwares [94].
2.5 Network-based malware detection techniques

Figure 2.15: The traditional mechanism of signature-based malware detection [11]

### 2.5.2 Heuristic-based detection technique

Since modern malwares can easily bypass the signature-based detection using sophisticated techniques, such as encryption, obfuscation, packing, polymorphism and metamorphism, the heuristic-based technique has been introduced. This technique is based on rules determined by the experts to differentiate between malware and benign software that characterise known malicious behaviour [11] [97]. It uses ML and data mining methods to identify the behaviour of the executable programs [98]. It has some limitations, such as its false-positive rate is more than the signature-based, the need for more storage space [99], and it could not operate well in real-time applications due to the high computational complexity [35].
2.5 Network-based malware detection techniques

2.5.3 Anomaly-based detection

Anomaly-based detection is used to detect anomalies that deviate from the normal behaviour by comparing the current runtime behaviour with a predefined normal one. If the inspected behaviour does not match with predefined behaviour, anomaly alarm will be triggered by the malware detector [100]. Anomaly-based methods are different than signature-based detection as they do not have any knowledge of the attack patterns. Software packages that have similar behaviour are collected in this approach to build a behavioural model which is used to detect variant samples (mutants) of malwares. The main difference between signature-based and anomaly-based technique is that the anomaly can detect unknown malware, i.e., zero-day attacks [101].

2.5.4 Specification-based detection

Specification-based (also called as stateful protocol analysis) is a technique derived from the anomaly-based detection to overcome the high rate of false alarm associated with it. It depends on the standard characteristics of network protocols to trace their states and discriminates unusual sequences of commands. The main disadvantages of this technique are the difficulty of detecting attacks that mimic the normal protocol behaviour and the high resource consumption when tracing and inspecting protocols states [35] [94].
2.6 Machine learning-based detection

Machine Learning (ML) reorganised as a separate field, perhaps in the 1990s, when scientists and researchers started giving it more importance as a subfield of AI (Artificial Intelligence). ML techniques focus on methods and models borrowed from statistics and probability theory that give excellent performance [102]. ML technique is a subfield of AI that builds systems capable of learning and improvement from experience automatically without being explicitly programmed [103].

ML has evolved from the pattern recognition study and computational learning theory in AI and borrows some concepts from statistics and probability science. It often overlaps with computational statistics and closely related to mathematical optimisation. ML algorithms use computational approaches to learn information or extract patterns directly from training data without depending on a predetermined formula. Fig. 2.16 illustrates the basic concepts of the ML technique. It is noticeable that the machine uses an appropriate learning algorithm and training data to train and build a model. This model will be employed later to predict outcomes for new data (i.e., unseen data during the training process).

ML process lifecycle can be summarised as:

- Define the problem statement, which includes the goal and assumptions.
- Determine the type of problem, for example, a classification or regression problem.
- Collect a suitable dataset.
- Data analysis and features extraction.
2.6 Machine learning-based detection

- Build a set of models using a training dataset and some appropriate learning algorithms.

- Evaluate the models using an unseen testing dataset and some proper evaluation metrics.

- Enhance the models (optimisation) if required for that.

Fig. 2.17 illustrates the flow and architecture of the underlying ML system.

There are multiple technical and functional definitions for ML; however, I selected one, which describes the basics of ML clearly, and it states as “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”

It is observable that the T, P, and E are the main components of any ML model as illustrated in Fig. 2.18 [13] [102]. Where ML algorithms make computers learn
2.6 Machine learning-based detection

automatically and improve their performance (P) over time with experience (E) at executing some tasks (T).

Conventional programming paradigms essentially involve the programmer or user writing a set of instructions using code which makes the machine carry out some specific operations or computations on the data to produce desired results. While the ML paradigm attempts to take data and predicted outputs
into account if any, and it uses a learning algorithm to build a model. For the future, this system or model will then be used to make the necessary decisions and predict the output for new inputs basing on the learned knowledge or experience from previous data points. Fig. 2.19 depicts a typical workflow for conventional programming paradigm and ML paradigm [102].

![Figure 2.19: A typical workflow (a) conventional programming paradigm (b) ML paradigm](image)

### 2.6.1 Need for machine learning

In general, the ML technique is used for two main aspects. The first aspect is the complex problems that relate to the analysis of large and complex datasets, such as medical issues, intrusion detection, and web search engines. The second one is the dynamic tasks that change over time. The behaviour of ML models adapts to their input data and gives an appropriate solution to such issues [16].

The main advantages of using ML are [102] [13]:

- **Increase in data size:** Due to the growing vast volumes of data nowadays,
2.6 Machine learning-based detection

ML is an effective tool to analyse and draw beneficial insights from data.

- **Improve decision making**: ML can be used to make the right decisions by employing various learning algorithms.

- **Pattern recognition**: Recognising hidden patterns and infer key insights from a dataset is an essential part of using the ML technique.

- **Solving complex problems**: ML has been used effectively to solve real complex problems in different areas such as cybersecurity, medical diagnosis, computer vision, and image processing.

Fig. 2.20 shows the main goals of using ML.

![Figure 2.20: The importance of ML](image)

ML techniques have been extensively used in host-based and network-based intrusion detection systems [104]. Nowadays, ML techniques are commonly employed to detect malware activities. These techniques not only can detect the known malware, but they can also detect the unknown malware by acquiring knowledge from the previously discovered malware [105] [106].
One of the significant aims of using the ML approach is the possibility of automated operation to infer knowledge about malware network traffic and detect its activities from a large amount of network traffic. ML-based intrusion detection models have a high detection rate and can adjust themselves in response to the traffic passing through a network. Besides, ML techniques, such as artificial neural network, can generalise the model from limited data [104] [107].

### 2.6.2 Machine learning methods

ML has several algorithms, techniques and methodologies that can be used to build models to solve real-world problems using valid datasets. ML methods typically can be classified according to several broad areas. ML methods based on the amount of human supervision in the learning process can be classified into [102] [16]:

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning

Supervised learning methods take input data samples of a training data and associated outputs (also known as labels or responses) with each input sample during the model training phase. Supervised learning methods use the samples of input data and associated outputs (also known as labels or responses) during the model training phase. This learning type attempts to model the relationship
between the input samples and their corresponding labels depending on multiple training data instances, where a learning algorithm is used to learn the mapping function \( f(x) \) from the input \( (x) \) to the output \( (Y) \). This function is also called a hypothesis, a predictor, or a classifier.

\[
Y = f(x)
\] (2.1)

Supervised learning aims to approximate the mapping function \( f : X \rightarrow Y \), where \( X \) is a set of features (columns) matrix of dimensions \( N \) (number of samples) by \( D \) (feature dimensions), \( Y \) is the set of possible labels of dimension \( N \times 1 \) indicating the class, i.e., label, of each input sample, \( S = ((x_1, y_1), \ldots, (x_m, y_m)) \) is a finite sequence of pairs in \( X \times Y \) which represents training data set, and \( f \) is the multi-class classifier algorithm that assigns each input sample to a class [108] [109].

Several well-known algorithms, such as Decision Trees (DT), Support Vector Machine (SVM), Naïve Bayes (NB), and K-Nearest Neighbour (KNN), are commonly used for modelling classification based tasks [110]. The working of Supervised learning is depicted in Fig. 2.21

Supervised learning methods usually require labelled training dataset, i.e., the associated outputs should be available, to build the models. However, the labelled datasets often are not available, and the need to extract useful patterns from the data still exist. In this case, unsupervised learning methods are beneficial and effective. The algorithms of these methods try to learn patterns and draw inferences from a given dataset consisting of input data only without responses (labels). Although there is more uncertainty in the results of unsupervised methods, much information from these models can be gained from the raw
The approaches that fall between supervised and unsupervised learning methods are called semi-supervised methods. These methods combine a small amount of pre-labelled data (the supervised learning part) and a large amount of unlabelled training data (the unsupervised learning part) [111].

The reinforcement learning methods are a little different from traditional supervised or unsupervised methods. In this method, an agent takes a particular action in an environment based on a rule or policy through observing the current state of the environment. Then this action is interpreted into a reward, which could be beneficial or harmful, and is fed back into the agent [102].

2.6.3 Classification algorithms

The classification tasks are a sub-field under supervised Machine Learning and one of the essential tasks of machine learning. The goal of the classification concept is to predict output labels (classes) that are categorical for input data depending on

Figure 2.21: Supervised learning method [15]

data [102].
what the model has learned in the training process. Each output response belongs to a specific distinct class. Classification can be binary, i.e., classification with only two distinct labels or multi-class classification, i.e., classification with more than two labels. Below is a description of the prevalent classification algorithms used with malware detection.

2.6.3.1 Decision Tree

Decision Tree (DT) is one of the robust and widely used modelling techniques in the ML field. Decision trees induce rules that can be used in data prediction and classification. DT classifies instances by representing data in a tree structure starting from the root to a leaf. There are two representations of DT: a node and branch that connects nodes. An example of a decision tree is shown in Fig. 2.22.

![Decision Tree](image)

Figure 2.22: Decision tree formulation [16]

The flow starts at the root node, and then it is passed through decision nodes that need choices to be made based on the features of the task. These choices
2.6 Machine learning-based detection

split the data across branches that indicate potential outcomes of a decision. The tree is terminated by leaf nodes (terminal nodes) that denote the action to be taken as the result of the series of decisions to make the final decision.

A decision tree is a predictor, \( h : X \to Y \), which predicts the label \( Y \) associated with an instance \( x \) by travelling from a root node of a tree to a leaf (which contains a specific label).

Three different ways are used to identify the best-suited features [13] [16]:

- Information Gain: Entropy is useful to measure the uncertainty in the data. The information gain is the difference between the entropy of the label before and after the split. A mathematical representation of entropy in the system before splitting is shown here:

\[
E = -\sum_{i=1}^{n} p_i \log p_i
\]

(2.2)

where \( p_i \) is probability distribution \( p_n = (p_1, ..., p_n \) with \( p_i \geq 0 \) for \( i = 1, ..., n \) and \( \sum_{i=1}^{n} p_i = 1 \)

The information gain concerning a feature \( f \) is computed as the difference between the entropy in a segment of data before the split (S1) and the partitions resulting from the split (S2):

\[
InfoGain(f) = Entropy(S1) - Entropy(S2)
\]

- Gini index: It is used to calculate the amount of probability of two random items belonging to the same class. Gini index can be expressed mathematically as:
2.6 Machine learning-based detection

\[ Gini \text{ index} = 1 - \sum_{i=1}^{n} (p_i)^2 \]  

(2.3)

Where \( p_i \) denotes the probability of an element being classified for a distinct class.

- Gain ratio: It is the ratio of information gain and information content. The feature that gives the maximum amount of gain ratio is the feature that is used to split it.

2.6.3.2 Nearest Neighbours

K-Nearest Neighbour (KNN) algorithm is simple and easy to implement a supervised machine learning algorithm that can be used for classification and regression problems. The idea of KNN is to memorise the training set and then to predict the label of any new instance based on the labels of its closest neighbours in the training set. KNN assumes that similar things are near to each other [112].

The value of K, which indicates the number of nearest neighbours, has a big effect on the performance of KNN algorithm. When the values of K are large, the KNN algorithm uses the previous value and may lead to an inaccurate result. At the same time, the too-small values of K might make the model sensitive to outliers. Hence, to choose an accurate value of K, the value should lie in the middle range between the smallest and largest values, so that it reduces the error and can give the best generalisation performance.

There are several distance measures used to measure the distance or similarity between the tested examples and the training examples. The distance measure determines the performance of KNN similar to the K value. Below the mathe-
mathematical formula of some distance measures [13]:

- **Euclidean distance**: It is the default option for numeric attributes and treats all the dimensions equally. The formula of the Euclidean measure is as follows:

\[
D(x, x') = \sqrt{\sum (x_d - x'_d)^2}
\] (2.4)

where \(x_d\) and \(x'_d\) are two vectors of training examples and tested examples respectively.

- **Hamming distance**: It is the default option for categorical attributes. Hamming distance is used to check whether the two attributes are equal or not. The distance is 0 when the attributes are equal; otherwise, it will be 1. The formula of the Hamming measure is as follows:

\[
D(x, x') = \sum 1_{x_d \neq x'_d}
\] (2.5)

- **Minkowski distance**: It is three distance metrics depends on different values of \(\rho\) for this power distance that are special cases of Minkowski distance. The formula of the Minkowski measure is as follows:

\[
D(x, x') = \rho \sqrt[\rho]{\sum (x_d - x'_d)^2}
\] (2.6)

Where \(\rho\) is the order parameter which permits different distance measures to be calculated:
2.6 Machine learning-based detection

If $\rho = 0$, the distance measure is the Hamming measure.

If $\rho = 1$, the distance measure is the Manhattan measure.

If $\rho = 2$, the distance measure is the Euclidean measure.

2.6.3.3 Support Vector Machines (SVM)

SVM (Support Vector Machines) is a learning algorithm used for classification and regression problems. The goal of SVM is to find an effective hyperplane (linear decision boundaries) that separates the classes, i.e., assembles items that have similar feature values, into groups [113]. The hyperplane can be defined as a generalisation of a line in 2-Dimensions and a plane in 3-Dimensions. The choice of the best hyperplane is determined by the largest separation or margin between the classes. The distance or space between the hyperplane and the closest point in the classification is called margin. The diagram in Fig. 2.23 represents data with two features $X$ and $Y$ and available classes being stars and triangles. Taking two hyperplanes and check the margins represented by $M_1$ and $M_2$. It is noticed that margin $M_1 > M_2$, so the choice of the hyperplane that separates the classes well is the new plane between the green and blue planes.

A linear equation can be used to represent the new plane as follows:

$$f(x) = ax + b \quad (2.7)$$

Defining the kernel mapping function is considered the inherent problem in SVM. Trial and error method is usually used to determine the best kernel function.
2.7 Summary

In this chapter relevant areas of research related to the undertaken research work in this thesis are introduced and briefly described. The chapter has presented the basic principles of how malware operates, type of malware families, and in more depth the features of advanced malware and ransomware. In more detail, ransomware and the ransomware attack landscape are introduced, and advance host and network-based malware detection methodologies and technologies are outlined. As the underpinning technology for malware detection, various ML algorithms and ML based classification techniques are discussed, targeting ML-based detection and classification of malware-related anomalies and threats. Relevant, state-of-the-art published research by peer groups and researchers have not been discussed. The following research chapters three, four and five, have dedicated related work sections for discussing in depth chapter-specific relevant research outputs published by peers.
Chapter 3

A Multi-Classifier ML Approach for Network-Based Crypto Ransomware Detection

3.1 Introduction

Crypto ransomware is the most virulent and aggressive type of malware as it uses cryptography techniques that have caused significant financial losses to diverse victims, as indicated in the previous chapter. As most families of crypto ransomware attempt to make a connection to a C&C server after the target is infected and before executing their malicious payload; therefore, a careful analysis of network traffic is required, as it can yield valuable results in detecting the ransomware attack.

This chapter presents a thorough behavioural analysis of crypto ransomware network activities, taking Locky, one of the prevalent families, as a case study. For
3.2 Related work

This purpose, a dedicated testbed is developed, and network features are extracted and classified into multiple new types. Moreover, a multi-classifier network-based intrusion detection system [36] is proposed that uses two independent classifiers operating in parallel, one on packet level and the second on flow level.

3.2 Related work

While several detection methods have been presented in the literature, state-of-the-art research clearly highlights that malware detection methods have been pushed to its limits due to the rapid evolution of ransomware attacks as outlined in chapters 1 and 2.

This section explores the academic state-of-the-art research, which has focused on the analysis and detection of ransomware activities based on network traffic. Cabaj et al. [114] employed the dynamic analysis method along with honeypot technology to analyse the network traffic of CryptoWall ransomware and to detect the infection chain. They used an automatic run-time malware analytical system called Maltester for dynamic malware analysis in a custom environment. In this research, they investigated six samples of CryptoWall. They identified some of the network activities of CryptoWall and presented practical results, i.e., identifying some proxy servers, the protocols used, and the hardcoded addresses of some servers. The authors used four virtual machines running on one physical server to perform the analysis. Furthermore, provided analysis is confined to the extraction of numerical values that are distinct to the CryptoWall family. The conducted analysis did not extract any generic behavioural feature that can be employed to the detection of other types of crypto ransomware.
Ahmadian et al. [79] categorised ransomware based on the type of encryption employed against user files into two basic classes: Non-Cryptographic Ransomware (NCR) and Cryptographic Ransomware (CGR). The latter class was further subdivided into three subtypes, called Private-key Cryptosystem Ransomware (PrCR), Public-key Cryptosystem Ransomware (PuCR), and Hybrid Cryptosystem Ransomware (HCR). They also proposed a framework called Connection Monitor & Connection Breaker (CM & CB) to detect high survivable ransomware (HSR) depending on the traffic of the key exchange between the victim’s machine and the C&C server. This framework detects some ransomware strains that attempt to contact their C&C servers using random domain names generated by a DGA (Domain Generation Algorithm). It analyses the domain names against a previously trained English language Markov-chain model. The evaluation results of the system for 20 samples of ransomware demonstrate 100% accuracy with zero false negative rate. However, this work proposed just one feature for HSR detection, which leads to low detection accuracy. Moreover, the effectiveness of using a Markov-chain model against short lengths of domain names is questionable and may cause high false positive rates.

Tseng et al. [115] presented a deep learning-based method to detect the communications between ransomware and the C&C server. They analysed the DNS and HTTP protocols of ransomware network traffic and extracted network signatures for some families, such as CryptoWall and Locky. The extracted information from PCAP files was employed as an input to the deep neural network, which consists of 7 layers, to obtain an optimal feature vector. The evaluation result shows that the proposed method achieved 93.92% detection accuracy.

Fasheem et al. [116] presented an Automatic Test Packet Generation (ATPG)
system to automatically detect the faults in each link in the network. ATPG is a model which produces a minimum set of test packets that are sent occasionally to the links of the network to localise the ransomware activity if it exists. ATPG creates a device-independent model by reading router configurations and recording the updated routes. Every link sends the received test packet to the receiver to check whether it is affected by ransomware action or not. If the packet is affected with ransomware, the node information is used to isolate the infected link from the network. The proposed method can detect known and unknown types of ransomware and can help to mitigate the ransomware from spreading to other links of the network. However, this approach requires that the host is already compromised by ransomware before it can be detected. Furthermore, this method becomes ineffective when ransomware tries to infect computers located on the same network and encrypts files at the same time.

Raunak and Krishnan [39] stated that more than 60% of the ransomware attacks gain access to the victim’s computer through the drive-by-download method that is often controlled by Exploit Kits (EK). Currently, drive-by-downloads are often controlled by EK, and the choice of EK is determined by the control panel depending on the vulnerabilities. They analysed the Rig Ek communications that had been used to distribute the droppers of Cerber and CryptXXX ransomware families during the year 2016-17. They indicated that the Rig EK compromises by redirecting the HTTP browser of the victim to another landing page hosted on the attacker’s website or a C&C server. They have proposed a framework using a combination of SDN and Certificate Authority Checker (CAC) to detect malicious Rig EK communication and protect users’ files from being encrypted.

Cabaj et al. [117] employed a modern network technique called Software-
Defined Networking (SDN), to detect and mitigate the ransomware attack. The presented method is based on the carefully analysing the HTTP communication characteristics of two ransomware families: Cryptowall & Locky. The proposed method was divided into three phases (learning, fine-tuning and detection). During the learning phase, the real ransomware samples were executed in a controlled environment to capture the network traffic generated by the infected machine. The HTTP messages were extracted and preprocessed from the captured traffic in this phase. The second phase is the fine-tuning phase in which the parameters of the detection method are adjusted. The centroid vector of the corresponding feature vectors, as well as the Euclidean distances from the learning set of vectors to the centroid vector, are calculated for Locky and CryptoWall families separately in this phase. The final step is the detection phase where the data of the established communication between the infected machine and the proxy server, which is gathered during the two previous phases, is utilised to detect the infection with ransomware. If suspicious behaviour is detected, the SDN controller blocks the IP or domain that extracted from the communication. The experimental results of the detection rate for Locky and Cryptowall were of 97-98% accuracy with 4-5% false positives, respectively. However, this approach is based only on the data size inserted into the outgoing HTTP POST messages.

Cusack et al. [118] used a recent property of networking hardware called Programmable Forwarding Engines (PFEs). PFEs allow processing network data at high rates of speed, while simultaneously extracting network flow records. These flow records were used by the proposed method to monitor the network traffic between the infected machine and the C&C server. They selected eight flow features made up of a combination of interarrival times, packet ratios, and burst lengths.
The features were used for building a random forest classifier to detect the traffic exchanged between ransomware and the C&C server. This method presented a moderate accuracy (86%) and a high false negative rate (about 10%).

The main gap is that a small number of research works addressed the ransomware detection [5] that focused on performing network-based analysis and extracting potential behavioural features to help in detecting the attack before carrying out the payload. Further, there are several works, such as [119] [120] [121] [122] [123], that used various host-based methods to detect the ransomware activities.

3.3 Setting up the testbed

This section explains the preparations to set up a security lab to run and analyse the network activities of Locky ransomware.

3.3.1 Malware testbed environment

Dynamic analysis of malware requires a safe and controlled environment to carry out the samples of malware and monitor its malicious activities. A dedicated testbed has been built that consists of three real computers and two virtual machines, as depicted in Fig. 3.1. The objective of this testbed is to run some samples of ransomware and capture their network traffic as PCAP files. These files are then analysed to extract a set of network features that characterise the communication behaviour between the ransomware and its C&C server (i.e., attacker). The details of the testbed’s components are indicated below:

- PC1 is used as a victim computer, where real ransomware samples are
3.3 Setting up the testbed

executed inside it.

- PC2 and its hosted virtual machines (i.e., VPC1 and VPC2) are three clean machines on the network representing the non-infected computers. The network traffic analysis of these computers allows investigation of how ransomware attempts to spread over a network.

- PC3 captures the ransomware network traffic and stores it in PCAP files using Wireshark tool. For that, its NIC (Network Interface Card) is set to work in promiscuous mode. Since it is necessary to keep PC3 protected from the ransomware’s spread attempts, Ubuntu Linux was chosen as the OS (Operating System), as the selected ransomware samples do not target Linux systems. Moreover, a firewall has been setup with a strict rule to drop any packet try to infect PC3 with ransomware.

- This testbed is separated from the university’s campus network in order to protect the network from the spreading attempts of ransomware. The PCs are linked to the Internet because most of the ransomware families (including Locky) require contact with a C&C server. The ISP (Internet Service Provider) who provides the dedicated internet service to the network security lab was informed in advance about my intention to run some ransomware samples on their public network.

3.3.2 Ransomware dataset

Having a valid dataset is a crucial part in allowing accurate analysis and extraction of related network features. The lack of datasets for ransomware [5] was a
3.3 Setting up the testbed

Figure 3.1: The malware testbed architecture

The principal challenge in this research work. A number of benchmark datasets have
been examined, such as [124] [125]:

- KDDCUP 99
- NSL-KDD (2009)
- Canadian Institute for Cybersecurity (CIC) datasets

Unfortunately, these datasets do not contain any records of a ransomware
attack, although the first attack was discovered in 1989 [78]. Thus, the dedicated
testbed was built to create a new dataset taking the Locky ransomware as a case
study as outlined above.

However, it was noticed that the testbed-generated dataset only contains uni-
directional traffic, i.e., the traffic originated from Locky (at the victim’s end)
3.3 Setting up the testbed

trying to find the C&C server, without any response traffic coming from the attacker’s side. This problem perhaps due to the probability of the C&C servers being down after the campaign was over, their IP addresses were blacklisted, or the attacker was able to identify my environment as a research lab, rather than a real victim.

Therefore, an extensive search has performed to find a reliable dataset containing bidirectional traffic captured when the campaign was still active. I found a dataset created within the Stratosphere project in the Czech Technical University (CTU), the Malware Capture Facility Project (MCFP) dataset [126] [127]. In this project, real malware samples were executed within a real network environment for up to several weeks, around the first appearance of Locky. It also contains a significant amount of normal traffic. The size of normal traffic is much larger than the malicious traffic, as it includes all traffic generated by different normal services running on the system. Contrariwise, the malicious dataset contains packets generated only by Locky, which intentionally tries to send a small number of packets over time to avoid detection by security systems. Table 3.1 demonstrates the details of the MCFP portion employed in this work.

3.3.3 Locky ransomware samples collection

Locky samples were collected from the following open source communities:

- virushare.com

- malware-traffic-analysis.net

The types and the hash values of the samples were inspected using virustotal.com to check that the samples are valid and have the correct labels.
3.4 Locky network traffic analysis and features extraction

This section analyses the network traffic of Locky ransomware to extract a set of related network features. Locky often propagates over spam emails that contain a harmful attachment in the form of macro-enabled office documents. This attachment carries out a script, called a downloader, to download Locky’s executable file from a harmful URL and install it on the victim’s system. When fully installed, Locky attempts to find and contact its C&C server(s) to exchange encryption keys and carry out the intended malicious actions through one of the following ways:

- It utilises an encrypted list of embedded IP addresses to establish a TCP session with the C&C server(s).

- If these embedded IP addresses are unreachable, i.e., blacklisted, or the ses-
sion was disrupted, Locky will try to find its C&C server(s) by carrying out the Domain Generation Algorithm (DGA) that periodically generates many pseudorandom domain names. Locky keeps sending DNS query messages about these names until the real C&C server(s) is found.

- If the DGA method fails, Locky uses the NetBIOS Name Service (NBNS) protocol hoping to find a previously infected computer on the local network that has resolved the name of the C&C server.

- If all the ways above fail, it uses offline encryption mode. In this mode, a predefined public key embedded in the ransomware is employed to encrypt the files without need of a connection with the C&C server. In this case, the RSA private key is either stored locally on the infected computer or kept at the attacker side so that the same key pair is used for the entire campaign. If the key is stored locally, the malware analyst could discover it through reverse engineering. Conversely, if the same key pair is used for the entire campaign, it means multiple offline victims are sharing the same key pair. So, if one of the victims pays the ransom and receives the private key, he can help other victims to decrypt their files without paying the ransom [77] [128].

If Locky successfully establishes a TCP session with its C&C server, the attacker can use it to control the payload execution. On the other hand, Locky uses the HTTP post request method to send data back to the attacker. Locky uses a wide range of network protocols that can be beneficial to extract potential network features. Table 3.2 indicates statistics of the DNS, NBNS, and HTTP protocols within the collected PCAP files.
3.4 Locky network traffic analysis and features extraction

Table 3.2: Network protocols’ statistics

<table>
<thead>
<tr>
<th>Protocol Name</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>3.77</td>
</tr>
<tr>
<td>DNS</td>
<td>7.25</td>
</tr>
<tr>
<td>NBNS</td>
<td>6.05</td>
</tr>
</tbody>
</table>

The Locky’s PCAP files were investigated using Python scripts and MATLAB to extract informative network-based features. The analysis has shown that the extracted features can be classified based on two aspects producing four different types of features: the behavioural and the non-behavioural features, and the detectable and the non-detectable features.

A behavioural feature is a feature that is very difficult to suppress by future malware variants, if it is built upon the same underlying technology, e.g., DGA. While a non-behavioural feature is a feature that appears within the traffic being analysed, however, it is possible this (feature) is absent in future variants even if they use the same underlying technology.

A detectable feature is a feature of the malicious traffic that can still be measured in the mixed traffic, i.e., the traffic of an infected computer where both the malicious and benign streams coexist. A non-detectable feature is one that appears in malicious traffic only; however, the malware was able to obfuscate it within the normal traffic, so that it becomes indistinguishable. The following subsections demonstrate these features in detail, supported by experiments and statistics.
3.4 Locky network traffic analysis and features extraction

3.4.1 TCP traffic analysis

3.4.1.1 Reset connections

It was observed that there are many TCP reset (RST, ACK) packets in Locky’s traffic used to terminate the malicious TCP connections abnormally. They are distinguishable when compared to regular traffic. Fig. 3.2 demonstrates the ratio of reset connections in the overall malicious traffic compared to that of the benign traffic.

![Figure 3.2: The reset connection ratio](image)

The reset connection ratio (%RSTConn) attribute, seems to be a related feature. However, after examining the time-based rate of the malicious (RST, ACK) packets, they are found to be well distributed over time. This distribution allows the Locky’s packets to blend in with normal (RST, ACK) packets without a significant increase in the time-based rate. In other words, it seems that Locky intentionally does not create many connections within a short period to avoid detection. Fig. 3.3 shows this distribution by comparing the frequency of the malicious reset packets beside the normal reset packets every minute. It is noticeable that this feature is not easily detectable when the mixed flow of packets
is monitored as a whole, i.e., malicious and benign streams.

Calculating the IP-wise reset connections ratio, (%RSTConnIPwise), can provide a recognisable result, in that, connections with the remote attack servers were expected to maintain the same reset ratio found in the overall malicious traffic, i.e., about 99%. Unexpectedly, I found that in most time-frames, e.g., a 15-minute time-frame, there are few connections in the normal traffic with a set of destination IP addresses have a similar reset ratio to that of the malicious traffic, which makes this attribute a non-detectable feature also. Fig. 3.4 illustrates an example 15-minute time-frame. It is noticeable that there are some benign IP addresses (Fig. 3.4 (b)) that have a reset connection ratio as high as that of the malicious IP addresses (Fig. 3.4 (a)).

However, these sets of IP addresses in the normal traffic vary across different time-frames, whereas the same set of IP addresses repeats in the malicious traffic. Fig. 3.5(b) illustrates how the set of benign IP addresses taken in a subsequent 15-minute time-frame is different from that shown in Fig. 3.4(b). While subsequent time-frames share the same set of malicious IP addresses for the malicious traffic, see Fig. 3.4(a) and Fig. 3.5 (a). This thing led to identifying a detectable feature of having the same set of IP addresses with a high reset connection ratio repeated almost every time-frame (%RSTConnRepIPSet).

On the other hand, these RST-based features are all classified as non-behavioural features, since apparently nothing can prevent new variants of Locky from terminating their sessions normally, i.e., using TCP FIN packet without RST.
3.4 Locky network traffic analysis and features extraction

Figure 3.3: The frequency of the TCP reset packets (a) within the 1\textsuperscript{st} hour (b) within the 2\textsuperscript{nd} hour (c) within the 3\textsuperscript{rd} hour
3.4 Locky network traffic analysis and features extraction

Figure 3.4: Samples of the IP-wise reset connections ratio (a) malicious samples (b) benign samples

Figure 3.5: Samples of the IP-wise reset connections ratio in a subsequent timeframe (a) malicious samples (b) benign samples
3.4 Locky network traffic analysis and features extraction

3.4.1.2 HTTP traffic

Attackers are continually building reliable C&C infrastructures and establishing covert communication channels for their malware to bypass intrusion detection systems. Ransomware uses common ports that are usually open for normal traffic, e.g., ports 80 and 443, to connect with the intermediate proxy server(s) (Fig. 3.1). After analysing the captured PCAP files generated by Locky, it can be noticed that it was using Trojan-like behaviour. It establishes a short TCP session with a remote server, where it uses the HTTP post method to send data to that server. Locky re-establishes a new connection with the same embedded address after a period repeating the prior activities, as shown in Fig. 3.6.

![Figure 3.6: Locky’s malicious channels](image)

It is noticeable that, unlike the normal traffic, the majority of Locky’s HTTP requests were POST methods. Fig. 3.7 shows the POST methods ratio found in the malicious traffic compared to that of the normal traffic concerning other
3.4 Locky network traffic analysis and features extraction

HTTP request methods, e.g., GET method.

![Figure 3.7: HTTP POST method ratio: Malicious Vs Benign](image)

This attribute (#HTTP-POSTs) seems to be an informative feature. However, the time-based frequency of this feature needs examination to determine if it is a detectable or non-detectable feature. Fig. 3.8 illustrates the frequency of HTTP-POSTs taken over 15 subsequent time-frames, each of 15 minutes long. It is clear that Locky significantly increases the number of HTTP-POSTs within the traffic stream, compared to the normal traffic. Therefore, #HTTP-POSTs is classified as a detectable feature. Besides, it is classified as a behavioural feature, considering that Locky, as a Trojan, is expected to POST data to its remote server with a higher frequency than GET requests.

Most of the collected Locky variants use POST methods without naming the User-Agent. Conversely, there was no POST method found without a User-Agent in the normal traffic. Hence, another detectable feature can be determined as Nil-User-Agent. However, this feature does not apply to all Locky variants, as several were found to be impersonating the Mozilla/4.0 user agent. Therefore, it is classified as a non-behavioural feature.
3.4 Locky network traffic analysis and features extraction

Figure 3.8: HTTP POST method frequencies that were taken over subsequent time-frames

3.4.2 DNS traffic analysis

DNS has been widely used by cyberattackers for several purposes:

- To avoid the need for an embedded list of servers’ IP addresses to be injected within the malware binary file, as they can be easily discovered and blacklisted. Instead, DGA is used to generate many pseudorandom domain names so that DNS requests are continuously generated until contact with the real server is made [129].

- DNS is used for sending instructions to the malware like bringing a particular domain name up or down. An example for this case is the Kill Switch domain name of the WannaCry ransomware [119]. If WannaCry can solve this domain name, it will stop its propagation across the networks.

- DNS can sometimes be employed to transfer data out of a secure network [65] where no outbound TCP connectivity is allowed. So, the malware can
3.4 Locky network traffic analysis and features extraction

divide the file data into several DNS requests sent to a fake DNS server, which resides at the attacker’s side. These DNS request packets are parsed by the fake DNS server to extract the data chunks, concatenate them, and then reform the original data.

Based on this knowledge, the DNS packets of the collected PCAP files are analysed to extract related features, as illustrated in the following subsections.

3.4.2.1 DNS name error

Many DNS name error packets were observed in the traffic of Locky. These name errors happened due to the use of the DGA algorithm that generates a large number of pseudorandom domain names, for which Locky has to send the corresponding DNS requests until the server is found. Fig. 3.9 shows samples of these name errors found in the Locky’s network traffic. Fig. 3.10 displays the statistics of the number of DNS name errors found in the malicious (Locky) traffic compared to that of the benign traffic.

Figure 3.9: Samples of the name errors within the Locky’s traffic
This significant difference between the malicious and benign streams may produce a related feature based on the number of DNS name errors (#DNS-NE). However, by applying the same logic explained earlier in the preceding section, it is also important to analyse the time-based value distribution for this feature. For that, the frequency of the DNS name errors is calculated over various time-frames, as shown in Fig. 3.11. It is apparent that Locky produces a large number of DNS name errors, which are detectable even within the mixed traffic. It is expected that any future variant of Locky ransomware cannot suppress this feature as far as it still based on the same underlying technology, i.e., DGA. For that, I have classified this feature (#DNS-NE) to be a behavioural feature.
3.4 Locky network traffic analysis and features extraction

3.4.2.2 Meaningless domain name

As stated earlier, Locky sends many DNS requests regarding the domain names generated by DGA. These names are pseudorandom, in that they are ambiguous to humans. Table 3.3 indicates some examples of these meaningless domain names found within Locky traffic. Such incomprehensible names were absent in the normal traffic, which makes this feature ((Meaningless Domain Name (MDN)) detectable. However, it requires an algorithm to recognise such names accurately.

Table 3.3: Some examples of the meaningless domain names

<table>
<thead>
<tr>
<th>No.</th>
<th>Domain name</th>
<th>Domain length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lxnorppktbolinepnh.ru</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>unsfcsaaxmxgv.xyz</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>sfiavgpadtlipui.click</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>oaedvogymkqaivswf.xyz</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>qpxyvqkebhuohsrs.pw</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>xakckpoovshnyxq.biz</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>hussbshcfbhqypv.info</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>fvxmechgihtd.org</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>hldorpxdaaxshby.nl</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>kwipuwsviahhs.ru</td>
<td>17</td>
</tr>
<tr>
<td>11</td>
<td>mcyjduv.pm</td>
<td>10</td>
</tr>
</tbody>
</table>

It is also observed that Locky attempts to resolve these meaningless names several times in case of failure. Fig. 3.12 displays the top frequencies of the DNS queries sent by a Locky sample for different meaningless domain names. Similar to the #DNS-NE feature, this feature (MDN) is classified as a behavioural feature.

Recognising the meaningless domain names was not a straightforward process. A combination of two algorithms was adopted to identify these domains. The first one uses the Shannon Entropy metric to measure the distribution randomness of the domain name [130]. Entropy can be computed using the following equation (1):
3.4 Locky network traffic analysis and features extraction

Figure 3.12: The top frequencies of meaningless DNS domain names

\[
H = -\sum_{i=1}^{n} p_i \log p_i \tag{3.1}
\]

where \( p_i \) is probability distribution \( p_n = (p_1, ..., p_n) \) with \( p_i \geq 0 \) for \( i = 1, ..., n \) and \( \sum_{i=1}^{n} p_i = 1 \)

The entropy was only computed for the first label of every domain name, as the second label usually represents the root DNS servers, such as org, com, and net. Table 3.4 shows some selected examples of domain names and their calculated entropy values.

Table 3.4: Examples of the entropy, max. no. of sequential consonants and max. no. of sequential vowels computed for some domain names

<table>
<thead>
<tr>
<th>Domain name</th>
<th>Domain name Type</th>
<th>Entropy</th>
<th>Max. no. of sequential consonants letters</th>
<th>Max. no. of sequential vowel letters</th>
</tr>
</thead>
<tbody>
<tr>
<td>dafyupcawgblqao.be</td>
<td>malicious</td>
<td>3.7736</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>qhswwewgdvlhhus.de</td>
<td>malicious</td>
<td>3.3788</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>unsfcsaxxmvgv.xyz</td>
<td>malicious</td>
<td>3.2089</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>unonqhplpqxmxred.su</td>
<td>malicious</td>
<td>3.6402</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>ggvvecppijadin.uk</td>
<td>malicious</td>
<td>3.5216</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>tiles.services.mozilla.com</td>
<td>normal</td>
<td>2.3219</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>aps.google.com</td>
<td>normal</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>platformid.adobe.com</td>
<td>normal</td>
<td>2.1219</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>platform-id-twitter.com</td>
<td>normal</td>
<td>3.4494</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>connect.facebook.net</td>
<td>normal</td>
<td>2.2359</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
3.4 Locky network traffic analysis and features extraction

As shown in Table 3.4, entropy alone may not be sufficient to measure the randomness of a domain name. For this purpose, a Python function called unpronounceable has been written. This function helps to determine the randomness of a domain name, based on the fact that it is unusual to see some sequential consonant or vowel letters within a normal domain name. This function returns the maximum number of sequential consonants and vowels in the first label of the domain name, as shown in Table 3.4. For example, the maximum numbers of the sequential consonants and vowels for the domain name grouxffcnnv.xyz are 6 and 2 respectively. Generally, the letter “y” can be regarded as a vowel or a consonant. However, the experiments conducted on the dataset indicate that considering “y” as a vowel enhances accuracy. Fig. 3.13 demonstrates my algorithm that combines the two methods earlier discussed, i.e., the entropy and unpronounceable function, to identify the randomness domain names. The threshold values for entropy and unpronounceable were determined experimentally to achieve high accuracy.

3.4.2.3 Further features of DNS

Having several DNS-based behavioural and detectable features motivate the examination of more DNS features utilising machine learning. Therefore, a total of 9 DNS raw attributes were extracted and fed into a feature selection engine using Weka, a well-known machine learning tool [131], following the configurations depicted in Fig. 3.14. Table 3.5 indicates a list of the ranked attributes extracted by Weka.

I have analysed the PCAP files against the first attribute in Table 3.5, i.e., dns-count-labels. It was found that Locky commonly uses DNS names of only
3.4 Locky network traffic analysis and features extraction

![Flow chart of the domain name randomness measuring algorithm](image)

Figure 3.13: The flow chart of the domain name randomness measuring algorithm

two labels. While benign traffic uses DNS names with several labels’ counts distributed mainly around 3 and 4, see Fig. 3.15. Hence, a feature of the DNS labels’ count ratio (%dns-count-labels) can be derived to distinguish between the malicious and benign traffic, based on the labels’ count as proposed by Weka. This feature is also classified as a detectable and non-behavioural feature because there is nothing that prevents future variants of Locky from using DNS names with several labels’ counts, as it is found in the benign traffic. However, most of the domains generated by DGA have two labels, and even if Locky uses domains with more than two labels, the malicious traffic still recognisable through the behavioural features, which are resilient against changes, and other non-behavioural features that extracted from two different levels.
3.4 Locky network traffic analysis and features extraction

Figure 3.14: Attributes selection using Weka

Table 3.5: The ranked attributes by Weka

<table>
<thead>
<tr>
<th>No.</th>
<th>Attribute name</th>
<th>Rank</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dns-count-labels</td>
<td>1</td>
<td>The number of labels of a DNS query name</td>
</tr>
<tr>
<td>2</td>
<td>dns-resp-ttl</td>
<td>2</td>
<td>Time to live of the DNS response packet</td>
</tr>
<tr>
<td>3</td>
<td>dns-ipv6</td>
<td>3</td>
<td>Communication with the DNS server using IPv6 address</td>
</tr>
<tr>
<td>4</td>
<td>dns-ipv4</td>
<td>3</td>
<td>Communication with the DNS server using IPv4 address</td>
</tr>
<tr>
<td>5</td>
<td>dns-resp-typ</td>
<td>4</td>
<td>DNS response packet type</td>
</tr>
<tr>
<td>6</td>
<td>dns-resp-len</td>
<td>5</td>
<td>The length of the DNS response packet</td>
</tr>
<tr>
<td>7</td>
<td>dns-qry-name-len</td>
<td>6</td>
<td>The length of a DNS query name</td>
</tr>
<tr>
<td>8</td>
<td>dns-time</td>
<td>7</td>
<td>The time between the DNS query and response packets</td>
</tr>
<tr>
<td>9</td>
<td>dns-qry-typ</td>
<td>8</td>
<td>DNS query type</td>
</tr>
</tbody>
</table>

3.4.3 NetBIOS name service (NBNS)

Network Basic Input / Output System (NetBIOS) is an OSI session layer protocols, which allows applications on the computers to communicate across a local network and the Internet. It can provide a connection (session) and connectionless (datagram) services besides to broadcast and multicast communications. NetBIOS name service (NBNS) is a part of the NetBIOS-over-TCP protocol suite and works somewhat like DNS work, in that it maps NetBIOS names to IP ad-
3.4 Locky network traffic analysis and features extraction

![Figure 3.15: The DNS labels’ count: Malicious Vs Benign](image)

Typically, NBNS uses UDP port 137 and can also use TCP port 137 [65].

It is noticeable from Table 3.2 that NBNS packets form 6.05% of the total malicious traffic, while they were almost absent from the benign traffic. It was also observed that when Locky receives a DNS name error packet about a particular domain name, it attempts to resolve this name again using NBNS, expecting other pre-infected computers on the network have an answer for the domain name. Fig. 3.16 displays the statistics for several suspicious NetBIOS queries found within the malicious traffic.

![Figure 3.16: The top frequencies of NBNS queries within the malicious traffic](image)
3.5 Building a multi-classifier intrusion detection system

Similar to the DNS features, the feature (MNBNS) is classified as a detectable and behavioural feature. Table 3.6 summarises all the extracted network features classified previously. The features that fall into the Behavioral and Detectable categories are considered to be class I features. Moreover, the features can be further classified into being packet-level vs flow-level and Simple (easy to measure) Vs Complex, as shown in Table 3.7. Class I features that are also classified as Simple are the most desirable ones.

Table 3.6: Summary of the extracted features classified into: Behavioral vs Non-behavioral & Detectable vs Non-detectable

<table>
<thead>
<tr>
<th>Detectable</th>
<th>Behavioral</th>
<th>Non-detectable</th>
</tr>
</thead>
<tbody>
<tr>
<td>#HTTP-POSTs, DNS-NE, #DNS-NE, MDN, #MDN, MNBNS</td>
<td>%RSTConnRepIPSet, Nil-User-Agent, dns-ipv6, dns-ipv4, %dns-count-labels, dns-count-labels, dns-resp-len, dns-resp-ttl, dns-resp-typ, dns-qry-typ, dns-qry-name-len, dns-time</td>
<td>%RSTConn, %RSTConnIPwise</td>
</tr>
</tbody>
</table>

3.5 Building a multi-classifier intrusion detection system

This section presents a multi-classifier intrusion detection system, which is designed to track crypto ransomware network activities depending on the extracted features. This system has two independent binary classifiers: packet-based classi-
3.5 Building a multi-classifier intrusion detection system

Table 3.7: Summary of the extracted features classified into: Packet-level vs Flow-level & Simple vs Complex

<table>
<thead>
<tr>
<th>Simple</th>
<th>Packet-level</th>
<th>Flow-level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nil-User-Agent, DNS-NE, dns-count-labels, dns-ipv6, dns-ipv4, dns-resp-len, dns-resp-ttl, dns-resp-typ, dns-qry-typ, dns-qry-name-len, dns-time</td>
<td>%RSTConn, %RSTConnIPwise, %RSTConnRepIPSet, #HTTP-POSTs, #DNS-NE, %dns-count-labels</td>
</tr>
<tr>
<td>Complex</td>
<td>MDN, MNBNS</td>
<td>#MDN</td>
</tr>
</tbody>
</table>

Fier (C₁) and flow-based classifier (C₂), working in parallel to detect the packet-level and the flow-level features shown earlier in Table 3.7. Fig. 3.17 illustrates the architecture of the multi-classifier detection system. It consists of two main modules: The Feature Extraction Module and the Decision Making Module.

The incoming packets are forwarded to the packet-level data preparation unit to extract the packet-level feature vector (V₁). Moreover, a buffer unit is employed to collect and store the incoming packets temporarily. Every two minutes, the data of the buffer is sent to the flow-level data preparation unit to extract the flow-level feature vector (V₂). The two vectors V₁ and V₂ are forwarded to classifier C₁ and C₂ respectively to detect any ransomware network action. Finally, a simple OR-based decision unit is used to receive the output of the individual classifiers independently, i.e., C₁ and C₂, and trigger an alarm if any of the classifiers detect a malicious activity.
3.5 Building a multi-classifier intrusion detection system

3.5.1 Data preparation

The data preprocessing unit receives the raw packet data as an input, handles any missing and outlier values, then prepares data for feature extraction. For example, it extracts the data of the domain name out of the incoming packet and computes its entropy value. The feature extraction unit takes the output from the data preprocessing unit and extracts the corresponding features accordingly. For example, it runs the algorithm shown in Fig. 3.13 to determine the randomness of the extracted domain name. As discussed before, two levels of data preparation (the packet-level and the flow-level) have been used. Table 3.8 depicts the selected features used for the experiments from Table 3.7. The data preprocessing and feature extraction units were built using some Python libraries, such as dpkt, socket, numpy, and csv. The feature extraction unit prepares the feature vector for the corresponding classifier in a csv format.
3.5 Building a multi-classifier intrusion detection system

Table 3.8: The selected packet-level and flow-level features

<table>
<thead>
<tr>
<th>Type</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet-level</td>
<td>MDN</td>
</tr>
<tr>
<td></td>
<td>DNS-NE</td>
</tr>
<tr>
<td></td>
<td>MNBNS</td>
</tr>
<tr>
<td></td>
<td>dns-ipv6</td>
</tr>
<tr>
<td></td>
<td>dns-ipv4</td>
</tr>
<tr>
<td></td>
<td>dns-count-labels</td>
</tr>
<tr>
<td></td>
<td>dns-resp-len</td>
</tr>
<tr>
<td></td>
<td>dns-resp-ttl</td>
</tr>
<tr>
<td></td>
<td>dns-resp-typ</td>
</tr>
<tr>
<td></td>
<td>dns-qry-typ</td>
</tr>
<tr>
<td></td>
<td>dns-qry-name-len</td>
</tr>
<tr>
<td></td>
<td>dns-time</td>
</tr>
<tr>
<td>Flow-level</td>
<td>#MDN</td>
</tr>
<tr>
<td></td>
<td>#DNS-NE</td>
</tr>
<tr>
<td></td>
<td>#HTTP-POSTs</td>
</tr>
<tr>
<td></td>
<td>%RSTConnReplIPSet</td>
</tr>
</tbody>
</table>

3.5.2 Models building

The raw PCAP files illustrated in Table 3.1 were employed to build the packet-based and flow-based labelled datasets to train and test classifiers $C_1$ and $C_2$ respectively, as shown in Fig. 3.18. Then, each dataset was split randomly into two disjointed subsets: the training and the testing datasets. Since there is no rule to specify the percentage of the training and testing datasets [13], the percentages indicated in Table 3.9 have been adopted.

Figure 3.18: Building labelled datasets out of the raw PCAP files
3.5 Building a multi-classifier intrusion detection system

Table 3.9: Percentages of the training and testing datasets

<table>
<thead>
<tr>
<th>Type</th>
<th>Training dataset</th>
<th>Testing dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet-based</td>
<td>70%</td>
<td>30%</td>
</tr>
<tr>
<td>Flow-based</td>
<td>85%</td>
<td>15%</td>
</tr>
</tbody>
</table>

Weka tool was utilised for training the classifiers with the training datasets using the cross validation option that can both validate the model and prevent overfitting. Although applying the cross validation method is considered enough [16] to estimate the performance of the model, however, it is recommended to assess the trained model using a testing dataset, which was not seen by the model during the learning phase.

Several popular learning algorithms, such as Random Forest, LibSVM, Bayes Net, and Random Tree, were selected to build each classifier $C_1$ and $C_2$ autonomously. Finally, the algorithm that provides the best accuracy was chosen by each classifier accordingly.

3.5.3 Models evaluation

Some classification metrics [102] [107], i.e., accuracy, False Positive Rate (FPR), precision, recall and F1 score, were employed to evaluate the performance of each model. These metrics are derived from the confusion matrix (Table 3.10) as shown by equations (3.2), (3.3), (3.4), (3.5), and (3.6) respectively.

Table 3.10: Confusion matrix

<table>
<thead>
<tr>
<th>Actual (ransomware)</th>
<th>Predicted (ransomware)</th>
<th>Predicted (benign)</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive (TP)</td>
<td>False Negative (FN)</td>
<td></td>
</tr>
<tr>
<td>False Positive (FP)</td>
<td>True Negative (TN)</td>
<td></td>
</tr>
</tbody>
</table>

Where:
3.5 Building a multi-classifier intrusion detection system

True positive (TP): Number of ransomware samples that were classified as ransomware (correct prediction). False Positive (FP): Number of benign samples that were classified as ransomware (incorrect prediction). True negative (TN): Number of benign samples that were classified as benign (correct prediction). False Negative (FN): Number of ransomware samples that were classified as Benign (incorrect prediction).

\[
Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.2}
\]

\[
FPR = \frac{FP}{FP + TN} \tag{3.3}
\]

\[
Precision = \frac{TP}{TP + FP} \tag{3.4}
\]

\[
Recall = \frac{TP}{TP + FN} \tag{3.5}
\]

\[
F1\text{score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.6}
\]

Tables 3.11 and 3.12 show the evaluation results for the packet-based and the flow-based models respectively. Fig. 3.19 and Fig. 3.20 illustrate the values of the chosen performance metrics. It is noticeable that for the selected dataset, classifier \( C_1 \) provides the highest performance when it employs the Random Tree algorithm. While classifier \( C_2 \) provides the highest performance when it uses Bayes Net algorithm.
3.5 Building a multi-classifier intrusion detection system

Table 3.11: Packet-based classifier evaluation

<table>
<thead>
<tr>
<th>Dataset Type</th>
<th>Algorithm name</th>
<th>Accuracy (cross validation 10 folds)</th>
<th>Accuracy (testing)</th>
<th>FPR</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet-based</td>
<td>Random Forest</td>
<td>97.45</td>
<td>95.83</td>
<td>0.042</td>
<td>0.962</td>
<td>0.958</td>
<td>0.958</td>
</tr>
<tr>
<td></td>
<td>LibSVM</td>
<td>96.15</td>
<td>91.67</td>
<td>0.083</td>
<td>0.917</td>
<td>0.917</td>
<td>0.917</td>
</tr>
<tr>
<td></td>
<td>Bayes Net</td>
<td>95.51</td>
<td>93.75</td>
<td>0.064</td>
<td>0.938</td>
<td>0.938</td>
<td>0.937</td>
</tr>
<tr>
<td></td>
<td>Random Tree</td>
<td>98.72</td>
<td>97.92</td>
<td>0.021</td>
<td>0.979</td>
<td>0.979</td>
<td>0.979</td>
</tr>
</tbody>
</table>

Table 3.12: Flow-based classifier evaluation

<table>
<thead>
<tr>
<th>Dataset Type</th>
<th>Algorithm name</th>
<th>Accuracy (cross validation 10 folds)</th>
<th>Accuracy (testing)</th>
<th>FPR</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow-based</td>
<td>Random Forest</td>
<td>99.74</td>
<td>96.49</td>
<td>0.036</td>
<td>0.965</td>
<td>0.965</td>
<td>0.965</td>
</tr>
<tr>
<td></td>
<td>LibSVM</td>
<td>99.56</td>
<td>94.74</td>
<td>0.056</td>
<td>0.947</td>
<td>0.947</td>
<td>0.947</td>
</tr>
<tr>
<td></td>
<td>Bayes Net</td>
<td>99.83</td>
<td>97.08</td>
<td>0.029</td>
<td>0.972</td>
<td>0.971</td>
<td>0.971</td>
</tr>
<tr>
<td></td>
<td>Random Tree</td>
<td>99.74</td>
<td>95.91</td>
<td>0.041</td>
<td>0.959</td>
<td>0.959</td>
<td>0.959</td>
</tr>
</tbody>
</table>

Figure 3.19: Performance metrics of each algorithm for the packet-based classifier

Figure 3.20: Performance metrics of each algorithm for the flow-based classifier
3.6 Summary

Building upon the state-of-the-art research on network-based detection, this chapter provides a thorough analysis of crypto ransomware network traffic, and it presents an advanced detection method, using Locky as a case study. A dedicated malware testbed was built, and Locky network activities were captured. Moreover, the PCAP files of Locky in the MCFP dataset were also collected and analysed thoroughly. The analysis shows that Locky has several network activities that can be used to extract potential behavioural features.

A domain name randomness measuring algorithm was proposed and applied to identify the meaningless domain names. A total of 18 features were extracted from two independent levels for the traffic of TCP, HTTP, DNS, and NBNS protocols. By using a new classification scheme, these extracted features were classified into four types: Behavioral vs Non-behavioral & Detectable vs Non-detectable. The features are related and can differentiate traffic generated by a compromised host.

Furthermore, a multi-classifier network-based ransomware detection system was proposed and prototyped. This system works on two different levels: the packet-level and the flow-level. The experimental results for the packet-level and the flow-level demonstrate a high detection accuracy for each level: 97.92% and 97.08% respectively and prove the effectiveness of the extracted features. The discovery of these new features significantly contributes to the field of research in network-based malware detection, and provide a strong foundation for the advancements of next-generation network-based malware detection systems.
Chapter 4

An ML based Approach for Detecting Algorithmically Generated Domain Names

4.1 Introduction

Cybercriminals often depend on C&C channels to launch their cyberattacks. They can run several harmful activities, such as data exfiltration, spamming and downloading malicious files by directing the compromised machines via C&C channels. To protect these secret channels from being detected and blocked by a firewall, attackers have started to build reliable C&C infrastructures to launch their cyberattacks remotely. Therefore, different C&C architectures and communication methods have developed to make the process of detection and disruption of these channels very difficult.

One of the main C&C communication approaches is domain fluxing, where
malware algorithmically generates many pseudo random domain names using DGA (Domain Generation Algorithm). The malware then tries to contact its C&C server by querying these domain names one after another, until an active one is resolved. The malware, such as ransomware and botnets, uses the DGA technique to make a connection with their C&C server. Therefore, detecting and blocking the DGA-based domain names can disrupt the covert channels between the victims and the controllers (i.e., cybercriminals) and mitigate the harmful effects of these severe attacks.

In this chapter, a new method for detecting algorithmically generated domain names, MaldomDetector [132] is proposed and implemented using a machine learning technique. This method consists of two modules and can detect the DGA-based domain names before the malware can make any successful connection with its C&C server. MaldomDetector can operate efficiently as an initial alarm to detect DGA-based domains of malware families while keeping a high detection accuracy.

4.2 Related work

Several research works have attempted to detect algorithmically generated domain names by analysing the names’ strings. Xu et al. [133] presented a method to detect DGA-based domain names through examining the strings of the domain names, by combining an n-gram method and deep convolutional neural network. They proposed an n-gram combined character-based domain classification (n-CBDC) model to explore the characters’ features of the domain names and extract inherent semantic differences between malicious and benign domain
names. The n-CBDC model does not require hand-extracted features or DNS context information. It only needs the domain name itself as an input to estimate automatically the probability whether the domain name was generated by DGA or not. The n-CBDC model was trained using a supervised learning method, which requires a labelled dataset collected during research work. The authors provided evaluation results for different types of DGA using bigram and trigram representations. The average detection accuracies for 2-gram and 3-gram were 94.15% and 98.29% respectively. However, n-gram based method is computationally expensive and language-dependent.

Yu et al. [134] presented a deep learning-based method to detect DGA domain names. They also proposed a labelling method based on heuristics rules, which provides a fast and easy way for labelling domains weakly, i.e., low confidence, in real network traffic. This heuristic method was used to obtain a large number of labelled domains to address the need for large amount of labelled dataset for training deep learning models. Presented experimental results showed that using the weakly labelled dataset has improved the predictive accuracy of DGA classifiers based on deep learning. However, this method has several parameters that must be estimated, resulting in high computational cost, and requires large amounts of training data to learn, i.e., need a long time for training.

Selvi et al. [135] built a dataset containing normal and DGA-based domain names and classified different malware families according to their type of DGA. Their dataset consists of 32k DGA-based domains and 32k Legitimate domains which is freely available for researchers. The authors proposed a masked n-gram model and presented a machine learning approach, which used a Random Forest algorithm to detect DGA-based domain names. N-gram is an adjacent sequence
of n items from a given sample of text. In masked N-grams, every character in
the second-level domain name is substituted by a character that represents its
type (i.e., consonant, vowel, digit, or special symbol). They extracted features
that rely on several characteristics, such as the lexical attributes of the domain
names, some statistical information, and masked N-grams. The experimental
results showed a detection accuracy of 98.91% and false positive rate of 0.76%.
However, most of the extracted features are statistical (i.e. mean, variance, and
standard deviation) that become less effective when the domain name is short.
Additionally, this method requires a relatively high training time (approximately
1.21 hours).

Lv et al. [136] analysed the difference in various characteristics of DNS log
communications between malicious and benign domain names. A total of 12
attributes from five categories were extracted from the characteristics of DNS
traffic. One of these categories relates to the characters of the domain name.
The authors used Hidden Markov Model (HMM) to build a malicious domain
name detection system along with tools from Spark technology to enhance the
classification. The attributes of the domain names were taken as an input to the
HMM model to determine the probability distribution of the states and identify
whether there is a malicious domain name or not. HMM requires expensive com-
putations and extracting data from the DNS response message. The classification
accuracy and the recall rate obtained from the evaluation results were 91.52% and
89.32% respectively.

Mac et al. [137] presented the investigation of various supervised learning
methods, such as Hidden Markov Model (HMM), C4.5 decision tree, SVM, Ex-
treme Learning Machine (ELM), Long Short-Term Memory network (LSTM),
CNN+LSTM, and bidirectional LSTM. They evaluated these various methods using a real collected dataset containing 38 DGA families with 168900 samples. The authors showed that featureless detection systems could produce good results without the need for forming a feature set, and they claimed that the feature-based detection systems, such as C4.5, SVM, and ELM, can be easily circumvented by malware authors. They also offer some important qualities such as transparency, efficiency, and the ability of fine-tuning the algorithms. The maximum recall obtained from the experiments for bidirectional LSTM and SVM was 93.09% and 90.20% respectively. The experimental results show that there is a slight difference in detection rate between implicit feature-based and handcrafted feature-based methods. However, feature-based detection systems are generally more reliable because they are based on the features extracted from making extensive and deep knowledge about the problem [102].

Shi et al. [138] presented an ELM (extreme machine learning) method based on features extracted from several resources: construction-based features that describe the structural and lexical properties, TTL-based features, IP-based features, and WHOIS-based features, to characterise domain names. ELM was proposed as a scheme for single-hidden Layer feedforward neural networks. A large dataset was collected from DNS queries received by five DNS servers of Shanghai Jiaotong University. Although they executed a filtering process to reduce the size of the captured DNS traffic and generate a more reliable dataset, an unbalanced dataset was used in this research. The experimental results showed that ELM provided high accuracy detection (96.28%) and fast learning speed. However, this presented method requires data from the DNS response message and access to the information in the WHOIS lookup web service, both of which may increase
4.2 Related work

the time taken to identify the malicious domains.

Song et al. [139] presented a method based on a Random Forest classifier for detecting algorithmically generated domain names. Ten lexical features were extracted from the characters of the domain names by analysing some whitelists and blacklists of domain names. Some of the extracted features are based on n-gram computations and need a dictionary of common English words. The highest achieved precision in this presented method was 93.5%, and the false positive rate was 3.49%.

Truong and Cheng [140] analysed DNS traffic to design a domain fluxing detection system that distinguishes domain names generated by legitimate users and pseudorandom domain names generated by malware. They extracted features from DNS traffic, including similar distribution of alphanumeric letters, the length of domain names and their expected values. Then, they built a classifier using some supervised machine learning algorithms, such as J48, Random Forest, KNN, SVM, and Naïve Bayes. The experimental results show that the decision trees (J48) algorithm achieved the best performance with an overall accuracy of 92.3% and a false positive rate of 4.8%.

The research works discussed above are based on a set of features that are either extracted out of whole DNS communications (i.e., DNS query and response messages) or that require data from external sources, such as WHOIS site. Moreover, most of them are language-dependent and require expensive computations. However, building a system that can efficiently detect malicious domain names based on features extracted from the initial DNS requests would be preferable as it offers potentially earlier detection.
4.3 MaldomDetector high-level architecture

MaldomDetector architecture is presented in this section. As illustrated in Fig. 4.1, it consists of two modules. The first, the Data Preparation Module, is employed to preprocess the incoming DNS request packets and extract the feature vector (V) out of the characteristics of the domain name characters. V is composed of two types of features: basic and derived. The basic feature set \{F_1, F_2, F_3, F_4, F_5, F_6\} is directly obtained from the requested domain name, whereas features \{F_7, \ldots, F_{15}\} are derived from the basic features. F_{16} is a derived feature that represents the output of RMA (Randomness Measuring Algorithm). RMA is a deterministic algorithm that accepts a subset of the basic features, i.e., \{F_1, F_2, F_3, F_4\}, as an input. It measures the randomness in the domain name strings, determining initially whether the domain name is benign or malicious. The details of RMA are presented in Section 4.3.

The second module, the Decision-Making Module, is a machine learning-based domain name classifier. It receives the whole feature vector V \[F_1, \ldots, F_{16}\] as input and classifies it as either a malicious or benign domain name.

4.4 MaldomDetector implementation

4.4.1 Raw dataset collection

A labelled dataset is required to train the classifier of MaldomDetector using a supervised learning technique. It is necessary that this dataset contains samples from several types of DGA families so that the classifier is trained using malicious domains generated by different types of malware. The quality of the dataset is
4.4 MaldomDetector implementation

Figure 4.1: The architecture of MaldomDetector system

crucial to any machine learning task. A high-quality pre-labelled ground truth dataset (i.e. verified malicious and benign data) was selected to train and evaluate MaldomDetector [133] [135] [134]. The malicious domains were collected from a known ground truth dataset, i.e., DGArchive, which was used by preceding researches [133] [134]. Real DGA-based domain names that appeared on the internet have been collected in the DGArchive project [59] [141]. In this project, various DGA families used by different malware types, such as Cryptolocker, Locky, and Bedep, were analysed thoroughly. Also, all the possible domain names generated dynamically by these malware kinds were enumerated or resolved to cover the majority of real and active DGAs.

Besides, a set of DGA-based domains obtained from the Bambenek consulting feeds [142], which is also a known ground truth dataset of DGA-based domain names collected by reverse engineering specific malware families seen in real traf-
4.4 MaldomDetector implementation

fic. Bambenek dataset has also been used in the previous research works, e.g. [134] and [143]. The proposed work in this chapter used Bambenek feeds to create the testing dataset required to test the performance of MaldomDetector on unseen domains. All the collected domain names from DGArchive project and Bambenek feeds were labelled as malicious. The summary of the malware families using DGA collected from DGArchive project besides the DGAs obtained from Bambenek dataset are shown in Table 4.1.

The ground truth benign domains were collected from the Alexa top domains site [144] that lists the domains of the most visited websites on the internet. Alexa top-ranked lists have been utilised in the previous research works [133] [135] [137] [138] [134] [143] because they represent reliable sets of normal domain names. Alexa ranks the websites based on their popularity using different criteria, such as page views and unique visitors, and it provides various lists of top sites, e.g. top 500 and top 1000. In the proposed work, 85,000 domain names from the top 1 million sites have been selected to build the required benign dataset. Since the domains of Alexa are ranked based on their popularity, the first top domains have been selected instead of random selection to build the ground truth benign dataset, because they are more representative of how a normal domain seems [133] [138] [134].

4.4.2 Domain name analysis

The domain names in the raw datasets of Table 4.1 were analysed to extract several potential attributes that can be employed to detect DGA-based domain names. The Top-Level Domains (TLDs) were excluded from this analysis be-
4.4 MaldomDetector implementation

Table 4.1: Malware families employing DGA that were collected from the DGArchive and Bambenek feeds

<table>
<thead>
<tr>
<th>No.</th>
<th>Family</th>
<th>Dataset source</th>
<th>Example</th>
<th>No. of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bimetal</td>
<td>DGArchive</td>
<td>cdcd1e19e9205f369885f972a02dfd60.info</td>
<td>10000</td>
</tr>
<tr>
<td>2</td>
<td>Banjara</td>
<td></td>
<td>ckkwestnessbiophysicalohax.com</td>
<td>10000</td>
</tr>
<tr>
<td>3</td>
<td>Deep</td>
<td>mghhwonojhekup4l.com</td>
<td>okgpgzouwqemvup.ru</td>
<td>7455</td>
</tr>
<tr>
<td>4</td>
<td>Black hole</td>
<td></td>
<td>3seylk6jum4tsd4.biz</td>
<td>10000</td>
</tr>
<tr>
<td>5</td>
<td>China</td>
<td></td>
<td>cpymccvemmmvekyb.com</td>
<td>10000</td>
</tr>
<tr>
<td>6</td>
<td>Cryptolocker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Dnschanger</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Dyre</td>
<td>Bambenek feed</td>
<td>ifhfmmatpw.com</td>
<td>10000</td>
</tr>
<tr>
<td>9</td>
<td>Emotet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Gameover_p2p</td>
<td>DGArchive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Gozi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Locky</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Murolet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Muroletweekly</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Necurs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Padcrypt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Ramnit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Rovnix</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Sphinx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Tuba</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Geodo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>P2P Gameover Zeus</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Post Tover Geoz</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

cause the DGA-based domains use the same TLDs utilised in normal domains. After inspection of many DGA-based domains, it was noticed that most of them contain meaningless strings where it is hard to pronounce or read them. The reason for this difficulty belongs to the existence of several sequential consonant or vowel letters in most DGA-based domain names. Table 4.2 shows some examples of malicious domain names. Fig. 4.2 compares the frequency of the number of sequential consonant letters between the benign and DGA-based domains. According to some sources, like [145], the letter “y” is a particular letter that can represent both types of speech sounds, i.e., vowel and consonant, depending on its position and the letters type surrounding it. Since most of the extracted features in this research are related to the pronunciation, such as the maximum number of sequential consonant and vowel letters, the status of the letter “y” might affect the values of these features and on the detection accuracy. Therefore, the two
4.4 MaldomDetector implementation

cases of “y” have been considered through the implementation of the experiments using RMA, and it is found that considering “y” as a vowel gives high accuracy.

![Figure 4.2: The frequency of the sequential consonant letters: Benign vs DGA-based](image)

<table>
<thead>
<tr>
<th>No.</th>
<th>Domain name</th>
<th>Max sequential consonant letters</th>
<th>Max sequential vowel letters</th>
<th>entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c54103e862d4f27ce7d54d4e66b16bc6fbb.info</td>
<td>8</td>
<td>2</td>
<td>3.5608</td>
</tr>
<tr>
<td>2</td>
<td>dxmalasslettingk.com</td>
<td>6</td>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>3</td>
<td>pgg961dpi68dr8avv4jfl8y1.com</td>
<td>14</td>
<td>3</td>
<td>3.6117</td>
</tr>
<tr>
<td>4</td>
<td>agyyningyrom.com</td>
<td>11</td>
<td>1</td>
<td>3.2808</td>
</tr>
<tr>
<td>5</td>
<td>ykvywe6kypqypqmp_qy.ru</td>
<td>8</td>
<td>1</td>
<td>2.9219</td>
</tr>
<tr>
<td>6</td>
<td>jchbifjyjumny.com</td>
<td>5</td>
<td>6</td>
<td>3.3927</td>
</tr>
<tr>
<td>7</td>
<td>vwupiyxywq.com</td>
<td>7</td>
<td>6</td>
<td>3.5159</td>
</tr>
<tr>
<td>8</td>
<td>rsgqtklovial.su</td>
<td>10</td>
<td>1</td>
<td>3.585</td>
</tr>
<tr>
<td>9</td>
<td>sanewik网游.pw</td>
<td>4</td>
<td>5</td>
<td>3.3186</td>
</tr>
<tr>
<td>10</td>
<td>onzggblewral65e6rinfo</td>
<td>12</td>
<td>1</td>
<td>3.6492</td>
</tr>
<tr>
<td>11</td>
<td>n周wuyspmpjpt work</td>
<td>6</td>
<td>6</td>
<td>3.2418</td>
</tr>
<tr>
<td>12</td>
<td>a37fi86mzfl6boqj4l6huyji656v2rlhikz74</td>
<td>10</td>
<td>3</td>
<td>4.0141</td>
</tr>
<tr>
<td>13</td>
<td>ccon78js2721fbb4b58f2359b82858e</td>
<td>5</td>
<td>2</td>
<td>3.58</td>
</tr>
</tbody>
</table>

Shannon’s entropy can serve as a valuable metric to measure the randomness in the distribution of the characters within a given domain name [130]. It can be computed using equation (4.1).

\[ H = - \sum_{i=1}^{n} p_i \log p_i \tag{4.1} \]
where \( p_i \) is probability distribution \( p_n = (p_1, ..., p_n) \) with \( p_i \geq 0 \) for \( i = 1, ..., n \) and \( \sum_{i=1}^{n} p_i = 1 \)

Entropy exceeding a threshold value can be a helpful indicator to identify DGA-based domain names. The normal distribution curves of entropy values for the DGA-based and benign domains are illustrated in Fig. 4.3. It is obvious from Fig. 4.3 that there is a clear differentiation in the probability distribution of the entropy values between the curves, which may suggest entropy is a useful feature to classify the domains. Thus, the entropy value for the domain name was selected as a feature to recognise the DGA-based domain names. The entropy values of some domain names are shown in Table 4.2.

\[ \text{Figure 4.3: The normal distribution curve for the Malicious and benign domains} \]

4.5 Randomness Measuring Algorithm (RMA)

RMA was constructed to identify malicious domain names initially through measuring the randomness in the domain name’s characters. It is an enhanced version
of my earlier work in chapter 3. RMA is a deterministic algorithm that accepts a subset of the basic features as an input, i.e., the maximum sequential consonants, the maximum sequential vowels, the entropy, and the domain name length. Then, RMA processes these features according to the threshold values shown in Fig. 4.4. These threshold values were determined based on the analysis of the domain names, conducted in Section 4.2.

![Figure 4.4: The flow chart of Randomness Measuring Algorithm (RMA)](image)

After examining the probability distribution of the entropy values in Fig. 4.3, three base points have been recognised in the distribution curves that can be used in RMA as thresholds, as displayed in Fig. 4.5. The following are the key points:

- The ratio of the benign domains that have entropy \( H \leq 2.0 \) is 18.99%, while the ratio of the DGA-based domains is 0.093%.
The ratio of the DGA-based and benign domains that have $H > 3.24$ is 77.83% and 10.44% respectively.

The values $2 < H \leq 3.24$ occur in the benign and malicious domains together in different proportions, making the recognition of these domains a non-trivial task, which requires additional complicated feature(s). Since the aim of this research work is building a detection system using uncomplicated features, two simple features, i.e., maximum sequential consonants & maximum sequential vowels, were added to the rules of RMA. These added features will help increase the differentiation between sample distributions and decrease the false rate. The supplementary rule states that most of the DGA-based domains have four or more sequential consonants or vowels, while a lot of the benign domains have fewer than four sequential consonants or vowels.

![Figure 4.5: Ratio of the entropy values in DGA-based and benign domains](image)

RMA was implemented in Python and evaluated employing various DGA families. RMA does not contain any parameters related to dictionary words or frequency distribution of letters; therefore, it is considered a language-independent
4.5 Randomness Measuring Algorithm (RMA)

algorithm. RMA has been applied on 20 types of malware families of DGArchive dataset, and the results are shown in Table 4.3. Moreover, RMA was applied on 85,000 of Alexa domain names, where the results demonstrate detection accuracy and FPR of 83.14% and 0.1686 respectively.

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Size (sample)</th>
<th>Classified samples</th>
<th>Accuracy %</th>
<th>False rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bamital</td>
<td>10000</td>
<td>99.38</td>
<td>62</td>
<td>99.38</td>
</tr>
<tr>
<td>2</td>
<td>Banjori</td>
<td>10000</td>
<td>9057</td>
<td>943</td>
<td>90.57</td>
</tr>
<tr>
<td>3</td>
<td>Bedep</td>
<td>7458</td>
<td>7246</td>
<td>212</td>
<td>97.1574</td>
</tr>
<tr>
<td>4</td>
<td>Blackhole</td>
<td>732</td>
<td>727</td>
<td>5</td>
<td>99.32</td>
</tr>
<tr>
<td>5</td>
<td>Chinad</td>
<td>10000</td>
<td>9881</td>
<td>119</td>
<td>98.81</td>
</tr>
<tr>
<td>6</td>
<td>Cryptolocker</td>
<td>10000</td>
<td>9423</td>
<td>577</td>
<td>94.25</td>
</tr>
<tr>
<td>7</td>
<td>Dnschanger</td>
<td>10000</td>
<td>8121</td>
<td>1879</td>
<td>81.21</td>
</tr>
<tr>
<td>8</td>
<td>Dyre</td>
<td>10000</td>
<td>9988</td>
<td>12</td>
<td>99.88</td>
</tr>
<tr>
<td>9</td>
<td>Emotet</td>
<td>10000</td>
<td>9900</td>
<td>100</td>
<td>99.0</td>
</tr>
<tr>
<td>10</td>
<td>Gameover p2p</td>
<td>10000</td>
<td>10000</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>11</td>
<td>Gozi</td>
<td>10000</td>
<td>7832</td>
<td>2168</td>
<td>78.32</td>
</tr>
<tr>
<td>12</td>
<td>Locky</td>
<td>10000</td>
<td>8338</td>
<td>1662</td>
<td>83.38</td>
</tr>
<tr>
<td>13</td>
<td>Murofet</td>
<td>10000</td>
<td>9799</td>
<td>201</td>
<td>97.99</td>
</tr>
<tr>
<td>14</td>
<td>Murofetweekly</td>
<td>10000</td>
<td>10000</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>15</td>
<td>Necurs</td>
<td>10000</td>
<td>8957</td>
<td>1043</td>
<td>89.57</td>
</tr>
<tr>
<td>16</td>
<td>Padcrypt</td>
<td>10000</td>
<td>9138</td>
<td>862</td>
<td>91.38</td>
</tr>
<tr>
<td>17</td>
<td>Ramnit</td>
<td>10000</td>
<td>8970</td>
<td>1030</td>
<td>89.7</td>
</tr>
<tr>
<td>18</td>
<td>Rovnix</td>
<td>10000</td>
<td>9963</td>
<td>37</td>
<td>99.63</td>
</tr>
<tr>
<td>19</td>
<td>Sphinx</td>
<td>10000</td>
<td>9853</td>
<td>147</td>
<td>98.53</td>
</tr>
<tr>
<td>20</td>
<td>Tinba</td>
<td>10000</td>
<td>9298</td>
<td>702</td>
<td>92.98</td>
</tr>
<tr>
<td>21</td>
<td>Benign</td>
<td>85000</td>
<td>70667</td>
<td>14333</td>
<td>83.14</td>
</tr>
</tbody>
</table>

Where:

\[
Accuracy = \frac{\text{number of correctly classified samples}}{\text{number of total samples}} \tag{4.2}
\]

\[
\text{False rate (negative or positive)} = \frac{\text{number of incorrectly classified samples}}{\text{number of total samples}} \tag{4.3}
\]
The evaluation results in Table 4.3 show that RMA provides high accuracy in detecting most DGA families and moderate accuracy on a few types, such as dnschanger and gozi. The detection process becomes difficult when a malware generates DGA-domain names containing meaningful words or low randomness in their strings. However, the majority of the DGA families generate random domain names because they are based on random algorithms to generate a large number of domains. Besides, the botmaster or attacker must register one or a few domains in advance to enable the malware to make a connection with the C&C server. Therefore, the malware developers usually avoid using wordlists in algorithmically generated domains to avoid the conflict with legitimate domains throughout the registration process. It is also noteworthy that although some DGA families in Table 4.1 have domain names that contain alphanumeric characters, such as Dyre, Bamital, and Murofetweekly, they still have some sequential consonants and vowels within their characters. As well, most of these domains have an entropy value \( (H) > 3.24 \), as indicated in some examples of Table 4.2. These families have been detected by RMA with high accuracy, as demonstrated in Table 4.3.

In order to improve the detection accuracy and build a system that can address the low randomness in the domain name characters, a machine learning-based system (MaldomDetector) has been built. MaldomDetector employs the output of RMA along with other engineering features to detect malicious domains efficiently. The next section demonstrates the steps of building the system.
4.6 Building and evaluating the domain name classifier (MaldomDetector)

4.6.1 Feature extraction and selection

Based on the domain name analysis presented in Section 4.2, two types of features have been extracted. The first type is the basic features, and the second is derived features. The basic features include the entropy, maximum sequential consonants, maximum sequential vowels, the total number of consonants, the total number of vowels, and the length of a given domain name. While the derived features have been computed from the basic features, based on the domain knowledge, as shown in Table 4.4.

Table 4.4: The basic and derived features

<table>
<thead>
<tr>
<th>Feature type</th>
<th>Feature name</th>
<th>Feature type</th>
<th>Feature name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>F1 entropy</td>
<td>Derived</td>
<td>F12 ratio-max-sequential-vowels-to-length-domain</td>
</tr>
<tr>
<td></td>
<td>F2 max-sequential-consonants</td>
<td></td>
<td>F13 ratio-max-sequential-consonants-to-consonants</td>
</tr>
<tr>
<td></td>
<td>F3 max-sequential-vowels</td>
<td></td>
<td>F14 ratio-max-sequential-vowels-to-vowels</td>
</tr>
<tr>
<td></td>
<td>F4 length-domain</td>
<td></td>
<td>F15 ratio-max-sequential-consonants-to-max-sequential-vowels</td>
</tr>
<tr>
<td></td>
<td>F5 consonants</td>
<td></td>
<td>F16 Randomness</td>
</tr>
<tr>
<td></td>
<td>F6 vowels</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>The entropy of a given domain name string.</td>
</tr>
<tr>
<td>F2</td>
<td>Maximum sequential consonant letters found within a given domain.</td>
</tr>
<tr>
<td>F3</td>
<td>Maximum sequential vowel letters found within a given domain.</td>
</tr>
<tr>
<td>F4</td>
<td>Length of the domain name string.</td>
</tr>
<tr>
<td>F5</td>
<td>The total number of consonant letters of a given domain.</td>
</tr>
<tr>
<td>F6</td>
<td>The total number of vowel letters of a given domain.</td>
</tr>
<tr>
<td>F7</td>
<td>The ratio of the entropy to the length of a given domain.</td>
</tr>
<tr>
<td>F8</td>
<td>The ratio of the total number of consonant letters to the total number of vowel letters of a given domain.</td>
</tr>
<tr>
<td>F9</td>
<td>The ratio of the total number of consonant letters to the length of a given domain.</td>
</tr>
<tr>
<td>F10</td>
<td>The ratio of the total number of vowel letters to the length of a given domain.</td>
</tr>
<tr>
<td>F11</td>
<td>The ratio of the maximum sequential consonant letters to the length of a given domain.</td>
</tr>
<tr>
<td>F12</td>
<td>The ratio of the maximum sequential vowel letters to the length of a given domain.</td>
</tr>
<tr>
<td>F13</td>
<td>The ratio of the maximum sequential consonant letters to the total number of consonants of a given domain name.</td>
</tr>
<tr>
<td>F14</td>
<td>The ratio of the maximum sequential vowel letters to the total number of vowels of a given domain.</td>
</tr>
<tr>
<td>F15</td>
<td>The ratio of the maximum sequential consonant letters to the maximum sequential vowel letters of a given domain.</td>
</tr>
<tr>
<td>F16</td>
<td>The output of RMA algorithm.</td>
</tr>
</tbody>
</table>

Feature importance was computed for all features in Table 4.4. A standard technique for calculating feature importance is to apply the correlation that is formally referred to as Pearson’s correlation coefficient (PCC) [146]. The formula
4.6 Building and evaluating the domain name classifier

(MaldomDetector)

of PCC for two variables is illustrated in the following equation.

\[
PCC = \frac{\text{cov}(x, y)}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2(y_i - \bar{y})^2}}
\] (4.4)

Where:

\(\text{cov}\) is the covariance for two variables \(x, y\). \(\sigma_x\) is the standard deviation of \(x\). \(\sigma_y\) is the standard deviation of \(y\). \(\bar{x}, \bar{y}\) represent the mean of \(x\) and \(y\) respectively.

\(\sigma\) represents the square root of its covariance. The covariance (\(\text{cov}\)) can be used to measure the linear relationship between two variables, i.e., it indicates how much the two variables vary together. In this work, the variable \(x\) can be any features of Table 4.4, while the variable \(y\) means the class.

PCC between each extracted feature in Table 4.4 and the response (class) was calculated using the SciPy library of Python. The rank and score of the importance for each feature are indicated in Fig. 4.6. The score of the feature importance can be utilised to reduce the number of extracted features through selecting those that have an importance score greater than a specific value (i.e., threshold) to be selected features. Since there is no specific rule to assign the threshold [147], the threshold with 0.2 has been chosen in this case. Therefore, the features that have an importance score greater than 0.2, i.e., F16, F7, F4, F5, F1, F2, F10, F12, F15, F8, F6, and F3, were selected. While the features F9, F11, F14, and F13 were excluded because their score is less than 0.2, as illustrated in Fig. 4.6. Since the features F16, F7, F4, F5, F1, and F2 have an importance score greater than 0.5, they have been considered the most related features for the problem in this research work. Table 4.5 displays the selected features that will be used to build the classifier.
4.6 Building and evaluating the domain name classifier
(MaldomDetector)

Figure 4.6: The plot of the feature importance

Table 4.5: The selected features

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature</th>
<th>Feature name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F1</td>
<td>entropy</td>
</tr>
<tr>
<td>2</td>
<td>F2</td>
<td>max-sequential-consonants</td>
</tr>
<tr>
<td>3</td>
<td>F3</td>
<td>max-sequential-vowels</td>
</tr>
<tr>
<td>4</td>
<td>F4</td>
<td>length-domain</td>
</tr>
<tr>
<td>5</td>
<td>F5</td>
<td>consonants</td>
</tr>
<tr>
<td>6</td>
<td>F6</td>
<td>vowels</td>
</tr>
<tr>
<td>7</td>
<td>F7</td>
<td>ratio-entropy-to-length-domain</td>
</tr>
<tr>
<td>8</td>
<td>F8</td>
<td>ratio-consonants-to-vowels</td>
</tr>
<tr>
<td>9</td>
<td>F10</td>
<td>ratio-vowels-to-length-domain</td>
</tr>
<tr>
<td>10</td>
<td>F12</td>
<td>ratio-max-sequential-vowels-to-length-domain</td>
</tr>
<tr>
<td>11</td>
<td>F15</td>
<td>ratio-max-sequential-consonants-to-max-sequential-vowels</td>
</tr>
<tr>
<td>12</td>
<td>F16</td>
<td>Randomness</td>
</tr>
</tbody>
</table>

4.6.2 Building a labelled dataset

Several Python modules have been written to build the required dataset for training and evaluating MaldomDetector. The function of the main module is to extract the values of the features given in Table 4.4 out of a large number of malicious and benign domain names selected from the raw datasets in Table 4.1, then storing them in CSV files. After that, the dataset was cleaned by handling the incorrect data and removing the duplicated samples. A group of unduplicated domains was selected from each DGA family of DGArchive data in Table 4.1 and
4.6 Building and evaluating the domain name classifier (MaldomDetector)

labelled as malicious domains to create a malicious (i.e., DGA-based) dataset. In contrast, a set of domains was selected from the Alexa site and labelled as benign domains to create a benign dataset. These malicious and benign datasets have the same number of samples, i.e., balanced dataset, and they were combined to form the final dataset required to train and evaluate MaldomDetector. Table 4.6 summarises the details of this dataset.

Table 4.6: The summary of the dataset used to build MaldomDetector

<table>
<thead>
<tr>
<th>Raw dataset name</th>
<th>Type</th>
<th>No. of samples</th>
<th>Dataset size</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGArchive</td>
<td>Malicious</td>
<td>85000</td>
<td>170000</td>
</tr>
<tr>
<td>Alexa (top sites)</td>
<td>Benign</td>
<td>85000</td>
<td></td>
</tr>
</tbody>
</table>

4.6.3 Model training

I used the classification learner app of MATLAB R2019a [148] to train the domain name classifier of Fig. 4.1 utilising the dataset of Table 4.6. The k-fold cross-validation option was chosen to train and evaluate the machine learning models in order to protect against overfitting and provide an accurate model performance estimation [149] [150]. The default value of k in the classification learner is 5; however, it was set to 10, as it gave optimal accuracies. At first, I explored all the learning classification algorithms that available in the classification learner, such as support vector machine (SVM) and decision tree, to train the model using all the selected features in Table 4.5. The hyperparameters of the selected algorithms were tuned manually to enhance their performance. However, it has been found that the algorithms provided the best performance when the default hyperparameters setting were used. After training a set of models using the default setting of the hyperparameters, five best models, i.e., Decision tree (Fine
Tree), Ensemble (Boosted Tree), Naive Bayes (Gaussian), and KNN (Coarse), were selected based on evaluation metrics. Fig. 4.7 illustrates the training of the models using the classification learner app of MATLAB.

![Figure 4.7: Training of the machine learning models using the classification learner application of MATLAB](image)

4.6.4 Evaluating the classifier

Binary classification metrics [102], such as accuracy, FPR, precision, recall and F1 score, have been used to evaluate the performance of the trained machine learning models. These metrics are obtained from the confusion matrix and defined as indicated in the equations 4.5 to 4.9. The evaluation results of the best models that were chosen after performing the training and validation task are shown in Table 4.7. Fig. 4.8 demonstrates the models’ performance.

\[
\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (4.5)
\]
### 4.6 Building and evaluating the domain name classifier
(MaldomDetector)

Table 4.7: The evaluation results of the models

<table>
<thead>
<tr>
<th>Algorithm name</th>
<th>Hyperparameter</th>
<th>Accuracy (10-fold cross validation)</th>
<th>FPR</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision tree</td>
<td>- Max no. of splits: 100</td>
<td>94.39 %</td>
<td>0.0533</td>
<td>0.9464</td>
<td>0.9410</td>
<td>0.9437</td>
</tr>
<tr>
<td>(Fine Tree)</td>
<td>- Split criterion: Gini’s diversity index</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Surrogate decision splits: off</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ensemble</td>
<td>- Ensemble method: Ada boost</td>
<td>94.42 %</td>
<td>0.0545</td>
<td>0.9454</td>
<td>0.9429</td>
<td>0.9441</td>
</tr>
<tr>
<td>(Boosted Tree)</td>
<td>- max no. of splits: 20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- No. of learners: 30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Learning rate: 0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Na&quot;ıve Bayes</td>
<td>Distribution names: Gaussian</td>
<td>92.33 %</td>
<td>0.0816</td>
<td>0.9192</td>
<td>0.9282</td>
<td>0.9236</td>
</tr>
<tr>
<td>(Gaussian)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>- Kernel function: linear</td>
<td>94.14 %</td>
<td>0.0536</td>
<td>0.9450</td>
<td>0.9363</td>
<td>0.9411</td>
</tr>
<tr>
<td>(Linear)</td>
<td>- Box constraint level: 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Kernel scale: auto</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>- No. of neighbours: 100</td>
<td>94.52 %</td>
<td>0.0402</td>
<td>0.9586</td>
<td>0.9505</td>
<td>0.9443</td>
</tr>
<tr>
<td>(Coarse)</td>
<td>- Distance metric: Euclidean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Distance weight: equal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
FPR = \frac{FP}{FP + TN} \quad (4.6)
\]

\[
Precision = \frac{TP}{TP + FP} \quad (4.7)
\]

\[
Recall = \frac{TP}{TP + FN} \quad (4.8)
\]

\[
F1score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4.9)
\]

Where: TP is True Positive, FP is False Positive, TN is True Negative, and FN is False Negative.

The Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) for the selected models are shown in Fig. 4.9. ROC can be used to assess the performance of the model over the entire operating range. It is apparent from Fig. 4.9 that the models have approximately similar performance.
4.6 Building and evaluating the domain name classifier
(MaldomDetector)

Figure 4.8: Performance metric of the models.

Figure 4.9: ROC curve and AUC of the selected models: (a) Fine Tree (b) Boosted Tree (c) Gaussian Na"ive Bayes (d) Linear SVM (e) Coarse KNN
4.6 Building and evaluating the domain name classifier (MaldomDetector)

Although using cross validation procedure is considered sufficient to evaluate the performance of the models [102] [151], an additional assessment for the models’ performance using a second dataset, i.e., Bambenek feeds, which was not used during the models’ training process. Some DGA families from the Bambenek dataset were selected to make the evaluation. While the benign domains were collected from the dataset that was already used in [135], and differ from benign domains of Table 4.6. Table 4.8 demonstrates the summary of the testing dataset, whereas Table 4.9 illustrates the supplementary evaluation results of the models. The results prove that MaldomDetecor is an efficient and reliable system for detecting DGA-based domains.

Table 4.8: The dataset used for further testing

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>No. of samples</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geodo DGA</td>
<td>Malicious</td>
<td>8500</td>
<td>17000</td>
</tr>
<tr>
<td>P2P Gameover Zeus DGA</td>
<td>Malicious</td>
<td>8500</td>
<td></td>
</tr>
<tr>
<td>Post Tover Goz DGA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clean-alexa-32k</td>
<td>Benign</td>
<td>8500</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.9: Testing results of the selected models

<table>
<thead>
<tr>
<th>Algorithm name</th>
<th>Accuracy %</th>
<th>FPR</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision tree (Fine Tree)</td>
<td>97.21</td>
<td>0.0542</td>
<td>0.9485</td>
<td>0.9985</td>
<td>0.9728</td>
</tr>
<tr>
<td>Ensemble (Boosted Tree)</td>
<td>97.19</td>
<td>0.0556</td>
<td>0.9473</td>
<td>0.9994</td>
<td>0.9726</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>96.26</td>
<td>0.0738</td>
<td>0.9312</td>
<td>0.9989</td>
<td>0.9639</td>
</tr>
<tr>
<td>Linear SVM</td>
<td>97.38</td>
<td>0.0508</td>
<td>0.9516</td>
<td>0.9984</td>
<td>0.9744</td>
</tr>
<tr>
<td>Coarse KNN</td>
<td>97.82</td>
<td>0.0406</td>
<td>0.9609</td>
<td>0.9969</td>
<td>0.9786</td>
</tr>
</tbody>
</table>

118
4.7 Discussion

Since the datasets utilised in preceding research works are not identical, and there is no standard DGA to generate domain names for comparison; thus, it is difficult to compare MaldomDetector with the systems presented in the related work section. Moreover, the variety in DGA implementations creates a wide fluctuation in the detection results when applying the detection approaches on these DGAs. This section will only summarise several properties of 11 detection methods discussed in Section 2, as shown in Table 4.10. As illustrated in Table 4.10, MaldomDetector has some advantages while it keeps high accuracy.

Table 4.10: The properties of the DGA-based domains detection systems

<table>
<thead>
<tr>
<th>No.</th>
<th>Method name</th>
<th>Source</th>
<th>Accuracy %</th>
<th>Language independency</th>
<th>External source independency</th>
<th>Does not need DNS response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Knowledge based random forest algorithm</td>
<td>[139]</td>
<td>N/Av</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>Hidden Markov Model (HMM)</td>
<td>[136]</td>
<td>91.52</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>Handcrafted features based J48</td>
<td>[140]</td>
<td>92.3</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>Handcrafted features based</td>
<td>[147]</td>
<td>N/Av</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>5</td>
<td>Implicit features based (deep neural network)</td>
<td>[137]</td>
<td>N/Av</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>6</td>
<td>Extreme machine learning</td>
<td>[138]</td>
<td>96.29</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>Machine learning based on masked n-grams</td>
<td>[135]</td>
<td>98.91</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>8</td>
<td>Deep neural network</td>
<td>[134]</td>
<td>&gt; 98.0</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>9</td>
<td>Proposed n-CBDC (n = 2, bigram)</td>
<td>[133]</td>
<td>94.15</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>10</td>
<td>Proposed n-CBDC (n = 3, trigram)</td>
<td>[133]</td>
<td>98.29</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>11</td>
<td>MaldomDetector</td>
<td>this research</td>
<td>97.82</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

The methods [133] [134] [135] [139] use a probabilistic language model, i.e., n-gram that assigns probabilities to every n-sequence of characters. The model estimates these probabilities by calculating the relative frequency for each n-sequence of characters within a given dataset. However, this means the model relies heavily on the training dataset and hence inefficient in dealing with unseen
types of DGA-based domain names [152] [153]. The presented method in [140] depends on the frequency distribution of alphanumeric letters of the domain names, and [137] (Handcrafted features-based) uses a dictionary matching score to measure the degree that a word in a domain name can be explained by the dictionary. MaldomDetector based on a set of pronunciation-based features, which does not rely on the training dataset. Furthermore, MaldomDetector does not adopt any probabilistic language model, i.e., a language-independent system.

The method proposed in [138] also uses two features that require access to an external site, i.e., WHOIS lookup service, to obtain data before detecting the malicious domains. However, getting this data adds a time delay and requires the system to be always online to function correctly. The methods presented in [136] and [138] require data from the DNS response packet, such as TTL (Time To Live) and the number of IP addresses, before making any decision. Although this data can be useful to reduce the false positive rates, it adds a time delay that can be exploited by the malware to make contact with its C&C server or exfiltrate information before the security systems can detect it.

This chapter aims to build a detection system requires little information to check the status of the requested domain names while attempting to detect malicious communications early and thus mitigating the risks to the network. Therefore, MaldomDetector was designed to work without using any data from an external site or the DNS response packet to classify the domains, even if they are possibly useful in the detection process. MaldomDetector has been built to depend only on a deterministic algorithm and computationally inexpensive features extracted out of the domain name’s characters. Thus, MaldomDetector can check the status of the domain names before sending them to the DNS server to
resolve them, as a first layer for detection.

4.8 Summary

Many domain names that belong to a number of DGA families were analysed, and two types of features, i.e., basic & derived, were extracted. An effective detection method, MaldomDetector, to detect algorithmically generated malicious domain names is introduced and prototyped. MaldomDetector uses an algorithm, i.e., RMA, to measure the randomness in the domain name strings. In the MaldomDetector system, the RMA’s output along with a set of basic and derived features, which extracted out of the initial DNS query message, are sent as a feature vector to a machine learning classifier for processing and classification. Some classification algorithms (Decision Tree, Ensemble, Naïve Bayes, and KNN) have been explored to build the DGA-based classifier.

Building upon the state-of-the-art on malicious domain name detection, MaldomDetector does not utilise any probabilistic language models. Instead, it uses a character-based method to detect DGA-based domain names. MaldomDetector performs measurements solely on the DNS request packet and does not need to wait for the DNS response packet to extract additional features or require information from any external sources. Detailed analyses validated that MaldomDetector can detect effectively various types of DGA-based domains generated by several types of malware, and the detection accuracy achieved by MaldomDetector is $\sim 98\%$. MaldomDetector is well suited as an early warning system for malicious DNS communications of a potentially compromised host.
Chapter 5

Multi-Feature and Multi-Classifier ML Approach for Network-based Ransomworm Detection

5.1 Introduction

In 2017, WannaCry ransomware launched a worldwide cyberattack and targeted computers worldwide by encrypting their data and demanding ransom. WannaCry belongs to ransomworm, which is a type of ransomware that combines the payload of ransomware with the propagation feature of a computer worm. It easily spreads to other hosts within the same network and subsequently to other networks by taking advantage of vulnerabilities within communication protocols. Thus, ransomworm requires a network-based detection approach and a thorough
5.2 Related work

State-of-the-art research investigated different detection or prevention methods based on various characteristics to mitigate WannaCry. Lee et al. [154] analysed the ransomware attack chain using three serious families, including WannaCry. They presented a host-based method called Moving Target Defense (MTD) to protect the user’s files. Many families of ransomware examine only the extensions of the victim’s files before encrypting them to reduce the computation cost. Thus, MTD changes the file’s extensions arbitrarily and continuously in the system, so that they become different from the original extensions, for example, .docx changed to .juev. However, there are several families of ransomware can attack the files regardless of their extensions, such as Zerolocker and Razy [155].

Akbanov et al. [156] presented a Software-Defined Networking (SDN) framework to detect and mitigate the WannaCry network activities. The authors have
used the OpenFlow protocol for analysing the network traffic. The DNS traffic (kill switches) and IP addresses were inspected based on dynamic blacklisting to detect any malicious domain names or IP addresses used during the WannaCry network communications. The lists of the malicious domains and IP addresses were specified using a local file or online blacklist database, where they are saved in CSV (Comma-Separated Values) format and loaded from the SDN controller. However, this system is not effective to detect previously unknown or new malicious IP addresses or domain names which are not present in blacklist database. Furthermore, WannaCry uses random IP addresses to spread to other networks, and the kill switch feature is no longer employed in many of its variants. Zhang et al. [157] proposed a static analysis approach to map ransomware into families. Eight families of ransomware, including WannaCry, were selected in this work. The opcode sequences from the ransomware samples were extracted and transformed into n-gram sequences. The term frequency-inverse document frequency (TF-IDF) was utilised to select feature n-grams of various lengths that can discriminate families. Then, the TF values of the feature n-grams were treated as a feature vector and fed into a machine learning classifier to classify the ransomware families. This approach achieved an accuracy of 91.43%.

Zimba and Mulenga [158] analysed the network characteristics of WannaCry ransomware using reverse engineering techniques and dynamic analysis. The authors analysed source codes of some WannaCry samples and examined a set of Indicators of Compromise (IOCs) using IDA Pro and Ollydbg tools. They investigated certain network characteristics of WannaCry and its network propagation techniques. However, the presented work lacks a detection system for identifying such malicious network activities.
5.2 Related work

Daku et al. [123] presented behavioural-based classification approach to classify some ransomware families using a machine learning technique. A set of host-based features were identified from the study of ransomware behavioral reports. The authors obtained these behavioural reports about ten different families (150 samples) including WannaCry from virustotal.com. They selected 12 host-based behavioural attributes to build a classifier system using machine learning approach. The highest classification accuracy obtained from the experiments is 78%. Alam et al. [159] proposed a host-based detection tool called RAPPER (Ransomware Prevention via performance counters), which is based on the Hardware Performance Counters (HPCs) as a data source for ransomware detection. HPCs are a set of special-purpose registers embedded into modern microprocessors (CPU) to observe the hardware performance in computer systems dynamically. The performance statistics of the system are different in the presence of a ransomware due to continuous encryption of the system files. The detection of high encryption activities depends on the HPC events observed about the encryption activities. RAPPER uses artificial neural networks and fast Fourier transfer to detect the ransomware attack where it considers any deviation from the normal operating behaviour of the system as an anomaly.

Hsiao and Kao [160] used static analysis techniques to explore WannaCry’s processes and function execution. They used the reverse engineering tool, IDA Pro, to disassemble the WannaCry samples. They did not analyse the network traffic to extract network features. Kao and Hsiao [161] presented a dynamic analysis method to extract IOCs from the file system, running processes, registry utilisation, and network activities for the WannaCry attack. This work provided information about WannaCry network processes that run inside the host, using
5.3 How a ransomworm works

This section investigates network activities of ransomworm for features extraction and for designing an effective ransomware detection system. Two families, i.e., WannaCry & NotPetya, are studied as examples of ransomworm work. The
network actions of them are illustrated briefly in the subsections below.

5.3.1 WannaCry network activities

The network activities of WannaCry can be summarised as follows:

- The WannaCry attack involves multiple components, such as a dropper, encrypter, and decrypter.

- The dropper first attempts to make an HTTP connection to a hard-coded unusual domain name, e.g., iuqerfsodp9ifajposdfjgosurijfaewrwegwea.com. If the connection is successful, it will terminate the installation, i.e., ends the process mssecsvc.exe. This mechanism, called a kill switch, was designed by the malware author to stop WannaCry from spreading out of control [165].

- WannaCry was designed to be self-propagating like a worm using Eternal-Blue, which is an embedded exploit that targets a known vulnerability in the Windows SMB protocol and enables WannaCry to infect other machines through NetBIOS [166]. NetBIOS is a service that allows applications on separate computers to communicate within a LAN. WannaCry checks the IP address of the infected computer and attempts to infect other unpatched hosts within the same network by exploiting the SMB vulnerability. In the same way, it also propagates to other interconnected networks by scanning external IP addresses randomly and attempting to communicate with them via SMB protocol [167]. The SMB protocol operates in one of the following three ways [168]:
5.3 How a ransomworm works

- Over the NetBIOS service via UDP (User Datagram Protocol) ports 137 and 138.
- Over the NetBIOS service as well via TCP (Transport Control Protocol) ports 139 and 137.
- Over the direct TCP/IP MS Networking access via TCP port 445, without the need for a NetBIOS layer.

- WannaCry installs the Tor anonymous networking software on the infected system in the folder of the malware’s working directory. The local Tor server is renamed and executed as taskhsvc.exe [161].

- WannaCry uses encrypted Tor channels for C&C communications. WannaCry produces several files including the ”c.wnry” file in the current directory when it is executed. This file (i.e., c.wnry) contains the configurations for the Tor C&C addresses, Bitcoin addresses, and other data. WannaCry installs the Tor network software on the infected system within the working directory [166]. The local Tor server establishes a SOCKS5 proxy on the loopback interface that listens on TCP port 9050. WannaCry connects to this proxy and attempts to contact the configured C&C hidden services. After a successful communication with its C&C server, WannaCry uses a custom encrypted protocol running on TCP port 80 over the Tor circuit to send the encryption keys allowing the attacker to communicate with the victims and to check the payment status [169].
5.3.2 NotPetya network activities

NotPetya is a variant of Petya ransomware family that appeared in June 2017 and was used for a global cyberattack [170]. It combines ransomware with the ability to spread itself across a network. NotPetya spreads to Microsoft Windows machines via the EternalBlue exploit, which was also used by WannaCry. When NotPetya infects a machine, it performs certain automated malicious activities, such as dropping files and attempting to propagate across a network. It can spread using the following methods [171] [172]:

- Network node enumeration. NotPetya gathers a list of endpoints’ IP addresses and attempts to connect to them via SMB ports 445 and 139.
- SMB copy and remote execution. NotPetya steals credentials or re-uses existing active sessions to copy and execute the payload to a remote host.
- Uses a file-share to infect other machines on the same network.
- Exploits SMB through EternalBlue.

NotPetya does not require a connection with the C&C server during its attack [173], but it can steal private data from the victim’s machine and send it to its C&C server via HTTP post.

5.4 Dataset description

The proposed MFMCNS was trained and evaluated using a real long-term ransomware traffic dataset, created in the Czech Technical University (CTU) [127]. The dataset is publicly available as a part of the Malware Capture Facility Project.
5.5 Network traffic analysis and feature extraction

(MCFP) [126]. In the MCFP dataset, real malware samples were executed in a real network environment for extended periods. The original Packet Capture (PCAP) files were used in the present work to extract network features and build the required dataset for training and evaluating the proposed MFMCNS system. Furthermore, the MCFP dataset contains a significant number of normal files captured during different periods. These files contain traffic generated by different benign services running on real computer systems. Table 5.1 shows the information summary about the malicious and benign PCAP files of the MCFP dataset utilised in this chapter. The third column indicates the names of the malware and normal captured files designated by me, while the fourth column indicates the ID of the capture folders in MCFP dataset. The number of WannaCry PCAP files are more than the number of NotPetya files; therefore, this research work depends more on WannaCry traffic to analyse the network activities of ransomworm.

5.5 Network traffic analysis and feature extraction

This section, analyses the spreading behaviour of WannaCry and NotPetya to understand their propagation methods and extract a set of network attributes. The malicious PCAP files from Table 5.1 were analysed based on the domain knowledge to extract a set of network features that describe the propagation behaviour of the ransomworm. Moreover, the network protocol analyser Wireshark [174] was used to inspect and analyse the network protocols visually. Several Python scripts have been written to process the raw PCAP files. Since the size of the
5.5 Network traffic analysis and feature extraction

Table 5.1: Brief information of PCAP files from MCFP dataset used in this chapter

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>File name</th>
<th>MCFP ID</th>
<th>Size (KB)</th>
<th>Capturing time (sec)</th>
<th>Hash value (MD5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WannaCry ransomware</td>
<td>wannacry252.pcap</td>
<td>CTU-Malware-Capture-Botnet-252-1</td>
<td>500</td>
<td>501.282</td>
<td>e16b903789e41697ecab21ba6e14fa2b</td>
</tr>
<tr>
<td>2</td>
<td>WannaCry ransomware</td>
<td>wannacry253.pcap</td>
<td>CTU-Malware-Capture-Botnet-253-1</td>
<td>10,914</td>
<td>895.954</td>
<td>d5dcd28612f4d6ffca0cfeaefd606bcf</td>
</tr>
<tr>
<td>3</td>
<td>WannaCry ransomware</td>
<td>wannacry254.pcap</td>
<td>CTU-Malware-Capture-Botnet-254-1</td>
<td>3668</td>
<td>3664.729</td>
<td>84c82835a5d21bbcf75a61706d8ab549</td>
</tr>
<tr>
<td>4</td>
<td>WannaCry ransomware</td>
<td>wannacry284.pcap</td>
<td>CTU-Malware-Capture-Botnet-284-1</td>
<td>7795</td>
<td>981.589</td>
<td>e16b903789e41697ecab21ba6e14fa2b</td>
</tr>
<tr>
<td>5</td>
<td>WannaCry ransomware</td>
<td>wannacry285.pcap</td>
<td>CTU-Malware-Capture-Botnet-285-1</td>
<td>7456</td>
<td>1014.20</td>
<td>e16b903789e41697ecab21ba6e14fa2b</td>
</tr>
<tr>
<td>6</td>
<td>WannaCry ransomware</td>
<td>wannacry286.pcap</td>
<td>CTU-Malware-Capture-Botnet-286-1</td>
<td>7282</td>
<td>893.831</td>
<td>d724d8cc6420f06e8a48752f0da11c66</td>
</tr>
<tr>
<td>7</td>
<td>WannaCry ransomware</td>
<td>wannacry287.pcap</td>
<td>CTU-Malware-Capture-Botnet-287-1</td>
<td>6932</td>
<td>863.66</td>
<td>d724d8cc6420f06e8a48752f0da11c66</td>
</tr>
<tr>
<td>8</td>
<td>NotPetya ransomware</td>
<td>notpetya288.pcap</td>
<td>CTU-Malware-Capture-Botnet-288-1</td>
<td>735</td>
<td>1792.329</td>
<td>051084202473f534605c98da8bc20f04</td>
</tr>
<tr>
<td>9</td>
<td>NotPetya ransomware</td>
<td>notpetya289.pcap</td>
<td>CTU-Malware-Capture-Botnet-289-1</td>
<td>750</td>
<td>2573.372</td>
<td>051084202473f534605c98da8bc20f04</td>
</tr>
<tr>
<td>10</td>
<td>NotPetya ransomware</td>
<td>notpetya298.pcap</td>
<td>CTU-Malware-Capture-Botnet-298-1</td>
<td>1461</td>
<td>2266.699</td>
<td>051084202473f534605c98da8bc20f04</td>
</tr>
<tr>
<td>11</td>
<td>NotPetya ransomware</td>
<td>notpetya299.pcap</td>
<td>CTU-Malware-Capture-Botnet-299-1</td>
<td>1209</td>
<td>2233.609</td>
<td>051084202473f534605c98da8bc20f04</td>
</tr>
<tr>
<td>12</td>
<td>benign</td>
<td>normal21.pcap</td>
<td>CTU-Normal-21</td>
<td>495</td>
<td>148.184</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>benign</td>
<td>normal22.pcap</td>
<td>CTU-Normal-22</td>
<td>227.76</td>
<td>1444.912</td>
<td>N/A</td>
</tr>
<tr>
<td>14</td>
<td>benign</td>
<td>normal23.pcap</td>
<td>CTU-Normal-23</td>
<td>207.01</td>
<td>3242.878</td>
<td>N/A</td>
</tr>
<tr>
<td>15</td>
<td>benign</td>
<td>normal24.pcap</td>
<td>CTU-Normal-24</td>
<td>206.84</td>
<td>14241.58</td>
<td>N/A</td>
</tr>
<tr>
<td>16</td>
<td>benign</td>
<td>normal25.pcap</td>
<td>CTU-Normal-25</td>
<td>595.90</td>
<td>5321.061</td>
<td>N/A</td>
</tr>
</tbody>
</table>

WannaCry traffic in Table 5.1 is much bigger than the size of NotPetya traffic, the analysing process has relied mainly on WannaCry traffic. Two types of network attributes, i.e., session-based and time-based, were extracted from analysing the WannaCry and NotPetya traffic, as illustrated in Subsection 5.5.3. As in [36], these extracted attributes are classified as detectable and non-detectable features. A feature of the malicious traffic that can still be measured in the mixed traffic (i.e., both the malicious and the benign streams coexist) of an infected machine is called a detectable feature. While a non-detectable feature is one that appears in the malicious traffic alone, however, the malware obfuscated it within the benign traffic stream so that it becomes difficult to recognise.
5.5 Network traffic analysis and feature extraction

5.5.1 Kill switch property

The dropper attempts to check the connection with one or more hardcoded domain names by sending some DNS queries before executing the WannaCry code. If the DNS query packet is resolved (i.e., the domain name has been registered), the propagation of WannaCry will stop. While if the connection is unsuccessful (i.e., WannaCry could not reach the domain), the dropper will extract and execute the binary code. This mechanism is called kill switch designed by the WannaCry authors to control its propagation. Discovering the kill switch domains could help in stopping payload execution and preventing more infections. It is difficult to read these domain names because they are meaningless and have high randomness in their characters. Fig. 5.1 shows some of the WannaCry DNS query messages. The proposed MaldomDetector for detecting malicious domains in chapter 4 can be used to measure the randomness in the characters of the kill switch domain names like the domain shown in Fig. 5.1 (i.e., iuqerf-sodp9ifjaposdfjhosuirjfaewrgwea). Moreover, MaldomDetector can identify these malicious domains during the DNS query attempt.

WannaCry queries the meaningless domain name several times during the attack, as illustrated in Fig. 5.1. It was also observable from the analysis of WannaCry traffic that few samples have used the kill switch feature during their attacks. However, this attribute may appear in new strains.

5.5.2 The TCP connection

The transport layer protocol carries the packet or datagram from the source to destination application or process. Various transport layer protocols, such as TCP
5.5 Network traffic analysis and feature extraction

5.5.2.1 Communicating with a public IP address without initiating a DNS query

The ransomworm tries to make connections using random public IP addresses without using the DNS service and scans them to propagate to other networks. If the port 445 of the target IP address is open, ransomworm will attempt to exploit it by delivering its payload and executing it inside the victim’s machine. After analysing the ransomware traffic, it was found that most of the public IP addresses used by ransomworm were chosen randomly without querying them previously. In contrast, most benign communications use the DNS service to obtain the destination IP addresses, as indicated in Fig. 5.2. It is noticeable
5.5 Network traffic analysis and feature extraction

that the communication with an external IP address (ext-ip-addr) is a detectable feature because the ratio of benign communications is near zero.

![External IP Addresses Ratio](image)

Figure 5.2: The ratio of communications with external IP addresses (ext-ip-addr) without using DNS service: Malicious & Benign.

In addition, it was observed that most of these ransomworm communications used SMB port 445, i.e., using EternalBlue exploit, while spreading across the networks, as indicated in Fig. 5.3.

![SMB Ports Ratio](image)

Figure 5.3: The ratio of the communications used SMB port: Malicious & Benign.

5.5.2.2 TCP three-way handshake

TCP performs three-way handshake with the remote peer while establishing any connection. It consists of three messages (SYN, SYN-ACK, and ACK) that are
5.5 Network traffic analysis and feature extraction

exchanged between the nodes before the actual data communication begins [17]. It was noticed that WannaCry mostly did not receive SYN-ACK messages after sending TCP SYN packets due to using random IP addresses. Therefore, ransomworm network traffic contained many unsuccessful TCP connections. Fig. 5.4 illustrates that most of the benign connection requests to establish TCP sessions are succeeded, while the ransomworm’s requests are unsuccessful. It is apparent from Fig. 5.4 that TCP SYN-ACK is an informative feature for ransomworm activity detection at the session-level.

![Figure 5.4: The ratio of the SYN-ACK (established TCP connections): Malicious & Benign](image)

However, the number of TCP SYN-ACK packets, i.e., established TCP connections, were counted every 10 seconds to determine whether it is a detectable or non-detectable feature in time-based level. The probability function of these numbers was calculated to understand how the SYN-ACK packets are distributed. The normal distribution curves of the established TCP connections in ransomworm and benign traffic are depicted in Fig. 5.5. It is evident from Fig. 5.5 that there is no clear differentiation in the probability distribution of the TCP SYN-ACK packets between the curves. This distribution may indicate that this
5.5 Network traffic analysis and feature extraction

feature is not easily detectable when the mixed network traffic is monitored as a whole, i.e., the malicious and benign traffic.

![Normal Distribution](image)

Figure 5.5: The normal distribution curves for the number of established TCP connections (SYN-ACK packets) every 10 seconds: Malicious & benign

5.5.2.3 Unestablished TCP connection cases

Since the ransomworm uses random IP addresses in its communications, most of the requested TCP connections were not established, and different messages were received after sending the SYN packet. From the analysis of TCP session traffic, three cases related to WannaCry & NotPetya unestablished connections were determined.

A- In the first case, about 2.7% of the malicious SYN packets reached the destination, but the source received reset (RST, ACK) packet instead of (SYN, ACK) which may mean that the destination port (445) is closed. While only two SYN packets received were reset (RST, ACK) packet in the benign traffic, as indicated in Fig. 5.6 and Fig. 5.7.

It is also useful to analyse the time-based value distribution of the SYN-RST
5.5 Network traffic analysis and feature extraction

Figure 5.6: Sending (RST, ACK) directly after receiving SYN packet.

Figure 5.7: The ratio of the sending SYN packets that received (RST, ACK) packet: Malicious & Benign) every 10 seconds: Malicious & benign.

feature to determine whether it is detectable or not. For that, the frequency of SYN-RST packets was computed over various time-frames. Fig. 5.8 illustrates this by comparing the frequency of the malicious SYN-RST packets alongside the normal packets for every 10 seconds. It is clear from the Fig. 5.8 that this feature (syn-rst) is detectable within the mixed traffic.

B- In the second case, the source host received an ICMP message from the
5.5 Network traffic analysis and feature extraction

Figure 5.8: Frequency of the SYN-RST packets (a) within 1st three minutes (b) within 2nd three minutes (c) within 3rd three minutes.
5.5 Network traffic analysis and feature extraction

intermediate networking device, such as a router, after sending malicious SYN packet. Most of these ICMP messages (about 72.2%) were type 3, which means that the address of the destination host or port is unreachable. The other ICMP type was 11, which means the TTL (Time To Live) value for the sending packet became zero (i.e., time exceeded) and a gateway has discarded the packet. Fig. 5.9 shows the ICMP messages received by ransomworm after sending two SYN packets. While Fig. 5.10 indicates the ratio of these packets in ransomworm and benign traffic. It can be observed that (syn-icmp) is a detectable feature.

![Figure 5.9: ICMP packet types received by ransomworm after attempting to establish TCP connections.](image)

C- In the last case, the source or ransomworm did not receive a response after sending the TCP SYN packet. This may mean that a network device between the source and destination host has dropped the packets or the port is closed. This case constitutes significant part of the ransomworm traffic, as illustrated in Fig. 5.11. It is apparent from Fig. 5.11 that the feature (syn-noresp) is a related feature in the session-based level for ransomworm detection.

Moreover, the time-based value distribution of the syn-noresp feature was also
5.5 Network traffic analysis and feature extraction

![SYN - ICMP Packets Ratio](image1)

Figure 5.10: The ratio of the SYN-ICMP packets: Malicious & Benign.

![SYN - No Response Ratio](image2)

Figure 5.11: The ratio of the SYN packets that received no response (syn-noresp): Malicious & Benign.

inspected to determine whether the feature (syn-noresp) is detectable or not in time-based level. For that, the frequency of the syn-noresp packet is computed over various time-frames. Fig. 5.12 illustrates this by comparing the frequency of the malicious syn-noresp packets beside the normal packets for every 10 seconds. It is apparent from Fig. 5.12 that this feature (syn-noresp) is detectable within the mixed traffic.

5.5.2.4 TCP session without data exchange

After scrutinising the ransomworm TCP sessions, it was observed that some of them have a very short duration, i.e., no traffic exchanged after the TCP con-
5.5 Network traffic analysis and feature extraction

Figure 5.12: Frequency of the SYN-No Response packets (a) within 1st three minutes (b) within 2nd three minutes.

...connection is established. These sessions terminate immediately after finishing the three-way handshake process by sending a FIN packet. After that, the ransomworm reconnects with the same destination IP address and port, and it tries to infect the host through exploiting the SMB protocol, as indicated in Fig. 5.13. In contrast, the normal traffic did not contain sessions that terminate immediately after establishing the TCP connection, as illustrated in Fig. 5.14. Since the ratio of this feature (sh-dur-sess) in benign communications is zero, it is considered a...
5.5 Network traffic analysis and feature extraction

detectable feature.

Figure 5.13: WannaCry traffic that indicates the short-duration connection and reconnection packets.

The Ratio of the Very Short Duration Sessions

<table>
<thead>
<tr>
<th>Malicious</th>
<th>Benign</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.85</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 5.14: The ratio of the short-duration sessions: Malicious & Benign.

5.5.3 Feature extraction

Two types of flow-based features were extracted based on the above network traffic analysis to detect the ransomworm propagation activities. The first type
is called session-based features, which depends on the TCP connection stream generated by ransomworm. The number of these features is 13. While the second type depends on the TCP connection traffic observed over specific time-frames called time-based features. The number of these features is 11. Tables 5.2 and 5.3 show the names of the extracted session-based and time-based features respectively in addition to a brief description for each feature. The extracted features were also classified into detectable and non-detectable features, as indicated in Tables 5.2 and 5.3.

Table 5.2: Session-based extracted features.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Feature Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{s1}$</td>
<td>ext-ip-addr</td>
<td>Detectable</td>
<td>External IP address. Check whether the public IP address of the destination was generated by the source host or determined using the DNS service.</td>
</tr>
<tr>
<td>$F_{s2}$</td>
<td>dst-port</td>
<td>Detectable</td>
<td>Destination port number.</td>
</tr>
<tr>
<td>$F_{s3}$</td>
<td>syn-ack</td>
<td>Detectable</td>
<td>Receiving SYN-ACK packet after sending SYN packet (i.e., successfully established TCP connections).</td>
</tr>
<tr>
<td>$F_{s4}$</td>
<td>syn-synack-dur</td>
<td>Detectable</td>
<td>The duration between SYN and (SYN, ACK) packets.</td>
</tr>
<tr>
<td>$F_{s5}$</td>
<td>fin-pkt</td>
<td>Detectable</td>
<td>The session ends with a FIN packet (session ends normally).</td>
</tr>
<tr>
<td>$F_{s6}$</td>
<td>fin-rst-ack</td>
<td>Detectable</td>
<td>The session contains (RST, ACK) and FIN packets.</td>
</tr>
<tr>
<td>$F_{s7}$</td>
<td>sess-dur</td>
<td>Detectable</td>
<td>The session duration. The period between (SYN, ACK) and FIN packets.</td>
</tr>
<tr>
<td>$F_{s8}$</td>
<td>sh-dur-sess</td>
<td>Detectable</td>
<td>Short duration session. The session ends directly after making the connection.</td>
</tr>
<tr>
<td>$F_{s9}$</td>
<td>syn-rst</td>
<td>Detectable</td>
<td>A connection attempt received an RST packet after sending the SYN packet.</td>
</tr>
<tr>
<td>$F_{s10}$</td>
<td>syn-icmp</td>
<td>Detectable</td>
<td>A connection attempt received ICMP packet after sending the SYN packet.</td>
</tr>
<tr>
<td>$F_{s11}$</td>
<td>syn-icmp-typ</td>
<td>Detectable</td>
<td>ICMP packet type.</td>
</tr>
<tr>
<td>$F_{s12}$</td>
<td>syn-icmp-cod</td>
<td>Detectable</td>
<td>ICMP packet code.</td>
</tr>
<tr>
<td>$F_{s13}$</td>
<td>syn-noresp</td>
<td>Detectable</td>
<td>A connection attempt that has no response after sending the SYN packet.</td>
</tr>
</tbody>
</table>
5.6 Building the network-based classifiers

5.6.1 Building the dataset

Several modules were written in Python to extract and preprocess the values of features in Table 5.2 and 5.3 (i.e., session-based and time-based features) from the ransomworm and benign PCAP files of Table 5.1, and save them in a CSV format. Then these CSV files were cleaned of incorrect data and duplicated rows before using them to train and evaluate the models, as illustrated in Fig. 5.15. The CSV files created from the ransomworm PCAP files were labelled as malicious, while the CSV files that created from the normal PCAP files were labelled as benign. After that, the session-based CSV malicious and benign files were combined to create the session-based dataset, as shown in Fig. 5.15. This dataset was divided randomly into two sub-datasets: training & testing, as illustrated in Table 5.4.

Whereas the time-based CSV malicious and benign files were combined to
create the time-based dataset, as indicated in Fig. 5.15. Since the time-based
consisted fewer samples than the session-based dataset, it was used without di-
viding into training and testing sub-datasets, as demonstrated in Table 5.4.

![Diagram](image)

Figure 5.15: Building labelled datasets out of the raw PCAP files.

<table>
<thead>
<tr>
<th>Dataset type</th>
<th>Type</th>
<th>Sample</th>
<th>Whole dataset size</th>
<th>Training set</th>
<th>Testing set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session-based</td>
<td>malicious</td>
<td>4128</td>
<td>8256</td>
<td>80%</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>benign</td>
<td>4128</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time-based</td>
<td>malicious</td>
<td>445</td>
<td>890</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>benign</td>
<td>445</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 5.6.2 Feature selection

After completing the preprocessing task on the values of the extracted features
and building the dataset, the feature importance, which is one of the techniques
of feature selection [175], was implemented to determine which features are most
relevant or valuable to the problem in this chapter. One of the common ways
for calculating the importance of the features is to use the correlation [176].
The relationship between two variables (features) may be nonlinear; therefore, Spearman’s rank correlation was utilised to evaluate the relationship between the variables. The strength of the relationship between two variables can be measured using the correlation coefficient. The Spearman’s rank correlation coefficient was calculated between each extracted feature in Table 5.2 & 5.3 and the class (response) using the SciPy library in Python. Spearman’s correlation coefficient can be calculated using equation 5.1.

\[ \rho(x, y) = 1 - \frac{6\sum d_i^2}{n(n^2 - 1)} \] (5.1)

Where: \(d_i\) is the difference between each rank of corresponding values of \(x\) and \(y\), and \(n\) is the number of ranks. The rank and score of the importance for each session-based and time-based feature are demonstrated in Fig. 5.16 and Fig. 5.17, respectively.

![Figure 5.16: The chart of the importance between each session-based feature and the label.](image)

The Spearman’s coefficient ranges from \(-1\) to \(1\), which means that the cor-
5.6 Building the network-based classifiers

Figure 5.17: The chart of the importance between each time-based feature and the label.

relation between two variables can be positive or negative. It is noticeable from Fig. 5.16 and Fig. 5.17 that both types of features (i.e., session-based, and time-based) have a positive and negative correlation with the response (label). The features that have importance score more than $|0.5|$ were considered to have a strong correlation, i.e., most relevant features. It is noticeable from Fig. 5.16 that the most relevant session-based features are $F_{s3}$, $F_{s1}$, $F_{s2}$, $F_{s13}$, $F_{s7}$, $F_{s4}$ and $F_{s5}$, while Fig. 5.17 shows that the most relevant time-based features are $F_{t2}$, $F_{t11}$, $F_{t9}$, $F_{t1}$, $F_{t10}$, $F_{t7}$, and $F_{t8}$.

A set of features can be selected from the whole number of extracted features by using the feature importance score, where the features that have an importance score greater than a specific value (threshold) will be selected. Since there is no specific rule to assign the threshold value [177], the threshold value of 0.01 was determined for the problem under consideration basing on network traffic analysis in Section 5.5. Thus, $F_{t3}$ (num-syn-ack) was excluded from the time-based features because its importance score is less than the threshold value, i.e.,
5.6 Building the network-based classifiers

0.01; besides, it is non-detectable.

Although $F_{s2}$ (dst-port) and $F_{t2}$ (num-smp-port) have high importance score, they were also excluded from the session-based and time-based features respectively to make the classifiers depend on features which are likely to appear in the future ransomworm families, where the port number could be different. The session-based and time-based features that were selected to build the classifiers are illustrated in Tables 5.5 and 5.6, respectively.

Table 5.5: The selected session-based features.

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature</th>
<th>Feature name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$F_{s1}$</td>
<td>ext-ip-addr</td>
</tr>
<tr>
<td>2</td>
<td>$F_{s3}$</td>
<td>syn-ack</td>
</tr>
<tr>
<td>3</td>
<td>$F_{s4}$</td>
<td>syn-synack-dur</td>
</tr>
<tr>
<td>4</td>
<td>$F_{s5}$</td>
<td>fin-pkt</td>
</tr>
<tr>
<td>5</td>
<td>$F_{s6}$</td>
<td>fin-rst-ack</td>
</tr>
<tr>
<td>6</td>
<td>$F_{s7}$</td>
<td>sess-dur</td>
</tr>
<tr>
<td>7</td>
<td>$F_{s8}$</td>
<td>sh-dur-sess</td>
</tr>
<tr>
<td>8</td>
<td>$F_{s9}$</td>
<td>syn-rst</td>
</tr>
<tr>
<td>9</td>
<td>$F_{s10}$</td>
<td>syn-icmp</td>
</tr>
<tr>
<td>10</td>
<td>$F_{s11}$</td>
<td>syn-icmp-typ</td>
</tr>
<tr>
<td>11</td>
<td>$F_{s12}$</td>
<td>syn-icmp-cod</td>
</tr>
<tr>
<td>12</td>
<td>$F_{s13}$</td>
<td>syn-noresp</td>
</tr>
</tbody>
</table>

Table 5.6: The selected time-based features.

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature</th>
<th>Feature name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$F_{t1}$</td>
<td>num-ext-ip-addr</td>
</tr>
<tr>
<td>2</td>
<td>$F_{t4}$</td>
<td>ave-syn-synack-dur</td>
</tr>
<tr>
<td>3</td>
<td>$F_{t5}$</td>
<td>num-fin-pkt</td>
</tr>
<tr>
<td>4</td>
<td>$F_{t6}$</td>
<td>num-fin-rst-ack</td>
</tr>
<tr>
<td>5</td>
<td>$F_{t7}$</td>
<td>ave-sess-dur</td>
</tr>
<tr>
<td>6</td>
<td>$F_{t8}$</td>
<td>num-sh-dur-sess</td>
</tr>
<tr>
<td>7</td>
<td>$F_{t9}$</td>
<td>num-syn-rst</td>
</tr>
<tr>
<td>8</td>
<td>$F_{t10}$</td>
<td>num-syn-icmp</td>
</tr>
<tr>
<td>9</td>
<td>$F_{t11}$</td>
<td>num-syn-noresp</td>
</tr>
</tbody>
</table>
5.6 Building the network-based classifiers

5.6.3 Building the classifiers

The Classification Learner App of MATLAB R2018b [178] was used to build the session-based and time-based classifiers of the detection system employing the dataset of Table 5.4. I have chosen the k-fold cross-validation option to train and estimate the models. It protects against overfitting problems and provides an accurate estimation of the predictive accuracy for the trained model with all the data [102]. K-fold cross-validation is a resampling procedure that splits the entire data randomly into k equal-sized folds. The default value of k in the classification learner is 5; however, it was set to 10 because the models have provided the best performance at this value. All the available learning algorithms in the classification learner, such as decision tree and support vector machine (SVM), were explored to train the session-based and time-based models using the features in Tables 5.5 and 5.6 respectively.

After training a set of models using the default setting of the hyperparameters for the learning algorithms, four best models were selected: Decision tree, k-Nearest Neighbours (KNN), SVM, and Discriminant Analysis. All these models provided best performance with the default settings of the hyperparameters.

5.6.4 Model evaluation

After training and validating various models, some criteria are needed to identify the best model(s) from a set of candidate models. There are several binary classification metrics, such as accuracy and False Positive Rate (FPR), which can be applied to evaluate the performance of the trained machine learning models numerically [102]. The formulas of the metrics used to evaluate the session-based
5.6 Building the network-based classifiers

and time-based models are indicated below.

\[
Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.2}
\]

\[
FPR = \frac{FP}{FP + TN} \tag{5.3}
\]

\[
Precision = \frac{TP}{TP + FP} \tag{5.4}
\]

\[
Recall(sensitivity) = \frac{TP}{TP + FN} \tag{5.5}
\]

\[
F1score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5.6}
\]

Where: FP is False Positive, TP is True Positive, TN is True Negative, and FN is False Negative. The evaluation results of the best session-based and time-based classifiers are illustrated in Tables 5.7 and 5.8 respectively.

Table 5.7: Session-based classifier evaluation.

<table>
<thead>
<tr>
<th>Dataset type</th>
<th>Algorithm name</th>
<th>Accuracy (10 cross validation)</th>
<th>Accuracy (testing)</th>
<th>FPR</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>session-based</td>
<td>Decision Tree (fine tree)</td>
<td>99.94 %</td>
<td>99.88</td>
<td>0.0024</td>
<td>0.9976</td>
<td>1</td>
<td>0.9988</td>
</tr>
<tr>
<td></td>
<td>KNN (cubic)</td>
<td>99.55 %</td>
<td>99.45</td>
<td>0.0048</td>
<td>0.9951</td>
<td>0.9939</td>
<td>0.9945</td>
</tr>
<tr>
<td></td>
<td>SVM (Quadratic)</td>
<td>96.23 %</td>
<td>95.94</td>
<td>0</td>
<td>1</td>
<td>0.9188</td>
<td>95.77</td>
</tr>
<tr>
<td></td>
<td>Discriminant Analysis (linear)</td>
<td>95.73 %</td>
<td>95.64</td>
<td>0</td>
<td>1</td>
<td>0.9127</td>
<td>0.9544</td>
</tr>
</tbody>
</table>

ROC curve and AUC can be used to visualise the performance of a model over
5.7 A Multi-Feature and Multi-Classifier Network-based System (MFMCNS) for ransomworm detection

Table 5.8: Time-based classifier evaluation.

<table>
<thead>
<tr>
<th>Dataset Type</th>
<th>Algorithm name</th>
<th>Accuracy (10 cross validation)</th>
<th>FPR</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>time-based</td>
<td>Ensemble</td>
<td>99.66 %</td>
<td>0.0022</td>
<td>0.9977</td>
<td>0.9955</td>
<td>0.9966</td>
</tr>
<tr>
<td></td>
<td>(bagged tree)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SVM</td>
<td>98.99 %</td>
<td>0.0089</td>
<td>0.991</td>
<td>0.9888</td>
<td>0.9899</td>
</tr>
<tr>
<td></td>
<td>(cubic)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>KNN</td>
<td>98.20 %</td>
<td>0.0112</td>
<td>0.9886</td>
<td>0.9753</td>
<td>0.9819</td>
</tr>
<tr>
<td></td>
<td>(cubic)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Discriminant Analysis</td>
<td>97.08 %</td>
<td>0</td>
<td>1</td>
<td>0.9416</td>
<td>0.9699</td>
</tr>
<tr>
<td></td>
<td>(linear)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

the entire operating range. ROC curves of the selected session-based and time-based models are shown in Fig. 5.18 and Fig. 5.19, respectively. It is noticeable that selected models have high-performance in detecting ransomworm activities.

5.7 A Multi-Feature and Multi-Classifier Network-based System (MFMCNS) for ransomworm detection

A Multi Classifier System (MCS) is a subcategory of the hybrid intelligent system, which is a free combination of intelligence techniques to solve a given problem [179]. In MCS, individual classifiers are built using the same machine learning methodology, and then these classifiers are combined using a fusion rule [180]. This section presents the design of proposed MFMCNS, which is specifically designed to track the ransomworm propagation activities. MFMCNS is a hybrid system that combines two techniques, i.e., multi-feature & multi-classifier. It is constructed using two different feature sets and combines two independent classi-
5.7 A Multi-Feature and Multi-Classifier Network-based System (MFMCNS) for ransomworm detection

Figure 5.18: ROC curve and AUC of the session-based selected models: (a) Fine Tree (b) Cubic KNN (c) Quadratic SVM (d) Linear discriminant
5.7 A Multi-Feature and Multi-Classifier Network-based System (MFMCNS) for ransomworm detection

Figure 5.19: ROC curve and AUC of the time-based selected models: (a) Bagged Tree (b) Cubic SVM (c) Cubic KNN (d) Linear discriminant
5.7 A Multi-Feature and Multi-Classifier Network-based System (MFMCNS) for ransomworm detection

Classifiers into a single system according to a fusion rule. Fig. 5.20 depicts the concept of MFMCNS.

Although the session-based and time-based classifiers provide high accuracy, as concluded from the evaluation results in Subsection 5.6.4, MFMCNS has been proposed for two objectives. The first is to improve the generalisation by using two different views (multi-feature). The second is to exploit the different local behaviour of the base classifiers (C1 & C2) to make the system more reliable. Three key components must be achieved to build any MCS:

- Specify the topology that is used to interconnect the individual classifiers.
- Generate a group of diverse (independent) classifiers.
- Specify a method for combining (fusion) the output of the classifiers.

The subsections below summarise how these main components have been achieved.
5.7 A Multi-Feature and Multi-Classifier Network-based System (MFMCNS) for ransomworm detection

5.7.1 System topology

The individual classifiers can be interconnected using three schemes [181]:

- Hierarchical
- Cascading (serial)
- Parallel

The parallel topology was selected to structure MFMCNS by connecting the session-based classifier (C1) and time-based classifier (C2) in parallel. The decisions (D1 & D2) of the classifiers have been combined to produce the final decision, as illustrated in Fig. 5.21.

![Figure 5.21: The architecture of MFMCNS.](image)

The captured packets are forwarded to two independent levels simultaneously, where the first level is called the session-based classification unit, and the second
one is called the time-based classification unit. The incoming packets to the preparation unit of the first level are preprocessed to extract the session-based features depicted earlier in Table 5.5 and create the session-level feature vector (V1). In contrast, the incoming packets to the second level are collected and stored temporally in a buffer unit. Then the buffer’s data (a group of data packets) is forwarded to the preparation unit every ten seconds to extract the time-based features shown in Table 5.6 and create the time-level feature vector (v2). After that, V1 and V2 are sent to classifier C1 and C2 respectively to classify them and generate appropriate decisions, i.e., D1 & D2. These decisions are then forwarded to a fusion unit to make the final decision and trigger an alarm when detecting a potential ransomworm activity.

### 5.7.2 Classifier dependencies

The proposed MFMCNS is useful especially if the individual classifiers are diverse, i.e., make uncorrelated errors for each other. According to [179], the diversity of multiple classifiers (MCS) can be ensured with either of the following approaches:

- **Diversify input data to classifiers.** This approach assumes that classifiers become complementary when they train on different (disjoint) input data. This diversification can be achieved by using either different data partitions or different feature sets.

- **Diversify the models.** In this method, diverse classifiers can be generated by using models with different biases built on one dataset. The different versions for the same model can also be employed to create diverse classifiers through varying the parameters of the algorithm.
5.7 A Multi-Feature and Multi-Classifier Network-based System (MFMCNS) for ransomworm detection

- Diversify output of the classifiers. The diversity can be achieved in the classifiers by manipulating the output of the individual classifiers so that each classifier classifies only a part of classes. Then all the class labels are restored through a fusion unit.

In MFMCNS, the first and second methods above were applied to get diverse classifiers (C1 & C2). Each classifier was built using a different feature set and training set, i.e., session-based dataset and time-based dataset, as illustrated in Section 5.6. Furthermore, to increase the diversity of the classifiers, the individual classifiers (C1 & C2) were chosen to be heterogeneous classifiers, i.e., the Decision Tree (fine tree) model was selected for C1 whereas Ensemble (bagged tree) model was selected for C2.

5.7.3 Fusion unit

There are different fusion methods to generate the final decision of MCS [181] [182]. In general, the fusion (combination) method can be divided into two types:

- Trainable: It requires training data, such as fuzzy integral and boosting.

- Non-trainable: It does not require training data such as majority voting, sum, and median.

The majority voting method was selected to combine the outputs of the classifiers C1 & C2 because MFMCNS has binary classifiers with approximately similar performance in terms of accuracy, and there is no sufficient training data to train the combiner. Majority voting is a simple way to combine the decisions made
5.7 A Multi-Feature and Multi-Classifier Network-based System (MFMCNS) for ransomworm detection

already by the individual classifiers at the decision level. The output decisions from both classifiers are combined as follows:

Consider M classifiers, K classes, and a given pattern Z that generates the feature vector B.

\[ C = \{c_1, c_2, ..., c_m\} \] denotes a set of classifiers.

\[ X = \{x_1, x_2, ..., x_m\} \] is a set of training samples.

\[ W = \{w_1, w_2, ..., w_k\} \] is a set of classes.

Since MFMCNS has only two classifiers, \(M=K=2\). In majority vote rule, the combination is at the decision level. I took the predicted output of the classifier as follow:

\[
C_m(x) = \begin{cases} 
1, & \text{if } x \text{ is recognised as malicious by } c_m \\
0, & \text{if } x \text{ is recognised as Benign by } c_m 
\end{cases}
\]

The classifiers are considered to give a posteriori probability \(P\) as:

\[
P = \frac{1}{M} \sum_{1}^{M} c_m(x) \in \{0, \frac{1}{m}, 1\} \tag{5.7}
\]

The values of \(P \in \{0, 0.5, 1\}\). Each classifier (C1 or C2) has two classes, i.e., malicious or benign, and whichever class gets the highest vote (>50%), will be selected as a final decision. Since MFMCNS has only two classifiers, I have considered the situation when the malicious class has 50% of the votes as a suspected case. The majority vote rule for two classifiers assigns a given pattern \((z) \rightarrow \text{class}(w_k)\)
5.7 A Multi-Feature and Multi-Classifier Network-based System (MFMCNS) for ransomworm detection

\[ w_k = \begin{cases} 
\text{malicious, if } P = 1 \\
\text{suspected, if } P = 0.5 \\
\text{benign, if } P = 0 
\end{cases} \]

Table 5.9 indicates the alarms that can be triggered by the fusion unit of MFMCNS.

<table>
<thead>
<tr>
<th>Output (C1)</th>
<th>Output (C2)</th>
<th>Vote</th>
<th>Alarm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malicious</td>
<td>Malicious</td>
<td>100%</td>
<td>Risk</td>
<td>An intrusion has been detected, and the infected network has to be isolated.</td>
</tr>
<tr>
<td>Malicious</td>
<td>Benign</td>
<td>50%</td>
<td>Suspicion</td>
<td>A suspected case of intrusion has been detected, and the infected network should be isolated.</td>
</tr>
<tr>
<td>Benign</td>
<td>Malicious</td>
<td>50%</td>
<td>Suspicion</td>
<td>A suspected case of intrusion has been detected, and the infected network should be isolated.</td>
</tr>
<tr>
<td>Benign</td>
<td>Benign</td>
<td>0%</td>
<td>Normal</td>
<td>No action required.</td>
</tr>
</tbody>
</table>

5.7.4 Further evaluation

Although the individual classifiers have been evaluated carefully, as shown in Subsection 5.6.4, this subsection presents further evaluation to prove the reliability of MFMCNS and illustrates how it triggers the final decision.

The shortage of ransomworm datasets is a challenge in this research. After performing extensive research to find a reliable dataset, I was able to collect new PCAP files shown in Table 5.10 to perform an extra evaluation of the proposed system, i.e., MFMCNS. Two WannaCry PCAP files were collected from PacketTotal [183]. PacketTotal is a public repository which allows publishing and download of PCAP files. This repository has been used in several previous works, such as [184] [185] [186].
New datasets have been generated using the files in Table 5.10, as indicated in Table 5.11. These new datasets were not used in the process of training and testing the individual classifiers C1 & C2. Since the number of samples in the session-based dataset is more than the samples in the time-based dataset, the evaluation of MFMCNS was divided into two phases. In phase one, the number of samples for the session-based and time-based datasets have been equalised to compute the votes and illustrate how MFMCNS makes the final decision. Whereas in the second phase, only a session-based dataset was used to evaluate the session-based classifier (C1) of MFMCNS.

Table 5.10: Brief information of other PCAP files used for extra evaluation.

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>File name</th>
<th>Source</th>
<th>Size (KB)</th>
<th>Hash value (MD5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Malicious</td>
<td>wannacry_256.pcap</td>
<td>[126]</td>
<td>13,777</td>
<td>d724d8cc6420f06e8a487 52f0da11c66</td>
</tr>
<tr>
<td>2</td>
<td>Malicious</td>
<td>wannacry_283.pcap</td>
<td>[126]</td>
<td>1599</td>
<td>e16b903789e41697ecab 21ba6e14fa2b</td>
</tr>
<tr>
<td>3</td>
<td>Malicious</td>
<td>wannacry_17.pcap</td>
<td>[183]</td>
<td>16,887</td>
<td>3e73d99dc8854e517d9876 2787a29c79</td>
</tr>
<tr>
<td>4</td>
<td>Malicious</td>
<td>wannacry_40.pcap</td>
<td>[183]</td>
<td>39,063</td>
<td>c1c4d07568cb9306d0697 8892972529</td>
</tr>
<tr>
<td>5</td>
<td>Benign</td>
<td>normal_27.pcap</td>
<td>[126]</td>
<td>434,440</td>
<td>N/AV</td>
</tr>
<tr>
<td>6</td>
<td>Benign</td>
<td>normal_29.pcap</td>
<td>[126]</td>
<td>446,690</td>
<td>N/AV</td>
</tr>
</tbody>
</table>
### 5.7 A Multi-Feature and Multi-Classifier Network-based System (MFMCNS) for ransomworm detection

Table 5.11: New datasets for extra evaluation.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Dataset type</th>
<th>Type</th>
<th>No. of samples</th>
<th>dataset size</th>
<th>Total dataset size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>session-based</td>
<td>malicious</td>
<td>245</td>
<td>490</td>
<td>980</td>
</tr>
<tr>
<td></td>
<td></td>
<td>benign</td>
<td>245</td>
<td>490</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>time-based</td>
<td>malicious</td>
<td>245</td>
<td>490</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>benign</td>
<td>245</td>
<td>490</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Session-based</td>
<td>malicious</td>
<td>4750</td>
<td>9500</td>
<td>9500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>benign</td>
<td>4750</td>
<td>9500</td>
<td></td>
</tr>
</tbody>
</table>

A MATLAB script was used to evaluate MFMCNS employing the datasets of Table 5.11. In the first phase, the detection accuracy for each classifier (C1 & C2) is determined separately, and their decisions for each sample are forwarded to the fusion unit to make the final decision. The outputs of the fusion unit are clustered based on the vote ratio, i.e., 100%, 50%, and 0%. While in the second phase, the detection accuracy of only session-based classifier (C1) is computed. The evaluation results in Table 5.12 demonstrate that MFMCNS is reliable with high accuracy and low FPR.

Table 5.12: Further evaluation results of MFMCNS.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Level type</th>
<th>Classifier</th>
<th>Accuracy</th>
<th>FPR</th>
<th>Majority vote</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(100%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(50%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(0%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(malicious)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(suspected)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(benign)</td>
</tr>
<tr>
<td>1</td>
<td>Session-based</td>
<td>Decision Tree</td>
<td>100 %</td>
<td>0</td>
<td>490 1 489</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(fine tree)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time-based</td>
<td>Ensemble</td>
<td>99.8 %</td>
<td>0.0041</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(bagged tree)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Session-based</td>
<td>Decision Tree</td>
<td>99.78 %</td>
<td>0.0044</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(fine tree)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.8 Discussion

Since the datasets used in previous research works are not identical, and their data source, i.e., host-based or network-based, is different; hence it is difficult to compare MFMCNS with the systems presented in the related work section. This section has only summarised some characteristics of 11 detection methods discussed in Section 5.2 in addition to the proposed system (MFMCNS), as illustrated in Table 5.13.

Table 5.13: Some properties of the ransomworm detection methods.

<table>
<thead>
<tr>
<th>Method name</th>
<th>Source</th>
<th>Method type</th>
<th>Accuracy</th>
<th>FPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Moving Target Defense (MTD)</td>
<td>[154]</td>
<td>Host-based</td>
<td>98.6%</td>
<td>N/AV</td>
</tr>
<tr>
<td>2 Software-Defined Networking (SDN)</td>
<td>[156]</td>
<td>Network-based</td>
<td>N/AV</td>
<td>N/AV</td>
</tr>
<tr>
<td>3 N-gram of opcodes</td>
<td>[157]</td>
<td>Host-based</td>
<td>Depends on N gram and no. of features. 91.43% multi-classification, 99.3% binary classification</td>
<td>N/AV</td>
</tr>
<tr>
<td>4 Reverse engineering technique</td>
<td>[158]</td>
<td>Network-based</td>
<td>N/AV</td>
<td>N/AV</td>
</tr>
<tr>
<td>5 Machine learning</td>
<td>[123]</td>
<td>Host-based</td>
<td>78%</td>
<td>N/AV</td>
</tr>
<tr>
<td>6 Artificial neural networks and fast Fourier transfer</td>
<td>[159]</td>
<td>Host-based</td>
<td>N/AV</td>
<td>N/AV</td>
</tr>
<tr>
<td>7 Static analysis</td>
<td>[160]</td>
<td>Host-based</td>
<td>N/AV</td>
<td>N/AV</td>
</tr>
<tr>
<td>8 Dynamic analysis</td>
<td>[161]</td>
<td>Host-based</td>
<td>N/AV</td>
<td>N/AV</td>
</tr>
<tr>
<td>9 Static and dynamic analysis</td>
<td>[162]</td>
<td>Host-based &amp; network-based</td>
<td>N/AV</td>
<td>N/AV</td>
</tr>
<tr>
<td>10 Static and dynamic analysis</td>
<td>[163]</td>
<td>Host-based</td>
<td>N/AV</td>
<td>N/AV</td>
</tr>
<tr>
<td>11 Term Frequency-Inverse</td>
<td>[164]</td>
<td>Host-based</td>
<td>N/AV</td>
<td>N/AV</td>
</tr>
<tr>
<td>12 MFMCNS</td>
<td>This research work</td>
<td>Network-based</td>
<td>99.73</td>
<td>0.0053</td>
</tr>
</tbody>
</table>

The approaches in [154] [157] [123] [159] are host-based which require the host to be infected first in order to identify the malicious actions of ransomware. However, the infection with ransomware can expose the system’s assets to the risk of encryption or exfiltration. This research work presents a multi-feature and multi-classifier network-based system, i.e., MFMCNS, for detecting the propagation
activities of ransomware before executing the payload.

Previous works in [160] [161] [163] presented analysis of WannaCry ransomware inside the victim’s system without presenting a detection approach. The approach [158] used reverse engineering techniques to analyse the WannaCry samples. However, it examined certain network artifacts of WannaCry propagation but lacked a mechanism for detecting the attack. MFMCNS presents a careful analysis of ransomworm traffic and provides a multi-feature and multi-classifier network-based detection system.

The method [156] is based only on two features, i.e., kill switch domain names and IP addresses, using dynamic blacklisting database. However, this method is not effective to detect new malicious IP addresses or domain names because they have to be identified before can be added to the blacklists. Moreover, WannaCry uses random IP addresses to spread to other networks, and the kill switch feature is no longer employed in the several strains of WannaCry. In MFMCNS, a total of 21 related features were extracted and selected from two different flow levels: session-based and time-based.

The method [162] investigates the characteristics of WannaCry attack using static and dynamic analysis. Some signatures were extracted from ransomware executable files, and six signature rules were written using Snort tool to detect WannaCry activities without mentioning the results. MFMCNS has evaluated using two different datasets where the average detection accuracy is 99.8% and the false positive rate is 0.0041.
5.9 Summary

In this chapter, some gaps in network-based ransomworm detection have been addressed by thoroughly analysing the network activities of ransomworm taking WannaCry and NotPetya as a case study. The analysis indicates that ransomworm has several network activities that can be detected. Based on the network traffic analysis, two sets of related features were extracted and selected from two different flow levels: session-based and time-based. Then, two independent machine learning classifiers were built using the two feature sets.

Furthermore, a multi-feature and multi-classifier network-based system, MFMCNS, has been proposed for detecting the spread activities of ransomworm. MFMCNS employs the classifiers working in parallel on different levels. The detection accuracy for the session-based classifier and time-based classifier is 99.88% and 99.66%, respectively, and each classifier can accurately detect the propagation activities of ransomworm. Two different datasets were used to test MFMCNS, and the evaluation results demonstrate that MFMCNS is a reliable and effective approach for detecting unseen malicious packets and validate the effectiveness of the extracted features. The proposed MFMCNS significantly contributes to the malware research field and provides significant support for the development of network-based malware detection systems.
Chapter 6

Conclusions and Future Work

6.1 Conclusions

Cyberattacks are existential threats targeting computer information systems, businesses, and individuals. Cybercriminals use a variety of methods to launch their cyberattacks, such as malware and denial of service. Malware is the key choice of weapon to execute various malicious actions. Ransomware is a type of advanced malware, and it is increasingly used by perpetrators to generate revenue through digitally imposed extortion. This threat is not restricted to any geography or operating system and can take action on any number of devices.

Ransomware gets much attention by the security communities because it causes extensive financial damages to a wide range of victims, such as organisations, individuals, businesses, and in particular the health sector. The families of ransomware are rapidly evolving, and they continue to be one of the most prevalent cyberattacks in recent years. The most severe types of attacks are based on crypto ransomware that uses robust encryption technique, i.e., hybrid
encryption asymmetric and symmetric types, to deny users from accessing their files.

The analysis of network traffic reveals that ransomware requires covert communications with its C&C server before executing their harmful payloads. These communications are a vital characteristic for detecting ransomware compromised hosts. Also, a detailed analysis of the traffic of WannaCry and NotPetya families has revealed significant communication-related interactions that can be effectively utilised to detect compromised hosts before infecting other hosts. Experimental results validated that the network-based detection approach using machine learning techniques are effective and accurate to detect ransomware activities during its incubation stage.

6.2 Main Contributions

In this section, a summary of the main research contributions is outlined, demonstrating how this research has contributed to the current body of knowledge.

- **Behavioural analysis of crypto ransomware network communications**

  Comprehensive analysis of crypto ransomware network activities was carried out to study and understand communication patterns and behaviours, taking Locky as a case study. This analysis has shown that crypto ransomware network traffic contains activities of several network protocols, including TCP, HTTP, and DNS. The traffic of these protocols was scrutinised. The analysis has also shown that crypto ransomware employs DGA to generate many pseudorandom domain names to contact its C&C server.
These investigations and observations helped define new feature extraction techniques and enable improved detection of ransomware.

- **Extraction of features from different network levels**
  
  Based on the thorough analysis of crypto ransomware network traffic in chapter 3, a total of 18 new features were extracted from TCP, HTTP, DNS, and NBNS traffic. These features are of significant importance to differentiate traffic generated by a compromised host. These features were classified into four different types: behavioural, non-behavioural, detectable, and non-detectable. This classification approach is novel and improves the overall detection accuracy of obfuscated malware communication.

- **Multi-feature and multi-classifier network-based crypto ransomware detection**
  
  In chapter 3, two independent binary classifiers were built using ML technique. These individual classifiers work on two different levels: packet-level & flow-level. The experimental results demonstrate a high detection accuracy for each level: 97.92% and 97.08% respectively and validate the effectiveness of the extracted features. Moreover, a multi-feature and multi-classifier network-based crypto ransomware detection system was proposed. This system consists of two main modules: the feature extraction module and the decision-making module. The function of the feature extraction module is to receive the incoming network packets, prepare the data, and extract two feature vectors (i.e., packet-level & flow-level). In contrast, the decision module contains the two individual classifiers connected in parallel and working on two different feature vectors to detect the packet-level
and the flow-level activities of crypto ransomware. The outputs (decisions) of these classifiers are combined using simple OR-based decision unit to provide a single classification.

- **Novel algorithm for identifying malicious domain names**

A new algorithm called Randomness Measuring Algorithm (RMA) was designed and validated in chapter 4 to identify malicious domain names by measuring the randomness of the domain name’s characters. RMA is a language-independent algorithm that accepts a set of simple character-based features and entropy value as inputs. RMA processes the inputs according to threshold values that were determined by carrying out a detailed analysis for a large number of domain names generated by various DGA families. RMA has been evaluated using many domains taken from 20 DGA families, where the average accuracy is 94.05% and 83.14% for malicious and benign domains, respectively. The evaluation results show that RMA is a significant contribution to advancing the detection of malicious DNS queries.

- **Detection of algorithmically generated malicious domain names**

An algorithmically generated malicious domain name detection system, called MaldomDetector, has been proposed and prototyped in chapter 4. MaldomDetector is capable of detecting DGA-based communications and successfully circumvent the progression of life ransomware to connect a C&C server. MaldomDetector is comprised of two components, data preparation and decision making. The data preparation is used to pre-process the in-
coming DNS packets and extract feature vectors out of the characteristics of the domain name strings. It utilises the above-described RMA. The decision making is comprised of a ML-based domain name classifier, processing basic and derived features as inputs to determine either a malicious or benign domain name. Key advantages of the proposed MaldomDetector approach are: (1) the need for little data and computing resources for checking queried domain names and (2) high detection accuracy of approximately 98% in detecting DGA-based domains. Hence, MaldomDetector is a novel and effective approach to raise an early warning about potential malicious DNS communications.

- Behavioural analysis of ransomworm propagation

In chapter 5, the propagation activities of ransomworm were thoroughly analysed using WannaCry and NotPetya ransomworm families, as case study examples. This detailed understanding of ransomworm behavioural propagation is a key contraption to the current body of knowledge on ransomworm.

- Extraction of features from different network flow levels and build labelled datasets

Detailed WannaCry and NotPetya traffic analytics revealed two types of unique features from two flow levels for detecting ransomworm spread activities. The first type is called the session-based features, which depends on the TCP connection streams generated by ransomworm. The second type depends on the TCP connection traffic collected at a specific time and called the time-based features. Two sets of 12 and 9 features were
selected from session-based data and flow-based data, respectively. Two labelled and independent datasets, session-based and flow-based, were built to train two individual machine learning classifiers. These labelled datasets will be made as freely available as a contribution to the malware research community.

- **Multi-Feature and Multi-Classifier Network-based System (MFM-CNS) for tracking ransomworm propagation**

Chapter 5 introduces a novel MFM-CNS, designed to track ransomworm propagation activities. MFM-CNS is a hybrid approach that combines two techniques, i.e., multi-feature & multi-classifier, and is comprised of a parallel topology consisting of two layers, i.e., session-based and time-based. The proposed approach for extracting multi-feature sets critically contributes to the field of research in network-based malware detection and provides a new method for the development of ransomworm and generally network-based malware detection systems.

### 6.3 Future work

While each of the technical chapters of this thesis has presented new approaches for detecting ransomware network activities, these approaches can be further extended and customised for deployment in antimalware tools and network security appliances. This section presents some proposed future works that would allow the continuation of research beyond the achieved objectives presented in this thesis.
6.3 Future work

One future work would be the deployment of the ML models, which have been built in this thesis, to make them available in a production environment so that they can be used in security products to provide predictions on malware activities. Security appliances build upon the proposed model-based ransomware detection can use an inference engine that is programmed by an externally trained classification model. It is possible to deploy such models using a library, such as R and scikit-learn, or a tool like Weka. Commercial solutions, such as Nvidia RAPIDS libraries offer accelerated ML solutions, enabling scalable technology to meet bandwidth demands beyond 100 Gb/s. Also, it is possible to re-implement the predictive aspect of the model utilising custom-purpose platforms that are comprised of general-purpose processing with accelerator technologies such as GPUs, tensor processors or FPGA technologies. Future work would primarily focus on challenges integrating the proposed model-based approach into anti-malware tools and network security appliances tailored for targeted platforms. Further research may focus on:

- Programming paradigms of security appliances for regular updating pre-trained models, that may incorporate detection capability of new malware variances.

- Achieving high inspection throughput using accelerators without compromising detection accuracy.

- Customising proposed models to interact and comply with standard APIs (Application Programming Interface), for monitoring, anomaly logging, threat response, and data visualisation.
Another future work would be the extension of network traffic analysis beyond ransomware families so that new features can be identified, enabling the development of a more generalised network-based malware detection solution. This may include the use of statistics (packet size distribution, flow bandwidth, beckoning, etc.), patterns and Layer-7 content signatures as features, combined with features presented in this thesis. The focus will be on investigating all types of suspicious behaviour, including legitimate access patterns used for reconnaissance purpose to identify intrusion or malware deliver strategy. Such an approach has the potential to detect malware potentially during the delivery and communication phases.

As outlined above, the use of L7 (Layer-7) signatures as features, captured as string patterns using regular expression (regex), is expected to provide an efficient discriminator for malware detection. Regex type pattern matching can identify many static and dynamic malware-specific patterns, enhancing significantly features extraction beyond traffic statistics. Titan IC, a QUB start-up company, developed a high-performance hardware-based regex processor that is widely used on Smart NICs and FPGA-based regex offload accelerators. The Nvidia BlueField 2 DPU combines advanced NIC technology, 8x ARM A72 processor cores and the Titan IC regex processor for L3-L7 traffic inspection at 50 Gb/s. The Nvidia DPU is an ideal platform not only for extracting essential traffic statistics but also to undertake rule-based L5-L7 content inspection using IDS/IPS/NGFW rules for enhancing features for the proposed ML-based malware detection approach. Within the context of network-based malware detection using ML, features extraction can be basically the matching of a complex string pattern, similar
to natural language processing (NLP), extracting tailored features, beyond current solutions. However, further research is required to derive such a new methodology for capturing unique features using hardware-accelerated regex pattern matching.

Finally, the protection of cloud-based services from external malware-related threads is another promising use-case for the proposed approaches in this thesis. The most common and widely used cloud computing service model is Software as a Service (SaaS), executed on VMs or containers. Targeted attacks on such platformers are achieved in form of false database queries or malicious script injections with the objective to manipulate and hijack the victim’s VM. Threat detection can be therefore performed at the hypervisor, through the examination of features gathered at network levels of a cloud node. Further research may focus on the integration challenges of the proposed ML-based detection methodology within cloud platforms. Research questions surrounding (a) scalable monitoring without violating VM isolation and confidentiality, (b) architectural approach for collecting telemetry at the network and from the hypervisor, (c) distributed computing model for ML-based security approach outside the VM using limited infrastructure resources, (d) effective orchestration, policy enforcement, threat response, will define future work.
Appendix A

Author’s Publications


Bibliography


*International Journal of Advanced Networking and Applications*, vol. 10, 

2009.

[53] R. Am and R. Chezian, “Intrusion detection system techniques and tools: 
A survey,” *Scholar Journal of Engineering and Technology (SJET)*, vol. 5, 


and detecting dga-based botnets using dns traffic analysis,” *Computers & 

[56] D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen, “A lstm based 
framework for handling multiclass imbalance in dga botnet detection,” *Neu-

of botnet detection based on dns,” *Neural Computing and Applications*, 

studying domain generation algorithm (dga)-based malware,” in *14th Inter-


[137] H. Mac, D. Tran, V. Tong, L. G. Nguyen, and H. A. Tran, “Dga botnet
detection using supervised learning methods,” in Proceedings of the Eighth
International Symposium on Information and Communication Technology,

[138] Y. Shi, G. Chen, and J. Li, “Malicious domain name detection based on

[139] W.-J. Song and B. Li, “A method to detect machine generated domain
names based on random forest algorithm,” in 2016 International Conference
509–513.

traffic features in managed network,” Security and Communication Net-


https://osint.bambenekconsulting.com.feeds/

[143] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, “Character level
based detection of dga domain names,” in 2018 International Joint Con-
ference on Neural Networks (IJCNN). IEEE, 2018, pp. 1–8.


