



Nova Southeastern University
NSUWorks

CEC Theses and Dissertations

College of Engineering and Computing

2017

Improved Detection for Advanced Polymorphic Malware

James B. Fraley

Nova Southeastern University, jf1280@mynsu.nova.edu

This document is a product of extensive research conducted at the Nova Southeastern University [College of Engineering and Computing](#). For more information on research and degree programs at the NSU College of Engineering and Computing, please click [here](#).

Follow this and additional works at: https://nsuworks.nova.edu/gscis_etd

 Part of the [Computer Sciences Commons](#)

Share Feedback About This Item

NSUWorks Citation

James B. Fraley. 2017. *Improved Detection for Advanced Polymorphic Malware*. Doctoral dissertation. Nova Southeastern University. Retrieved from NSUWorks, College of Engineering and Computing. (1008)
https://nsuworks.nova.edu/gscis_etd/1008.

This Dissertation is brought to you by the College of Engineering and Computing at NSUWorks. It has been accepted for inclusion in CEC Theses and Dissertations by an authorized administrator of NSUWorks. For more information, please contact nsuworks@nova.edu.

Improved Detection for Advanced Polymorphic Malware

by

James B. Fraley

A Dissertation Proposal submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

in

Information Assurance

College of Engineering and Computing

Nova Southeastern University

2017

We hereby certify that this dissertation, submitted by James Fraley, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

James D. Cannady, Ph.D.
Chairperson of Dissertation Committee

Date

Kelly L. Hughes, Ph.D.
Dissertation Committee Member

Date

Peixiang Liu, Ph.D.
Dissertation Committee Member

Date

Approved:

Yong X. Tao, Ph.D., P.E., FASME
Dean, College of Engineering and Computing

Date

College of Engineering and Computing
Nova Southeastern University

2017

An Abstract of a Dissertation Submitted to Nova Southeastern University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Improved Detection for Advanced Polymorphic Malware

by
James B. Fraley
May 2017

Malicious Software (malware) attacks across the internet are increasing at an alarming rate. Cyber-attacks have become increasingly more sophisticated and targeted. These targeted attacks are aimed at compromising networks, stealing personal financial information and removing sensitive data or disrupting operations. Current malware detection approaches work well for previously known signatures. However, malware developers utilize techniques to mutate and change software properties (signatures) to avoid and evade detection. Polymorphic malware is practically undetectable with signature-based defensive technologies. Today's effective detection rate for polymorphic malware detection ranges from 68.75% to 81.25%. New techniques are needed to improve malware detection rates. Improved detection of polymorphic malware can only be accomplished by extracting features beyond the signature realm. Targeted detection for polymorphic malware must rely upon extracting key features and characteristics for advanced analysis. Traditionally, malware researchers have relied on limited dimensional features such as behavior (dynamic) or source/execution code analysis (static). This study's focus was to extract and evaluate a limited set of multidimensional topological data in order to improve detection for polymorphic malware. This study used multidimensional analysis (file properties, static and dynamic analysis) with machine learning algorithms to improve malware detection. This research demonstrated improved polymorphic malware detection can be achieved with machine learning. This study conducted a number of experiments using a standard experimental testing protocol. This study utilized three advanced algorithms (Metabagging (MB), Instance Based k-Means (IBk) and Deep Learning Multi-Layer Perceptron) with a limited set of multidimensional data. Experimental results delivered detection results above 99.43%. In addition, the experiments delivered near zero false positives. The study's approach was based on single case experimental design, a well-accepted protocol for progressive testing. The study constructed a prototype to automate feature extraction, assemble files for analysis, and analyze results through multiple clustering algorithms. The study performed an evaluation of large malware sample datasets to understand effectiveness across a wide range of malware. The study developed an integrated framework which automated feature extraction for multidimensional analysis. The feature extraction framework consisted of four modules: 1) a pre-process module that extracts and generates topological features based on static analysis of machine code and file characteristics, 2) a behavioral analysis module that extracts behavioral characteristics based on file execution (dynamic analysis), 3) an input file construction and submission module, and 4) a machine learning module that employs various advanced algorithms. As with most studies, careful attention was paid to false positive and false negative rates which reduce their overall detection accuracy and effectiveness. This study provided a novel approach to expand the malware body of knowledge and improve the detection for polymorphic malware targeting Microsoft operating systems.

Acknowledgments

I would like to thank Professor James Cannady for providing guidance and leadership as the Chair of my dissertation committee. His brutal honesty has made me a better student and person. I will always look back fondly on his instruction, guidance and wisdom. I also want to thank Dr. Hughes and Dr. Liu for working diligently on my committee. Your comments and feedback served to build a better end-product and researcher. I also want to thank Dr. Ackerman and Dr. Seagull who provided guidance and inspiration early on in the process.

I would also like to thank David Wasson who provided technical guidance and helped me build an infrastructure to process literally hundreds of experimental models. He provided a sound technical advice and allowed me to test ideas that neither of us had ever tackled. Some for good reason. I am grateful that he took my calls and provided technical insights when things broke. He helped me breakthrough by answering questions, giving up his time and providing technical expertise when it was desperately needed.

I am very appreciative of my employer both McAfee and Intel. My journey was only made possible by the company's support and encouragement. McAfee has been exceptionally supportive, and I have appreciated the support from my senior management. I also have been surrounded by exceptional colleagues at McAfee who have increased my knowledge in information security by leaps and bounds. It has been my distinct honor to work and learn from all of you over the years.

I am so grateful to Theresa who partnered with me on my life journey. I thank her for inspiring, encouraging and fully supporting my academic journey. I appreciated her patience and willingness to let me work some insane hours for a long time. I am grateful that my mother instilled a belief that anything is possible if you work hard enough. Jake and Dylan, I hope my hard work and dedication provides a good example for you as you begin your academic journey. Lastly, I appreciate my family's support and encouragement (especially David Bressi) throughout the entire process.

List of Tables

Tables

1. Advanced Detection Studies. Sources Used in Study	92
2. Confusion Matrix	125
3. Example Confusion Matrix	128
4. Example Unweighted Testing Results	128
5. Experimental Resource Requirements	134
6. MB Baseline Classification Results	144
7. MB Baseline Experimental Results	144
8. MB Reduced Feature Selection Classification Results	145
9. MB Reduced Feature Selection Experimental Results	145
10. MB Data Amplification Classification Results	145
11. MB Data Amplification Experimental Results	146
12. MB Experimental Results	147
13. IBK Baseline Classification Results	152
14. IBK Baseline Experimental Results	152
15. IBK Reduced Feature Selection Classification Results	152
16. IBK Reduced Feature Selection Experimental Results	153
17. IBK Data Amplification Classification Results	153
18. IBK Data Amplification Experimental Results	153
19. IBK Experimental Results	155
20. DLMLP Baseline Classification Results	160
21. DLMLP Baseline Experimental Results	160
22. DLMLP Reduced Feature Selection Classification Results	161
23. DLMLP Reduced Feature Selection Experimental Results	161
24. DLMLP Data Amplification Classification Results	161
25. DLMLP Data Amplification Experimental Results	162
26. DLMLP Experimental Results	163

List of Figures

Figures

1. System Overview 101
2. ROC Results 124
3. Comparative Baseline Testing Results 129
4. ROC Analysis 130
5. MB Graph Results 146
6. IBk Graph Results 154
7. DLMLP Graph Results 162

Table of Contents

Abstract ii

Acknowledgements iii

Approval/Signature Page iv

List of Tables v

List of Figures vi

Chapters

1 Introduction 1

Background 1

Problem Statement 10

Dissertation Goal 11

Research Questions and/or Hypotheses 14

Relevance and Significance 16

Barriers and Issues 18

Assumptions, Limitations and Delimitations: 20

Summary 23

Definition of Terms 25

List of Acronyms 31

2 Review of the Literature 33

Overview of Topics 33

Synthesis of Current Literature 39

Research Methods 77

Gaps in Current Literature 81

Strengths and Weaknesses of Current Studies 85

Similar Research Methods 87

Summary 93

3 Methodology 94

Overview 94

Research Design 96

Research Procedures 97

Prototype Environment 100

Threats to Validity 108

Sample 121

Data Analysis 123

Data Formats for Results 127

Resource Requirements 130

Summary 135

4 Results	136
Research Goals	136
Review of the Methodology	136
Experimental Outcomes	138
Data Analysis	163
Findings	166
Summary of Findings	167
5 Conclusions, Implications, Recommendations, and Summary	169
Conclusions	169
Implications	171
Recommendations	172
Summary	174
References	184
Appendix A – Advanced Detection Studies	191
Appendix B – Cuckoo Installation and Configuration	193

Chapter 1

Introduction

Background

Malware (malicious software) represents a significant problem for today's highly networked and distributed computer systems (Cesare & Xiang, 2010). In general, malware is used to represent a variety of annoying or hostile software programs specifically designed to access, interrupt or establish communication channels, and/or perform data exfiltration from compromised computer systems without the owner's informed consent (Kumar, S., Rama Krishna, C., Aggarwal, N., Sehgal, R., & Chamotra, S., 2014). The ability to detect malware is a major challenge due to the proliferation and complexity for detecting malware across a wide range of endpoints throughout the enterprise (Qu & Hughes, 2013). The growth of malware threats continues to far exceed the security industry's projections and estimates (Qu & Hughes, 2013). According to the McAfee Labs Threat Report (2014), the security company collected over approximately 235 million malware samples in the first quarter of 2014. In 2015, McAfee estimated that the company collected over 3 million new samples a day (McAfee, 2016). A majority of samples collected were detected by McAfee's worldwide sensor network protecting computer systems and networks for both commercial and Government organizations (McAfee, 2014). These samples were collected based on the suspicious activity or behavior of the malware by typical network protection devices and security appliances (McAfee, 2014). The unprecedented growth of new malware increased by 174% for the same quarter from the previous year (2013) and exceeded an additional 100 million new

samples (McAfee, 2014). McAfee's 2015 Threat Report stated that malware rose by 13% in first quarter (McAfee Labs, 2015). Advanced detection methods to detect new malware continues to be a major challenge for security researchers and commercial security companies given the growth and sophistication of malware (Qu & Hughes, 2013).

The ability to detect and classify malware provides a means to identify and stop the proliferation of malware across the network. Today's detection relies upon signatures of known malware to identify malware at the network and host (Cesare & Xiang, 2011). Signature detection of advanced malware is complicated by polymorphism (Muhaya, Khan, & Xiang, 2011). However, signature detection alone does not provide adequate protection for networks and end points from polymorphic malware. The McAfee Threat Report (2016) stated that on average the security saw almost 40 million new malware samples in each quarter for 2015.

In order to understand the malware threat landscape – it is helpful to understand the various types of malware and the motivational factors driving malware development. Malware is often developed for a specific purpose to attack or disrupt a specific target or environment (Richardson, 2011). Typically malware aims to steal information, compromise sensitive networks, and establish launch points for future attacks (Brenner, 2008; Chu, Holt, & Ahn, 2010; Wall, 2012). The term “malware” includes viruses, Trojan programs and other malicious tools installed on hosts or endpoints to automate some type of compromise that may lead to more sophisticated or complex intrusions (Brenner, 2008; Richardson, 2011; Wall, 2012). Another type of malware – botnets extends this capability and combines multiple aspects of existing malware into a single

program enabling hackers to form a network of infected computers world-wide (Bächer, Holz, Kotter, & Wicherski, 2005). Botnets can provide a foothold to create a number of attacks ranging from the distribution of spam, denial of service attacks, and advanced network scanning (Bächer et al., 2005). There has been significant research involved with malware and role it plays in cybercrimes and the financial impact attacks play on organizations (Bächer et al., 2005; Wall, 2012). Additional research has been involved with the social factors that influence the creation, distribution, and use of malware in the hacker community (Chu et al., 2010). Sophisticated malware enables hackers to sell access or steal information from their botnets across the world (Bächer et al., 2005). Botnets have revolutionized attacks on infrastructures and has advanced the targeted attack on organizations (Chu et al., 2010). Hackers have established forums and Internet Relay Chat (IRC) to sell and distribute malicious software, stolen data, and hacking tools that enable less skilled malware developers to gain direct access to advanced malware services (Chu et al., 2010).

Ransomware has become major malware challenge within the last few years (McAfee Labs Threats Report, 2016). Ransomware better known as “scareware” has become quite popular with cyber criminals (Kharraz, Robertson, Balzarotti, Bilge, & Kirda, 2015). Most ransomware is polymorphic in nature (McAfee Labs Threats Report, 2016). Ransomware malware and the criminals operating the malware see to take advantage of people’s fear of releasing private information, losing critical data, or facing irreversible hardware damage (Kharraz et al., 2015). In most cases ransomware locks or encrypts the victims’ computers until they make a payment to unlock or re-gain access to their data (Kharraz et al., 2015). Ransomware has been on the malware landscape for over 10 years, however the volume of these attacks has dramatically increased over the

last few years - 500% in 2013, 550% in 2014 and nearly 600% in 2015 according to Kaspersky's Threat Report (2016). The Crypto locker ransomware has been detected on over 750,000 computers in the United States (Gostev, Unuchek, Garnaeva, Makrushin, & Ivanov, 2016). Ransomware attacks have migrated from individuals to organizations such as healthcare/hospitals and local governments. Once such ransomware case involved an entire police department who had to pay the ransom in order to unlock or to decrypt the departments document repository (Gostev et al., 2016). This has led to new energy for detecting polymorphic ransomware and has mobilized many of the leading security vendors to share findings because of the levels of sophistication and the current ineffective defensive techniques (Gostev et al., 2016).

In order to improve detection advances in malware classification and detection are needed to provide better end-point protection. Malware detection has traditionally been largely signature-based (Qu & Hughes, 2013). These signatures are based on the hashing of the malware files, executables or binaries (Sulaiman, Ramamoorthy, Mukkamala, & Sung, 2005). Detection is then based on the a-priori signature or hash of the complete binary or partial characteristics of byte sequences of known malware samples (Wüchner, Ochoa, & Pretschner, 2014). Security vendors collect, analyze and create extensive repositories of malware signatures that are used for detection within anti-virus and other security protection. Signature-based detection continues to be one of the main tools used for detection on commercial anti-virus (Wüchner et al., 2014). However, signature based detection suffers from several major limitations. The first major limitation is speed. New malware released into the wild will not be detected until the signature is captured by some security mechanism and the executable/binary analyzed. In reality, once the file/binary has been captured it can take weeks for one of the many security vendors to

publish the malicious reputation for a specific file. Unfortunately, polymorphic malware can be developed and distributed via the internet across the world in minutes and hours (Mandiant Research, 2014). Polymorphic malware can then self-generate new signatures in order to appear to security devices as a new file. These new signatures will need time to be discovered, analyzed (determined malicious or benign) and released by commercial security vendors for protection (Szor, 2005). Signature-based defense will always suffer a time gap between the release of a new or unknown malware types, determination of maliciousness by the security community and signatures for the corresponding malware (Wüchner et al., 2014). Unfortunately, it is during this time gap that the malware goes undetected, reaches the intended targets and propagates itself throughout compromised networks.

Malware developers understand the time gap and exploit malware determination frequency cycle referred to as “catch, analyze, deploy” (Wüchner et al., 2014). The challenge for malware researchers is to shorten this gap or approach the problem from a different perspective (Cesare, Xiang, & Zhou, 2007). The second limitation of signature-based detection is polymorphism. Polymorphic malware is an umbrella term used to refer both polymorphic and metamorphic malware (Campo-Giralte, Jimenez-Peris, & Patino-Martinez, 2009). Polymorphic and metamorphic malware exploit the malware frequency determination cycle by dynamically creating a new executable and changing its signature. By changing the signature, most current defensive technologies ignore the executable as it has a new identity. The new executable will have a new signature and evade existing signature detection (Cesare, Xiang, & Zhou, 2013). Malware developers use the new signature to propagate and infect a large number of hosts (Wüchner et al.,

2014). The detection of malware is increasingly more difficult over time due to the sophistication of the malware programs (Guri, Kedma, & Sela, 2013). Previous polymorphic detection rates using advanced classification range from 68.75% to 81.25% (Amos, Turner, & White, 2013). These detection rates used Bayes and Multilayer Perceptron techniques to establish a baseline for effective detection of various types of polymorphic malware (Amos et al., 2013).

Malware researchers have developed techniques to break the polymorphic malware executable into segments and attempt pattern matching on malware segments (David & Yahav, 2013). This approach has been extremely helpful as malware developers tend to re-use and share functions (Rekdal & Bloemerus, 2013). However, advanced malware developers understanding this approach employ other techniques to protect code from being analyzed by encrypting segments of execution code and leveraging other obfuscation techniques to avoid sub-function signature development (Campo-Giralte et al., 2009). Malware developers design malware to invade, self-mutate and propagate malware throughout the infected network using advanced techniques that are generally used one-time making the research very difficult (Bossert, Hiet, & Inria, 2014). Detection of advanced malware grows increasingly more difficult because of the sophistication and concealment of the executable. Malware detection and classification research methods must be advanced in order to detect new types of malware. Advanced malware analysis methodologies need to be developed in order to improve detection and classification the evolving nature of polymorphic malware.

Identification of malicious or benign programs (files, portable executables, etc.) must involve new discovery techniques and leveraging several types of advanced analysis

to identify malware (Cesare & Xiang, 2013). There are typically two types of advanced analysis – static and dynamic. Static analysis never executes the malware, but rather decodes the compiled code into human readable segments (Sulaiman et al., 2005). Dynamic analysis is typically conducted via a platform referred to as a “sandbox”. The sandbox provides an environment to execute the program and captures each sequence of the runtime behavior (Cesare & Xiang, 2014). Both analysis tools are extremely helpful during malware analysis as researchers are able to understand how the program executes and what functions are actually executed by the malware (Sulaiman et al., 2005). Static and dynamic analysis allows researchers to evaluate and determine the intent of the malware programs and the potential targeted environment. Researchers can then classify malware into families of malware based on the results from static and dynamic analysis. The ability for advanced analysis to produce classification and signature detection provides a means to perform advanced detection.

Advanced detection relies on feature extraction of the malware being analyzed (Qu & Hughes, 2013). Recent malware research has divided detection into two tasks: malware classification and malware detection (Qu & Hughes, 2013). Classification of malware enables researchers to understand the family of malware and generally the intent of the attacker (Qu & Hughes, 2013). Detection involves the identification of novel instances of malware and detecting copies or variants of known malware (Qu & Hughes, 2013). Both classification and detection requires feature extraction of the malware binary and class features associated with the malware (Cesare & Xiang, 2013). Detection of new malware relies heavily on statistical machine learning (Qu & Hughes, 2013). Searching for malware variants uses the concept of similarity searching to query a

database of known malware types and functions (Qu & Hughes, 2013). Similarity queries or nearest neighbor searches relies upon techniques within machine learning or referred to as instance-based learning (Yerima, Sezer, McWilliams, & Muttik, 2013). Instance-based learning utilizes inherent distance functions within machine learning to highlight similarity or dissimilarity between objects (Yerima et al., 2013) . These search algorithms use distance functions and develop mathematical metrics for various searches and are much more efficient than traditional database queries (Yerima et al., 2013).

Traditional research and commercial malware detection systems have relied upon static string signature searching techniques to classify and detect malware (Cesare & Xiang, 2011). This type of search relies upon capturing sections of malware executable code that uniquely identifies the malware (Cesare & Xiang, 2011). String signature searching performs well from a query performance and produces low false positive results. Therefore, string searching is heavily utilized in real-time systems for malware detection (Cesare, Xiang, & Zhou, 2013). Unfortunately, string signature searching performs poorly when searching for polymorphic malware variants. Poor string search performance for polymorphic malware variants is due to the closely related but non-identical signatures (Cesare et al., 2013). Polymorphic malware variants have similar properties but the malware changes between instances causes ineffective searches. Additional research needs to be performed in order to find advanced search techniques for polymorphic and metamorphic malware.

The advancement in functionality and behavior of computer malware can be categorized into five distinct generations (Noreen, Murtaza, Shafiq, & Farooq, 2009). The first-generation malware was quite simple, i.e., they caused infection by simply

attaching themselves to the code sections of benign executables (Noreen et al., 2009).

The malware in second generation had some additional functionality such as self – replication (Noreen et al., 2009). The malware of third generation had stealth capabilities that make this type more difficult to detect for signature-based detection (Noreen et al., 2009). The malware belonging to the fourth generation uses armoring techniques to protect the code from static analysis (Noreen et al., 2009). Finally, the malware of the current generation use polymorphic techniques to encrypt functions and obfuscate their code with every replication (Noreen et al., 2009). Most of today’s polymorphic malware utilizes armoring and obfuscating protections to hide and propagate themselves across the environment.

The development and analysis of today’s malware requires experienced programmers. The development of malware requires years of experience due to the sophistication and nature of the programming. Therefore, malware developers have “malware creation engines”, which generate different versions of malware in order to re-use logic and functions through and a complex set of algorithms (Noreen et al., 2009). The malware creation engines use various techniques such as compression, function insertion and instruction substitution to develop a new variant (Noreen et al., 2009). However, these variants can be discovered through static and dynamic analysis as they have the same functionality and semantics (Noreen et al., 2009). Thus, polymorphic and metamorphic malware retains similar functionality and executable code as it replicates itself. These functions and code base can essentially provide genetic markers for detection (Pfeffer, Call, & Chamberlain, 2012).

The goal of polymorphic and metamorphic malware is to dynamically generate new versions of malware in order to evade signature detection techniques (Qu & Hughes, 2013). Today's malware is capable of mutating or creating copies of itself making it very difficult for signature-based detection to be effective (Qu & Hughes, 2013). Therefore, malware research must consider new techniques to identify and detect malware variants to improve endpoint or host security (Qu & Hughes, 2013).

Problem Statement

There is currently no single method to effectively detect polymorphic malware at the host or end-point. Advanced malware is spreading across the enterprise through internet downloads, email attachments and malicious hyperlinks, and mobile devices/hosts and is a major challenge for system-owners and cyber-security professionals (Campo-Giralte et al., 2009). Signature-based defense approaches such as anti-virus software do not provide adequate protection from polymorphic malware (Campo-Giralte et al., 2009). According to McAfee's report (2016), the security vendor collects over 3 million new samples of malware per day. Due to the overwhelming number of new and advanced malware, new detection techniques are needed to discover advanced malware and must be researched to provide improved detection (Borojerdi & Abadi, 2013). New detection methods must be relied upon to examine untrusted programs and prevent malware from causing damage and disruption to computer systems (Cesare & Xiang, 2013). Better detection malware detection methods may prevent the proliferation of end-point infections. Identifying and preventing malware from executing improves overall system security and represents significant cost avoidance savings for organizations (Qu & Hughes, 2013). New malware research must provide new detection

techniques to identify and classify today's sophisticated malware (Qu & Hughes, 2013).

The research must also understand the sophistication of today's malware and the evolving nature of targeted malware attacks in the future (Mandiant Research, 2014).

This research focused on developing improved detection rates using various machine learning cluster algorithms with a limited set of multidimensional topological features. As part of the study, the research developed a feature extraction methodology, developed a prototype environment and leveraged a testing protocol to better understand advanced clustering algorithm performance and feature weighting to improve detection for host or endpoint protection.

Dissertation Goal

The goal for the research was to develop an experimental prototype system to provide improved detection for polymorphic malware. The prototype system utilized various feature extraction methodologies for all types of samples. These features were assembled and evaluated with advanced clustering algorithms within machine learning to deliver improved detection. This study quantitatively examined and prototyped various techniques to test the theory that machine learning with advanced clustering can provide improved detection for polymorphic malware. This study is grounded in previous quantitative experiments that leverage supervised and unsupervised machine learning for malware research (Boro, et al., 2012; Pradesh, 2014). Further, previous research provided a basis for various feature extraction techniques and feature isolation models to enhance malware detection for previous research using machine learning (Chaumette, et al., 2011; Devesa, et al., 2010). This study extends previous malware detection techniques by combining file properties, static and dynamic analysis for feature

extraction. The feature extraction methodology used to assemble the multidimensional topological data and became the input (dataset) for machine learning. Based on the research of nearly 40 related and recent malware studies, it is believed that this is the first study to evaluate the multidimensional topological features (file properties, static analysis and dynamic analysis) using advanced clustering algorithms. Ultimately, the detection of polymorphic malware was the focus and emphasis for the study. Current literature offers detection rates for polymorphic detection using advanced classifiers range from 68.75% to 81.25% (Amos et al., 2013). The study developed deliver detection rates of 99.43% well above 81.25% with better than acceptable accuracy rates.

As part of the study, a feature extraction methodology used an integrated analysis environment to perform feature extraction from all samples (malware, benign and unknown) samples as input for machine learning. The feature extraction methodology constructed produced a rich dataset for each sample. Feature extraction was achieved through static analysis and dynamic analysis. This multidimensional topologic approach delivered the unique dataset for each sample used for polymorphic malware detection.

The study evaluated detection rates for three advanced clustering algorithms using machine learning within WEKA. The clustering algorithms used for this studying included: 1) Advanced Ensemble Classification (Bootstrap Aggregating (Meta Bagging (MB)), 2) Instance Based k-Nearest Neighbor (IBk) and 3) Deep Learning Multi-Layer Perceptron (DLMLP). The study demonstrated improved malware detection based on advanced cluster analysis using multidimensional topological data that delivered true positive detection rates above the established baseline range of 68.75% to 81.25% for polymorphic malware (Amos et al., 2013). Additionally, the research delivered improved

false positive rates well beyond the range 15.86% to 33.79% established by the same study (Amos et al., 2013). The ability to achieve these goals was based on developing an environment to analyze, extract, assemble and classify the malware.

The Instance Based k-Nearest Neighbor (IBk) algorithm was chosen to replace LSH as the algorithm improves upon the “locality” aspects of the clustering algorithm. The IBk is a k-nearest-neighbor classifier that utilizes a similar distance metric used in the LSH algorithm. The calculated Euclidean distance function is used by IBk as a critical search parameter within the algorithm (Manikandan, Ramyachitra, Kalaivani, & Ranjani Rani, 2016). The IBk algorithm was selected as it provided newer techniques for clustering outcomes.

The Deep Learning Multilevel Perceptron (DLMLP) clustering algorithm was selected to replace the BP algorithm. Recent literature discussed the benefits of DLMLP over BP as it improves upon the dimensions of belief for training and test datasets (Gruber, Cammerer, Hoydis, & Brink, 2017). The main idea in replacing the BP algorithm with DLMLP was recently presented by Gruber et al. (2017). New research has found that artificial neural networks (ANN) can leverage belief propagation for clustering or classification but do so inefficiently (Gruber et al., 2017). ANN’s build networks of neurons, share information between neurons and propagate results throughout the network using weights or beliefs (Gruber et al., 2017). This approach to propagation is done so inefficiently as it is done many times throughout the entire network (Gruber et al., 2017). A more efficient way to achieve similar or better results is to establish shallow neural networks and combine other algorithms to achieve similar propagation (Gruber et al., 2017). The goal was to utilize the DLMLP algorithm for

improved detection. Given that the goal of this research was to establish new detection using advanced algorithms. DLMLP was selected for model efficiencies and to leverage nascent algorithms that improved upon other algorithms.

Research Questions and/or Hypotheses

This study quantitatively examined and prototyped various techniques to test the theory that machine learning with advanced clustering can provide improved detection for polymorphic malware. Therefore, the research question was - Can machine learning utilizing clustering algorithms with multidimensional topological feature extraction deliver improved detection for polymorphic malware? The research did in fact deliver improved malware detection given multidimensional topological data extracted from static analysis, dynamic analysis and file properties. This study developed an experimental prototype that analyzed over 1M samples using MB, IBk and DLMLP. These samples were used to evaluate polymorphic malware detection using machine learning with advanced clustering.

Previous results suggested that various cluster algorithms could deliver impressive detection results with limited datasets (Fraley & Figueroa, 2016). Initial research demonstrated various clustering algorithms produced impressive detection rates with machine learning for smaller datasets. This study analyzed much larger datasets. Each algorithm was allocated a dataset containing 200,000 samples containing known malware, known benign and unknown or undetermined files. Earlier demonstration results produced well above the 81.25% for a small dataset (Fraley & Figueroa, 2016). The research produced detection rates for MB, IBk and DLMLP that exceeded 99.99%.

Feature extraction was accomplished using both static and dynamic analysis become the inputs for machine learning. The multidimensional topological data contained file features, static analysis features and dynamic analysis features for input for the experimental research model to demonstrate improved detection (Liu, Chen, & Guan, 2012; Tamersoy, Roundy, & Chau, 2014). These features were utilized with machine learning with advanced clustering algorithms to produce the achieved detection rates.

Based on the research of nearly 40 related and recent malware studies, it is believed that this is the first study to evaluate the combination of topological features (file properties, static analysis and dynamic analysis) with machine learning using advanced clustering. Most studies use either static or dynamic analysis. Ultimately, the focus of the study was to demonstrate improved detection rates for malware using multidimensional topological features with machine learning utilizing larger datasets. It is important to note that polymorphic malware has the ability to change behavior and execute different embedded functions over time (Campo-Giralte et al., 2009). Therefore, detection of polymorphic malware becomes more difficult and baseline detection rates range from 68.75% to 81.25% (Amos et al., 2013). Using this approach, the study was able to produce detection rates for MB, IBk and DLMLP that exceeded 99.999%.

This study developed a prototype to quantitatively evaluate malware detection effectiveness and accuracy. Accuracy can be expressed as the Number of correct assessments divided by the Number of all assessments. Effective detection rates were evaluated using standard statistical measures such as True Positive Rate (TPR), False Positive Rates (FPR), True Negative Rate (TNR) and False Negative Rate (FNR). Using these statistical measures allowed deeper understanding for the various detection rates.

The focus of the experimental prototype was to evaluate detection rates for the various clustering algorithms given the various features, weighting of the features and the advanced cluster algorithms used with a series of machine learning tests. There are a number of statistical measures to be utilized to evaluate malware detection performance. Sensitivity and Specificity are two standard measures that evaluate effective detection rate (Kolter & Maloof, 2006). Sensitivity and Specificity statistical measures have been used to evaluate malware detection performance for a number of previous malware detection experiments (Kolter & Maloof, 2006). This analysis is provided later in the report.

Relevance and Significance

There is currently no single method to identify and detect polymorphic malware at an endpoint. Advanced malware affects almost every organization, business and government entity connected to a network and internet (Pramono & Suhardi, 2015). Traditional signature-based detection systems are ineffective due to the dynamic nature of polymorphic malware (Cesare et al., 2013). Polymorphic malware can be delivered through a number of distribution channels: email, embedded files, program updates, and internet web sites (Cesare et al., 2007). Major security companies spend billions of dollars to solve general malware issues (McAfee, 2014). The detection of polymorphic malware is more difficult because of the sophistication of the malware programs (Guri et al., 2013). Polymorphic malware developers continue to advance malicious programs to evade or avoid detection by security protection devices (Qu & Hughes, 2013). There is no single answer for detecting polymorphic and metamorphic malware. However, researching polymorphic malware may provide greater insight into classes of malware

and effective detection techniques for various classes. Malware research continues to be needed in order to provide details regarding the evolving nature of polymorphic malware.

Many of today's cyber-criminal activities on the internet can be directly attributed to malware or malicious programs (Ulrich Bayer, Kirda, & Kruegel, 2010). Malware comes in different sizes and shapes – Trojans, viruses, bots, etc. and give miscreants a wide range of possibilities for achieving nefarious activities (Ulrich Bayer et al., 2010). As a result, security companies see a huge number of new malware samples each day – McAfee estimates at least 30 per second according their most recent report (McAfee, 2014). Globally speaking, in 2014, cybercrime is estimated to cost businesses more than \$400 billion a year (McAfee Labs, 2015). Other experts believe that a \$400 billion estimate is conservative and the total cost of cybercrime in 2014 could have approached nearly \$600 billion worldwide (McAfee Labs, 2015). The rate of growth for cost impact has soared from \$56 billion in 2004 to today's \$400 billion estimate (Schneidewind, 2010; McAfee Labs, 2015). In 2011, the Federal Bureau of Investigation (FBI) reported that on average, companies that report a network breach have lost an average of \$150,000 per incident (Richardson, 2011). Data breaches today are estimated to cost US businesses at least \$200 billion (McAfee Labs, 2015). Many of these breaches have been attributed to unauthorized money transfers and account hijacking on mobile devices (A. Sharma & Sahay, 2014). Being able to detect and identify malware infections can deny hackers financial reward, improve security and reduce the organizational cost of mitigating these infections.

This research was built upon the current knowledge base for polymorphic malware research by improving detection through machine learning. This research developed an experimental approach to improve advanced detection by leveraging

multidimensional feature extraction for three advanced clustering algorithms using machine learning. By utilizing this experimental approach, this research extends the body of knowledge for detection of polymorphic malware. By achieving improved detection rates for polymorphic malware there is the potential to reduce the financial impact on businesses worldwide.

Barriers and Issues

There were a number barriers and issues with the experimental research. The first and most imposing issue is obtaining the malware data set. In order to perform this research, the malware data set needed to contain representative samples of contemporary polymorphic malware. These data sets are usually collected by large security commercial organizations and are shared with partners for a fee. The researcher was able to obtain a sufficient sample datasets for known malware, known benign and a collection of unknown samples. The researcher was able to secure over 2 million total samples in order to develop each of the sample datasets. The collection consisted of 1,009,108 known malware samples, 756,322 known benign samples and 748,976 unknown files. In addition, the researcher ensured that known malware samples in the dataset were targeting Microsoft operating systems. Obtaining the collection of samples required and consumed a significant amount of time, a number of transportable drives and long data transfer times.

The second issue is that in addition to the malware data, the expanded data set must contain representative non-malware and unknown samples. The non-malware data set must be included as part of the research in order to evaluate the effectiveness of detection. A number of vendors were able to share a large amount of the benign files. In

particular, Virus Total provided 90% of the 756,322 known benign samples. Unknown samples also needed to be included as part of the study. A number of vendors were able to share “undetermined” or “unknown” samples as part of this study. Virus Total and Malwarebytes provided unknown files samples for this study. Contemporary non-malware has characteristics and behaviors that protect intellectual property of commercial software. These non-malware behaviors and characteristics are often mimicked by malware developers in order to bypass detection. This research evaluated known malware, known benign and unknown samples in order to evaluate overall detection effectiveness (False Positives and False Negatives). Initially, it was thought that the benign samples and unknown samples would be more difficult to obtain than the malware samples. This was not the case, the malware samples targeting Microsoft operating systems were much more difficult to obtain.

A related dataset issue was storing and securing the experimental dataset. The objective of the research is to analyze millions of samples in order to evaluate efficacy and validate the proposed detection approach. The study required over 2 million samples. It was expected that the storage for malware and non-malware samples would not exceed 1 TB. However, this was underestimated. Malware samples are typically stored in a .zip compressed file format with a password. In many cases, vendors who share samples also used the Unix TAR command to ship compressed files (30-35 GB for 15 Tarballs). This limited the network bandwidth and storage needed for these files. However, the uncompressed files consumed over 1.3 TB of hard drive storage.

Once the various test data sets had been pseudo-randomly generated, the feature extraction process would simply use the unique id (file hash) associated with the sample

for feature extraction. As a separate process, selected files needed to be unzipped and password supplied in order to access the sample. Additional unique identifiers were developed as many of the files had a mixture of MD5, SHA1 and SHA256 hashes. All files obtained, known malware, known benign and unknown files were stored in a secure and protected repository. The secure storage of malware data set was needed to ensure non-detonation within the experimental storage environment. The malware samples were kept in a state such that the malware did not detonate or infect the analysis platform/environment while stored.

Lastly, the software tools required for the prototype environment did have represent some challenges. Sufficient software and hardware computing resources had been acquired prior to beginning study. However, the open source version of Cuckoo had several new features as part of the new release. Installing the new release presented some major challenges. In addition, previous versions of Cuckoo processed samples in a different fashion and produced automatic dynamic analysis reports. The upgraded version of Cuckoo provided some advanced features that were more suitable for this study. Online resources were used to eventually assist with installing and configuring Cuckoo using the Ubuntu operating system (Kolo, 2016). Details regarding installation dependencies, networking and configuring VirtualBox is provided in Appendix A (Kolo, 2016). Managing the overall analysis environment was somewhat challenging but achievable. Difficulties with installing Cuckoo impacted schedule by a few weeks.

Assumptions, Limitations and Delimitations:

There were a number of assumptions, limitations and delimitations with the research conducted. Each of these areas are discussed in detail below.

Assumptions

There were a limited number of assumptions for this study. The first assumption was that the dataset collected from across the malware repositories represent the malware population from across the internet. The population of malware selected was malware captured relatively recently. Secondly, there was no determination made regarding the genetic relationships for the polymorphic malware samples used for the study. In other words, some samples may be related but the study assumes that these relationships do not matter for the purpose of detection. Lastly, it was assumed that the samples collected and maintained in a secure .zip container does not disturb or alter the malware sample itself. The .zip container is a common industry protection mechanism that protects working environments from malware outbreaks. All malware was secured and stored in a .zip file format in order to protect the prototype environment. It is believed that these assumptions were minimal and did not affect the research outcomes.

Limitations

This study included malware samples from various communities of interest for malware research including Symantec, Virus Total, contagio, Virus Sign, and VxHeaven. There was a limitation regarding collecting samples across these available sites. The limitation for this study was that only polymorphic samples with dates within the last eighteen months was used for detection research from these sites. Therefore, the sample set may not be representative for older malware or other malware sites. The purpose of study was to evaluate detection for current malware. Older samples may have value but more recent malware will have more value. Collecting samples from multiple sites and

then performing random selection should provide the needed sampling technique for generalizability.

Delimitations

Delimitations provide boundaries for the research or study (R. Yin, 2015). Delimitations include topics or areas that the study chose to intentionally exclude from consideration. For the purpose of this study, the three clustering algorithms have been selected due to demonstrated detection results from previous studies (Alam, Horspool, & Traore, 2013; Murphy, Weiss, & Jordan, 1999; Tamersoy et al., 2014). There are most certainly other advanced clustering algorithms that could have been used for this study. However, the intent of this research was to demonstrate that detection can be improved by adding a combination features from file properties, static and dynamic analysis. The proposed research expands current detection body of knowledge by using proven clustering algorithms with the expanded features (static, dynamic and file properties) leveraging the machine learning environment. Other delimitations include selecting only malware samples specifically targeting the Microsoft operating systems. Malware targeting other operating systems was not a consideration for this study. However, there were a limited number of samples collected that were not targeting Microsoft operating systems across the community (Symantec, Virus Total, contagio, Virus Sign, and VxHeaven). Non-Microsoft malware samples collected were minimal. Non-Microsoft malware samples were discarded prior to random selection. Microsoft targeting malware is believed to represent a majority of the targeted polymorphic malware (Ahmadi, Ulyanov, Semenov, Trofimov, & Giacinto, 2016) Therefore, this study focused on only samples targeting Microsoft.

Summary

Malware represents some of the most serious security concerns for today's Internet. Security breaches and cyber-attacks can be directly attributed to malware or multi-stage cyber-attacks. Malware can compromise networks and computers in the form of botnets, viruses, worms, ransomware and advanced persistent threats (APTs). These cyber-attacks are launched using targeted and advanced malware techniques to steal personal, proprietary or financial information. The high number of attacks and the associated negative notoriety make malware one of the most popular areas for advanced research. Much of today's advanced research has been concentrated on developing techniques to collect, study, and mitigate malware. This research focused on detecting "real" malware and samples found "live" on the internet. As improved detection becomes a reality – mitigation or elimination of malware for end-points can be greatly enhanced. Unfortunately, current host-based detection approaches that leverage signature-based detection is largely ineffective for new polymorphic malware. Polymorphic malware avoids or evades signature detection by using advanced obfuscation or encryption techniques. This research set out to address these shortcomings, new research was conducted to develop dynamic detection approaches to identify potential malware threats. This study proposed a novel malware detection approach that provided improved detection for polymorphic malware. The research should enhance and compliment traditional end-point detection approaches. This study's approach extracted key features from file properties, static and dynamic analysis and through advanced cluster analysis determined the likelihood of files (samples) to be benign (good) or malicious (bad). The proposed approach analyzed the malware executable (program) in a controlled environment in order to better understand behaviors,

function calls and the inclusion of dynamic libraries. The research conducted leveraged this information through machine learning to improve detection. In order to better understand the malware threat landscape, a review of past research literature and previous malware studies is provided next.

Definition of Terms

Advanced Persistent Threat: A deliberately slow-moving cyber-attack that is applied quietly to compromise information systems.

Anomaly detection: The search for network connections which do not conform to an expected normal traffic.

Bot: A term short for robot. Criminals distribute malicious software that can turn a computer into a bot. When this occurs, a computer can perform automated tasks over the Internet without one's awareness.

Botnet: Criminals use bots to infect large numbers of computers. These computers form a network, or a botnet. Criminals use botnets to send out spam email messages, spread viruses, attack computers and servers, and commit other kinds of fraud.

Bring Your Own Device (BYOD): Mobile devices that are personally owned, not a corporate asset.

Clustering: Method that organizes objects with similarities into one cluster, and objects with dissimilarities into other clusters.

Crimeware: Tools that drive hackers' attacks and fuel the black market (e.g., bots, Viruses, Trojan, spyware, adware, etc.)

Cryptocurrency: A digital medium of exchange that uses encryption to secure the process involved in generating units and conducting transactions.

Cyber Attack: An attack, via cyberspace, that targets an enterprise's use of cyberspace for the purpose of disrupting, destroying, or maliciously controlling a computer environment/infrastructure; destroying the integrity of the data; or stealing controlled information.

Data Breach: An organization's unauthorized or unintentional exposure, disclosure, or loss of sensitive PI, such as social security numbers; financial information, such as credit card numbers; date of birth; or mother's maiden name.

Data Breach: An organization's unauthorized or unintentional exposure, disclosure, or loss of sensitive PI, which can include PII, such as social security numbers; or financial information, such as credit card numbers.

Data Security Incident: A violation or imminent threat of violation of a computer security policy, acceptable use policy, or standard security practice

Denial of Service (DoS): The prevention of authorized access to resources or the delaying of time-critical operations.

Detection Rate: The percentage of the number of intrusion instances detected by the system over the total number of intrusion instances present in the test set.

Distributed Denial of Service (DDoS): An approach whereby the hacker attempts to make a service unavailable to its intended users by draining system or networking resources, using multiple attacking systems.

False Alarm Rate: The percentage of the total number of incorrectly classified normal instances over the total number of instances.

Hacker: Unauthorized user who attempts to or gains access to an information system.

Indicators of Compromise (IOCs): Pieces of forensic data, such as data found in system log entries or files, that identify potentially malicious activity on a system or network.

Intrusion Detection System (IDS): Hardware or software that gathers and analyzes information from various areas within a computer or a network to identify possible security breaches, which include both intrusions and misuse.

Intrusion Prevention System (IPS): Systems that can detect and attempt to stop an intrusive activity, ideally before it reaches its target.

Malicious Code: Software or firmware intended to perform an unauthorized process that will have an adverse impact on the confidentiality, integrity, or availability of an information system.

Malware: Programs or executables targeted to infect a user's device. When successful, the hacker is able to control the user's device, which may lead to data loss or escalation in the hacker's privileges on the information system.

Mobile Device: Smart phones, tablets, portable cartridge/disk-based, removable storage media (e.g., floppy disks, compact disks, USB flash drives, external hard drives,

flash memory cards/drives that contain nonvolatile memory; NIST, 2013d, 2013e).

Personal Information: Information from individuals that can uniquely identify a specific person.

Personal Identifiable Information (PII): Information that can be used to distinguish or trace an individual's identity, such as his or her name, social security number, or biometric records, alone, or when combined with other personal or identifying information that is linked to a specific individual, such as date and place of birth or mother's maiden name.

Privileged Account: An information system account with approved authorizations of a privileged user.

Privileged User: A user that is authorized to perform security relevant functions on a computer server that ordinary users are not authorized to perform (NIST, 2013e).

Ransomware: A type of malware that encrypts files and prevents the user from accessing data until the user pays a certain amount of money (ransom) to decrypt the files

Rootkit: A set of tools used by an attacker after gaining root-level access to a host to conceal the attacker's activities on the host and permit the attacker to maintain the access through covert means.

Security Event: Any observable security occurrence in a system network.

Security Incident: A violation or imminent threat of violation of a computer security policy, acceptable use policy, or standard security practice. These include

an accessed occurrence that actually or potentially jeopardizes the confidentiality, integrity, or availability of an information system or the information that the system processes, stores, or transmits.

Security Information and Event Management (SIEM) Tool: Application that provides the ability to gather security data from information system components and present that data as actionable information via a single interface.

Spyware: Software that is secretly or surreptitiously installed on an information system to gather information on individuals or organizations without their knowledge.

Testing Phase: A detection phase in which the distance (e.g. Euclidean distance) between each test instance and the normal cluster's centroid is measured to determine whether or not that instance is normal.

Training phase: A period in which a normal profile is built and or updated.

Trojan: A computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes the program.

Unauthorized User: A user who accesses a resource that he or she is not authorized to use.

Virtualization: Hiding the discrepancy between the virtual and physical allocation of information technology resources.

Virtual Machine: A separate logical instance of resources for a user or application that in reality is shared physical hardware.

Virus: A type of malicious software program that infects computer system programs, data or operating system files and is capable of replicating itself to other systems.

Worm: A self-replicating, self-propagating, self-contained program that uses networking mechanisms to spread malicious code.

List of Acronyms

ACC:	Accuracy
API:	Application Programming Interface
BP:	Belief Propagation
CC:	Correlation Coefficient
C&C:	Command and Control
DLMLP:	Deep Learning Multi-Layer Perceptron
FNR:	False Negative Rate
FPR:	False Positive Rate
GUI:	Graphical User Interface
LSH:	Locality Sensitive Hashing
IBk:	Instance Based k-Nearest Neighbor
MB:	Meta Bagging or Advanced Ensemble Classification (Bootstrap Aggregating (Meta Bagging))
ML:	Machine Learning
ROC:	Receiver Operating Characteristic curve (or ROC curve.)
SDK:	Software Development Kit
SCED:	Single Case Experimental Design
SCED CCD:	Single Case Experimental Design Changing Criterion Design

SVM: Support Vector Machine

TNR: True Negative Rate

TPR: True Positive Rate

Chapter 2

Review of the Literature

Overview of Topics

New malware development techniques render current signature protections for polymorphic malware practically useless from a timeliness perspective (Rodríguez-Gómez, Maciá-Fernández, & García-Teodoro, 2013). Being able to detect polymorphic, metamorphic and zero-day malware requires advanced detection techniques that provide rapid adaptation, scalability and produce low false positive rates (Borojerdi & Abadi, 2013). There are numerous research studies that offer attractive alternatives for detecting polymorphic and metamorphic malware. Given the security issues concerning malware, it is not surprising that a majority of the today's security research is focused on developing enhanced detection using techniques that collect, study, and mitigate malicious code (Kolbitsch et al., 2009). Some studies are focused on botnets and botnet networks, others are focused the infected executables from websites and others study behavioral aspect of Windows and mobile devices (Seigneur & Kölnendorfer, 2013). There are other studies who strictly look at the Windows API or system calls (Ye, Wang, Li, Ye, & Jiang, 2008). New research attempts to capture a comprehensive snapshot of malicious behaviors and activities in order to classify the malware sample in question (Cesare & Xiang, 2013). The crucial aspect for this and other malware research is to understand the significance of the malware problem, investigate new ways to detect the multitude of malware types and then benchmark those results against the current approaches. An overview of malware and the various aspects of malware are presented next.

Malware is software or a set of programs whose sole purpose is to damage, disrupt or steal information from computer systems or networks (Kauranen & Makinen, 1990). Malware is often a broader term that includes viruses, worms, Trojans, botnets, backdoors, exploits, etc. The most well-known type of malware is a virus. The term “virus” was initially used by Fred Cohen in 1983 while conducting research for his dissertation at the University of Southern California (Cohen, 1985). The fundamental reason for creating such software or set of program is to create chaos, disrupt business or seek financial motives by harming computer systems (Kauranen & Makinen, 1990). The creation of various types of malware has launched an entire commercial industry known as “anti-virus software” with revenues skyrocketing to several billion dollars (Noreen et al., 2009). Malware has been a major threat to computer and networks since the early 1990s (Noreen et al., 2009). However, the malware sophistication has significantly improved since the 1980’s when Fred Cohen coined the term. Cohen’s virus-based research would propel numerous other researchers and discovery of “classes” of malware.

Another type of malware is something referred to as a Worm. A Worm is defined as a type of malware that exploits vulnerabilities of unpatched systems with self-propagation means to spread pervasively throughout a network (Weaver, Paxson, Staniford, & Cunningham, 2003). Nazario (2004) describes worms as having the ability to take advantage of system vulnerabilities that enable propagation via a network and allows the execution of arbitrary code on a remote system. Largescale worm outbreaks have decreased significantly since the early 2000s. Panda Security (2014) report that Worm malware account for approximately 6% of all malware infections in the first quarter of 2013. Zero-day worms are still today a real concern for system and security

professionals. Zero-day worm attacks still represent a real threat to organizations due to the lack of detection and the availability of high speed networks (Kaur & Singh, 2014). Organizations that have highly interconnected networks are susceptible to such worm malware attacks and should be a major concern for Internet users (Kaur & Singh, 2014). Worms also can deliver other types of malware such as Trojans.

Trojans invade systems and reside within the system in order to execute commands or instruction given by external threat actors (Gordon & Chess, 1998). Trojans enable threat actors to take control of compromised systems (K. Chen, Zhang, & Lian, 2013). Trojans conceal themselves inside computer system with the hopes of not being discovered. Some Trojans are active soon after being installed. Other types of Trojans wait for an instruction or some condition to be satisfied before executing (K. Chen et al., 2013). After receiving remote instructions, Trojans can transact and receive commands in order to disrupt computer operations, gather and exfiltrate sensitive information or attempt to gain access and privileges on host computer systems (Gordon & Chess, 1998). Trojans tend to be detected readily by signature-based detection such as anti-virus and other end-point protection software. However, systems are still susceptible to new Trojans attacks if user awareness and system hygiene are not addressed (K. Chen et al., 2013). Zero-day attacks taking advantage of system vulnerabilities are still real possibilities. However, systems policies requiring analysis of first time run executables make this type of attack less possible.

Botnets represent one of the greatest infrastructure threat to the Internet (Barakat & Khattab, 2010). For years, the research community has investigated and described the impending issues and proposed countermeasures for disarming this capable adversary.

Botnets is typically a term to describe a network of infected end-hosts that become bots under the control of a bot master (Barakat & Khattab, 2010). The bot master represents known controls from a human or prescribed operations known as a bot-master. Cyber criminals and adversaries use botnets to launch Denial of Service (DOS) attacks, Distributed Denial of Service (DDoS) attacks and aid in the propagation polymorphic across the enterprise (Li, Duan, Liu, & Wu, 2010). Detection for bots come in the form in the form of monitoring for abnormal network, memory and system behavior. Unfortunately, botnet malware can remain dormant for long periods of time and become active for a short period of time for a special purpose (J. Zhang et al., 2011). Unless systems are closely monitored, botnet malware has completed execution before notification and remediation action could be taken (Rodrigues, 2011).

Backdoors represent a malware mechanism that allows attackers surreptitious access to a computer systems (Y. Zhang & Paxson, 2000). Backdoors have existed for many years and were initially designed into operating systems in order to facilitate access by system administrators (Bohra, Neamtiu, Gallard, Sultan, & Iftodet, 2004). Today's systems are supposed to be free of backdoors that facilitate unauthorized access to the computer systems (K. Chen et al., 2013). Typically, malware backdoors embed themselves into systems and networks in order to provide a means of repeatable access for external attackers. Backdoors can exist for both interactive and non-interactive services on systems (K. Chen et al., 2013). Interactive services simply run commands or carry out instructions on the compromised system (Y. Zhang & Paxson, 2000). Non-interactive services include services such as relaying email spam or file transfer services for data to be exfiltrated outside the organization (Y. Zhang & Paxson, 2000). Backdoors

are very difficult if not impossible to detect and they take advantage of authorized services, ports and protocols.

Taking advantage of flaws within a computer system, network or mobile platform requires that a vulnerability exist and attackers have the means to exploit a weakness (Ritchey & Ammann, 2000). Malware exploits target these vulnerabilities and seek to gain access to and somehow compromise the system or network (Rodrigues, 2011). Computer systems, networks and mobile devices will always have some underlying vulnerability. System designers and developers try to minimize the exposure of such weaknesses (Ritchey & Ammann, 2000). System and network vulnerabilities may be introduced by other activities such as system integration, system operations and maintenance or poor system or network hygiene (Kim & Hong, 2014). However, today's connectivity to networks make malware exploits more possible as attackers take advantage of network connectivity to probe and understand system vulnerabilities. Malware exploits target networks, servers and applications in order to disrupt operations or steal information for financial gain (Alam, Horspool, & Traore, 2014). The detection for malware exploits is even more difficult as they appear to simply be using standard network and authorized system calls.

Ransom Malware or "Ransomware" has become quite prevalent over the past few years. Ransomware has also become known as scareware as cyber criminals use the malware to prey on the fears of infected computer users by stealing or encrypting user data (Kharraz et al., 2015). Ransomware preys on people's fear of losing control of personal or private information, losing access to highly critical or sensitive data or damaging hardware such that access is no longer possible (Kharraz et al., 2015). Certain

types of ransomware will essentially encrypt data or system files such that access to the system or data can only be achieved by unlocking the system with a “key” (Gazet, 2010). Ransomware generally provides payment information and once the transaction is complete, users are provided decryption keys in order to regain access (Gazet, 2010). Payments are generally made in bitcoin or some other method that makes tracing the payment to the recipient more difficult or near impossible (Kharraz et al., 2015). Although ransomware has been around for over 10 years, not until recently has the volume of ransomware raised major concerns within the security community. According to Symantec (2014), the number of ransomware attacks increased by over 500% on 2013. In addition, ransomware has been in the press with attacks on local police, hospitals and small municipalities. In 2013, the Cryptolocker ransomware was in the press for infecting nearly 250,000 computers worldwide (Symantec Corporation, 2014). Given the substantial growth and spotlight on ransomware attacks, developing protection and detection mechanisms should be a major research area for the security community. However, detecting ransomware and protecting organizations from this type of attack is difficult without having insight into the tactics and sophistication of these attacks.

The discussion above highlights the various types of malware and the impact that they play on system owners, security professionals and ordinary end-users. Protecting, detecting and remediating malware has become one of the fastest growing markets in the technology sector. More research is needed in order to keep pace with the advanced malware and detection become more critical each day.

Synthesis of Current Literature

Several researchers have described the history of malware in “waves” or levels of sophistication. The first wave spans from the late 1970s through the early 1990s (T. Chen & Robert, 2004). The first malware outbreak or “wave” can be described as inquisitive, exploratory and unsophisticated (Wüchner et al., 2014). The first wave was really considered to be exploratory or accidental with no clear malicious intent (T. Chen & Robert, 2004). This period can be symbolized by the first self-spreading internet malware (Morris Worm) that infected approximately 10% of the computers connect to the internet (T. Chen & Robert, 2004).

The second and third wave covers almost eleven years from 1990 through 2001. The second wave covers the 1990 through 1999 and can be characterized as the first use of polymorphic malware and encryption techniques to evade detection by anti-virus (T. Chen & Robert, 2004). The third wave spanned only a couple of years 1999 through 2001 and was consumed with malware being distributed through email (T. Chen & Robert, 2004). Malware such as the Melissa, PrettyPark, or LoveLetter viruses were distribution through email and executables had additional functionality that allowed to propagate and maintain persistence (T. Chen & Robert, 2004). The third wave also was the first wave where malware achieved remote system access in order steal sensitive information.

The fourth wave considered to take place from 2001 through 2009. The rise of malware during this era was characterized by increasing sophistication of malware that leveraged multiple vulnerabilities to infect and propagate infections through instant messaging or peer-to-peer file sharing (Shafiq, Khayam, & Farooq, 2008).

The fourth wave also took advantage of unprotected applications and application interfaces by dynamically downloading macros and additional malicious payloads (Rafique & Chen, 2014). The fourth wave also saw malware adapting to more effective detections and changes to malware developer goals and motivations. Malware such as CodeRed, Slammer, or Nimda took advantage of multiple host vulnerabilities to infect hosts and utilized email to propagate throughout networks to exploit system vulnerabilities (Rafique & Chen, 2014). The fourth wave of malware also seized the opportunity to compromise unprotected network shares and served up drive-by infections via web servers (Rafique & Chen, 2014).

The fifth wave raised the level of intent, targeting and sophistication of the malware (Wüchner et al., 2014). During this time period, roughly 2010 until present day, malware has become more targeted and developed by highly skilled professionals (Wüchner et al., 2014). Malware such as Stuxnet was developed and attacked Supervisory Control and Data Acquisition (SCADA) systems and was linked to sabotaging the Iranian nuclear program (Virvilis, Gritzalis, & Apostolopoulos, 2013). This wave was the first to have malware referred to as Advanced Persistent Threats (APTs) and Zero-Day exploits. Malware from this period is designed to attack specific systems and targets and the level of sophistication goes beyond that of previous commodity malware developed by previous waves (Virvilis et al., 2013). APTs such as Stuxnet, Duqu or Flame have been linked to Government sponsors (Wüchner et al., 2014). The motivation of these professional malware developers is economic, espionage or sabotage for targeted systems (Wüchner et al., 2014). Motivations for commodity malware are different and are more experimental and economic based (Wüchner et al., 2014).

Commodity malware is typically developed by amateurs and organized crime. However, commodity malware developers will take advantage of collaborating and sharing with professional developers to gain insight into advanced exploit methods and techniques (Wüchner et al., 2014). In addition to methods and techniques, these groups will often share concealment and obfuscation techniques needed to avoid detection (Wüchner et al., 2014). Zero-day and APTs will go unknown for long periods of time due to advanced stealth and concealment techniques (Kaur & Singh, 2014). These waves represent an ongoing threat to organizations and the internet as a whole.

In summary, the development of malware has changed over time in both motivation and sophistication. There has been a radical shift from accidental outbreaks to very targeted and specific attacks. Many of the malware attacks have economic and organizational motivational factors. Malware developers have changed over time as well from amateurs to highly skilled and trained professionals. The malware itself has adopted sophisticated exploitation, propagation and replication techniques to avoid and evade detection. Today's advanced malware use various techniques such as encryption, environmental sensing and embedded compilers to hide functions from static and dynamic analysis. Therefore, malware research spans years of research and has multiple topical areas. The synthesis of current literature is organized by topic area and is organized sequentially. The goal is to provide foundational as well as a time perspective for each malware research topic area. In some cases, there have been and continue to be arguments from various researchers as to which characteristics constitutes certain malware types or classes. The following is an overview of the foundational topics and related research in conducting malware detection research.

Virus Research

John von Neumann was the first to develop the concept of today's computer virus and is considered to be the seminal work for viruses (von Nuemann & Burks, 1966). Von Neumann developed his theory of the computer virus during the mid-1960's while speculating on the fact that programs could produce and generate "code". He conceived and developed the concept of computer programs self-replicating through what he called "reproducing automata" (von Nuemann & Burks, 1966). Although von Neumann did not develop the actual computer program, he envisioned the idea of self-replicating automata that could go viral if not addressed (T. Chen & Robert, 2004). In essence, von Neumann predicted the spawning of malware prior to the actual technical implementation of the concept (T. Chen & Robert, 2004). In 1971, Bob Thomas created the Creeper program and was the first implementation of von Neumann's virus concept (T. Chen & Robert, 2004). The Creeper program is considered to be the first virus and piece of malware released into the wild (T. Chen & Robert, 2004). The Creeper program was not developed as a piece of malware but rather to test the concept self-replication through a network (Arpanet) and to inform users of its existence on the computer once it achieved a foothold (T. Chen & Robert, 2004). The Creeper is not considered to be malware in the truest sense of the word as it was not designed to actually harm any computer system (T. Chen & Robert, 2004). However, Thomas's Creeper program is largely considered to be the "father" of all future worms and viruses for its self-replication techniques (T. Chen & Robert, 2004). Although, Thomas' program became the first program to actually achieve self-replication from a malware perspective – the study of such automata would not be studied for ten years.

Cohen's research is considered the first formal work in the field of computer viruses (Noreen et al., 2009). Cohen's seminal research on "viruses" was conducted in 1983 through 1984. His research would be later published in late 1985. Cohen is credited for bringing the attention of viruses to the larger computer community with this research (Noreen et al., 2009). Cohen explored a number of approaches to detecting viruses. His research also was aimed at not only detection but the removal of unwanted software/programs. Cohen also started to classify various forms of viruses. Cohen's detection and removal methods did not rely upon the information sharing or transitivity of information flow (Cohen, 1985). Instead his detection techniques were based on identifying various code traits used by viruses to exploit Turing machines (Cohen, 1985). Cohen's research outlines several types of computer viruses and the ability of the attacker to quickly gain administrative rights to systems once infected (Cohen, 1985). Cohen also began to describe future issues with malware and variants that would be difficult to detect. Cohen's research also described the problems with removing a virus from a system once the system was infected (Cohen, 1985). Cohen also began the process to articulate advanced classes of viruses that could mutate to avoid detection and removal (Cohen, 1985). Cohen's research would serve to accelerate other malware research and begin to paint a picture for new attacks on computer systems.

Research conducted in 1987 by Maria M. Pozzo and Terence E. Gray, provided an approach to detect modification of executable code – also known as viruses and a new type of malware known as Worms (P. K. Singh & Lakhotia, 2002). Pozzo and Gray presented various methods for detecting changes in executable code and housing a virus. Their approach analyzed the run-time executable(s) and detected whether the

executable(s) had been modified since installation (P. K. Singh & Lakhota, 2002). This early virus detection utilized encryption to store code values for the executable modules of a program (A. Singh, Walenstein, & Lakhota, 2012). The encrypted value for each module was used to create a hash value known today as “signature”. These set of signatures would be used to enable the detection of modified executables as the hash values would no longer match the values of the original module. Although the implementation approach would not be sufficient for today’s rapidly changing code environment, the signature concept would be used by an entire anti-virus industry to register, track and share virus hashes (A. Singh et al., 2012) .

Fred Cohen, in 1989, advanced a number of malware theories regarding virus detection and protection through a series of published papers in order to highlight malware issues. Cohen (1989) presented a formal model for defining computer viruses. Cohen’s virus model formally defines sets of transitive integrity-corrupting mechanisms called "viral-sets". These viral-sets contained various characteristics that uniquely identified the virus and the type of attack (Cohen, 1989b). Further, Cohen’s research explored the deeper computational properties for the defined viral-sets in order to expose the underlying code. Additional research conducted by Cohen, shifts from detection of viruses to computer systems protection from attack of these viral sets (Cohen, 1989a). Cohen also presented additional research regarding the automated detection of modified executables in order to prevent the spread of viruses of networked computer systems (Cohen, 1989b). His work illustrated various virus models and the detections needed by host computers (Kauranen & Makinen, 1990). Further, Cohen’s models were used to simulate the infection and protection of trusted and untrusted systems (Kauranen &

Makinen, 1990). These models were then used to demonstrate both theoretical and operational infections to illustrate the feasibility of viral attacks (Cohen, 1989b). Cohen's research provides greater insight into optimal protection mechanisms needed to stop the propagation of secondary infections to other systems (Kauranen & Makinen, 1990).

In 1990, Kerchen et al. at the University of California – Davis proposed to analyze malware in new ways. Their approach was to use static analysis techniques to discover whether code was indeed malware (Kerchen et al., 1990). The tools and techniques used provided heuristic tools to detect malicious code in a UNIX environment. The tools used could detect computer viruses prior to loading and executing the malware (Kerchen et al., 1990). Kerchen et al. (1990) used two tools to accomplish malware detection. The first detection tool searched for duplicate system calls in the compiled and linked program (Kerchen et al., 1990). The second detection tool used static analysis of the executable to determine the files/libraries used by the program to write to during execution (Kerchen et al., 1990). Through the use of both tools, Kerchen et al. were able to understand whether the program could be identified as a malicious or benign. The approach presented in this research would lead other researchers to investigate new ways to look the growing malware problem.

In 1990, Kephart and White began to look at computer viruses from an epidemiological model similar to those being used to perform advanced disease research (J. Kephart & White, 1991). Kephart and White performing research at the IBM Thomas J. Watson Research Center began to address malware from an immune system perspective. Their research parallels viral outbreaks and infections of the human body for computer systems (J. Kephart & White, 1991). This paper was the first published

research to compare the infection of the human body with computer virus epidemics. Their work began a theoretical discussion of viral propagation using deterministic and stochastic models (J. Kephart & White, 1991). The study also describes the conditions under which widespread computer viral epidemics would likely occur across interconnected networks. One of the key outcomes of their research was to raise the argument that imperfect defenses can still be highly effective at preventing the widespread propagation of malware (J. Kephart & White, 1991).

In 1992, Cohen presented some advanced findings regarding defensive models for computer viruses. Cohen proposed a formal definition of “computer worms” and detailed the properties that would define this class of virus. He defined “computer worms” as a malware subclass of viruses based on certain properties (Cohen, 1992). Cohen presented an alternative formal definition of a virus based on the foundational work presented by Professor Len Adelman in 1989 (Spafford, 1991). Cohen adopted Adelman’s definition of viruses based on set theory. Although these virus definitions were not specific the definition covered a broad range of replicating programs including Worms (Cohen, 1992). Cohen performed some analysis of internal code of Worms and viruses. He later went on to discern the differences between viruses and worms by the nature of the functions and self-replication needed for sustainment. Cohen demonstrated that viruses merely create replicas. On the other hand, Worms were presented to be more purposeful viruses because of their reliability, ability to spread and their ability to maintain malicious functionality with replication (Cohen, 1992). This deeper analysis provided a stark contrast of Worms and viruses. These definitions launched additional research and created classes of malware based on executable functions. The debate regarding the

characteristics for worms and viruses continues to rage throughout today's research community.

Kephart & Arnold (1994) provide extended research by identifying viral signatures from executable code using statistical methods for the various functions contained within the virus. The researchers later used the analogy of a human body with the immune system to model viruses and viral attacks (J. Kephart et al., 1995). The researchers believed that new viruses acting as "intelligent agents" could begin to infect and propagate themselves across connected networks in new ways (J. Kephart et al., 1995). In 1995, IBM was developing techniques to prevent computer infections by using biologically inspired anti-virus protection (J. Kephart et al., 1995). The researchers were early adopters for implementing neural network-based virus detection (J. Kephart et al., 1995). This early research leveraged neural network learning to discriminate between infected and uninfected programs. Further, this research was extended to identify new viruses and remove the infected files automatically (J. Kephart & Arnold, 1994).

Spafford (1994) began to define how computer viruses operate and distinguish classes of malware based on these operations. Spafford discussed the nature of true viruses and began to describe the capabilities contained in the malware. Spafford defines true viruses as having two major components: one that handles the spread of the virus, and the other delivering a payload task (Spafford, 1994). Spafford was one of the first researchers to recognize that the payload task may follow an infection and not be present initially. Instead, the payload task may await for a condition of a triggering event (Spafford, 1994). Spafford describes how viruses work and how the virus must add itself to another piece of executable code (Spafford, 1994). Spafford's research classified

various forms of computer viruses including Worms, shell code, intrusive code, and companion code (Trojans). Spafford went to great lengths to highlight that companion viruses are not real viruses unless the more encompassing definition of virus is used. Again, classification of viruses began to differentiate the malware properties. The research also draws parallels how viruses meet properties associated with life as defined by some researchers in the area of artificial life and self-organizing systems (Spafford, 1994). Spafford also begins describe an artificial "life" for viruses within computer systems and related environments (Spafford, 1994).

In 1996, Bontchev presented a new threat vector and an approach for detecting the presence of macro viruses in Microsoft Word for Windows. This type of attack was relatively new and the new “virus” would rely on the availability underlying program that supported “macros” (Bontchev, 1996). Bontchev also began to dive deeper into the MS Word macro attack. Bontchev found that the typical anti-viral software companies developed inadequate protections for this type of attack. Bontchev also demonstrated that while typical virus replicate themselves in certain ways, the Word macros lived from document to document (Bontchev, 1996). Bontchev also discussed the need for advanced integrity checking for application programs. This paper also discussed the significant threat that this type of attack could represent for application programs who enable macros and the needed protections to prevent such an attack (Bontchev, 1996).

In 1997, Kephart et al. began to contemplate and research a decade of the growing computer virus problem. The authors believed that the anti-virus community was engaged in an escalating arms race with malware developers (J. Kephart, Sorkin, Swimmer, & White, 1999). However, the authors believed that the “war” was still manageable and

winnable (J. Kephart et al., 1999). Their solution was to develop a blueprint that automated the detection and remediation viruses on computer systems. The authors believed that detection rates would increase and human experts would solve the problem in the long term using such a blueprint (J. Kephart et al., 1999). The researchers also acknowledge that the internet was becoming a fertile ground for new breeds' malware (J. Kephart et al., 1999). Kephart et al. believed that solution would be to develop a protection systems similar to the human immune system for computers (J. Kephart et al., 1999). Just as the human immune system senses the presence of previously unknown pathogens, the researchers envisioned protection for computer systems that would one day automatically detect and remove malware. However, their estimation of the growing malware problem would be severely underestimated.

In 1998, White began to identify and articulate the challenges of the growing computer virus problem (White, 1998). White's desire was to raise awareness and alarm the growing computer community of the many open issues facing researchers for virus detection and protection (J. Kephart et al., 1999). White's research highlights five problematic issues: 1) development of new heuristics for virus detection, 2) the study of viral spread and epidemiology, 3) deploying distributed digital immune system for detecting new viruses, 4) detection of worm programs and 5) proactive versus reactive approaches towards detection of virus programs. Many of these issues still exist today.

In 1998, Bontchev extended his previous computer virus by expanding the definition of "viruses". His research expanded the "virus" definition and pushed for greater understanding of the virus ecosystem. Bontchev explored the incomplete nature of definitions of computer viruses (V. Bontchev, 1998). This work discussed advanced

classification and analysis of computer viruses. Bontchev also discussed the incomplete nature of the anti-virus software and possible attacks bypassing anti-virus software. Further, Bontchev called for advanced testing methods for anti-virus software. The research also discussed system issues and social aspects of the growing computer virus problem. Bontchev also went on to discuss useful applications of using self-replicating software that are not malware.

In 1999, Jeffrey Kephart and Steve White presented an update to their previous work conducted in 1993. This updated research proposed two new epidemiological models of computer virus spread (J. Kephart et al., 1999). The two models were developed to explain epidemic and non-epidemic spread of viruses outside the laboratory environment. This work was conducted in order to explain the non-existent outbreak of viruses in the workplace. The researcher's predicted a global virus outbreak in their previous study (J. O. Kephart, 1993). However, the researcher's interest was heightened because only a small fraction of all well-known viruses seemed to have appeared in real business environment (J. Kephart et al., 1999). The researchers wanted to investigate whether the low rate of virus infections was real or if the theoretical epidemic threshold was too high and reporting was lost. The researchers develop localized model for software exchange in order to observe and explain the sub-exponential rate of viral spread (J. Kephart et al., 1999).

Worms

In 1989, Joyce Reynolds, in her work at the University of Southern California began to describe a new type of malware known as a Worm (Reynolds, 1989). Reynolds describes the infection and cure of the newly released Internet Worm. Reynolds' work

begins to envision the impact of a worm on the greater Internet community. Her work describes how Worms could propagate through a series of inter-connected computer systems and begins to paint a picture of the damage that Worms could cause through such an attack (Reynolds, 1989). Her work also evaluated the social and ethical issues of attacking the Internet ecosystem. A large part of Reynolds' work was based on reviewing and detailing the inadvertent release of the Internet Worm on the evening of November 2, 1988.

Other malware research followed and began to leverage Cohen's viral models. These models were used to describe and classify additional types of malware. Additional research efforts began to classify malware by execution properties, executable traits, program characteristics and behaviors (Cohen, 1989b). There began a debate over types of malware. The classification of malware became a hot topic as researchers began to describe all malware as viruses. Some researchers believed that a distinction between various classes of malware needed to be drawn. The malware term Worm is largely credited to the work of Spafford (Spafford, 1989). Spafford (1989) defined a worm as "a program that can run independently and can propagate a fully working version of itself onto other machines". Spafford's work at Purdue University wanted to make sure that Worms were classified differently from viruses (Denning, 1989). Spafford analyzed and described the characteristics of a Worm to self-replicate and spread itself to computer systems over a network. Spafford also explored issues with Worm replication over the integrated network known today as the Internet. Spafford is credited with dissecting the November 1988 Internet Worm incident that infected thousands of machines (Denning, 1989). Spafford describes how the Worm attack known as the Morris Worm worked its

way through a series of inter-connected computer systems. Spafford raises concerns with possible attacks on other computer systems/networks in which commerce, transportation, utilities, defense, space flight and other critical activities depended on system inter-connectivity (Denning, 1989). Other researchers would begin additional research into various malware classes.

Eichin and Rochlis (1989) also began to analyze the inadvertent Internet Worm attack in 1989 at the Massachusetts Institute of Technology. Their published paper defines the classification of the Internet "Worm" as a "virus" (Eichin & Rochlis, 1989). Their paper leverages some of the work from Cohen but began to deviate from classifying malware by executable code (Eichin & Rochlis, 1989). Instead, Eichin and Rochlis begin to describe the possible intent of development teams of releasing malware to attack specific computer targets. Their research was the first published work that focused on the targeting strategies employed to execute a specific malware attack (Eichin & Rochlis, 1989). This research also discussed the effective and ineffective defenses used by the larger "Internet" community as a whole (Eichin & Rochlis, 1989). Their work at MIT detailed a step by step account of the Internet crisis of 1988. Their work outlined the various defensive security flaws that were exploited to attack the "inter-connected" systems. Their work also described the propagation of the Worm/Virus across the Internet. This research emphasized the corrective actions needed to prevent future attacks.

Seeley at the University of Utah also analyzed the same November 1988 Internet Worm incident. Seeley analyzed the program, systems and the executables needed to propel the attack (Seeley, 1989). Seeley examined the Worm program, a 99-line

bootstrap program written in the C language and a needed object file used in UNIX systems such as VAX and Sun (Seeley, 1989). The basic goals of the attack was to locate systems across the network, penetrate those systems by exploiting security flaws with remote connections and replicate and execute the Worm (code) on the remote system (Seeley, 1989). Penetration of a remote systems was accomplished in one of three ways; 1) taking advantage of security flaws in the “listening” server, 2) exploiting the “trap door” in the SMTP mail service, or 3) guessing passwords for administrative accounts or taking advantage of non-set passwords to elevate credentials (Seeley, 1989). Seeley also outlined the defensive measures used by the Worm to prevent detection by inhibiting analysis of the program. The worm’s simplest means of hiding itself was to change the program name and directory (Seeley, 1989). Seeley’s work set the stage for analyzing malware from an attacker’s defensive perspective.

In 2002, other researchers continued to investigate the 2001 Worm attack. Moore et al. (2002) analyzed the Code Red worm which infected thousands (359,000) of hosts across the Internet in 2001 in less than 14 hours. Additional research for more advanced Worm malware highlight how the Storm Worm can be used to create botnets used by bot masters to send spam emails or perform distributed denial of service attacks (DDOS) (Kanich et al., 2008). The estimated cost of the Code-Red malware epidemic to be in excess of \$2.6 billion (Moore et al., 2002). The researchers used various techniques to analyze how multiple worm-infected computers worked to propagate the Code-Red Worm and consume network bandwidth in a targeted and coordinated manner (Moore et al., 2002).

In 2004, Williamson et al. presented the idea of Virus Throttling. Virus Throttling was a technique used to slow the spread of worms and viruses by disrupting their propagation activities largely over TCP/IP (Williamson et al., 2004). This research was conducted in conjunction with Massachusetts Institute of Technology (MIT) and Hewlett Packard (HP). The Virus Throttling concept was used to prevent an infected machine from infecting other machines on the same network (Williamson et al., 2004). The end result of such a technique was that there were fewer machines infected and there was less traffic generated by the virus over the network (Williamson et al., 2004). The technique worked well for Worms that used TCP/IP protocols and seemed to have promise for other protocols. The propagation of Worms at this time was largely attributed to the use of Instant Messaging over corporate networks. Malware being spread over Instant Messaging was a growing concern and represented a significant threat at this time (Williamson et al., 2004). Virus Throttling was a technique used to address a specific malware using certain protocols. This was a step forward in terms of addressing specific malware behaviors.

Trojans

Ken Thompson, in 1984, wrote an additional seminal paper regarding Trojan malware (Thompson, 1984). His lecture “Reflections of Trust” was widely publicized for identifying a problem known today as a Trojan malware. His presentation was awarded the 1984 Turing Award for clearly presenting, explaining, and demonstrating a practical and dangerous Trojan attack using a UNIX standard compiler (Wheeler, 2005). His presentation demonstrated how to modify the Unix C compiler to inject a Trojan piece of code into an executable program (Wheeler, 2005). The injected code modified the

operating system login program to escalate privileges and grant root access to the UNIX system (Wheeler, 2005). Thompson then demonstrated the ability to modify and recompile the compiler itself with additional code designed to detect the existence of Trojans in compiled code (Wheeler, 2005). Once the additional code was added, the “Trojan code” attack would be removed from the source code so that no source code could be detected (Wheeler, 2005). Thompson presented that these Trojan attacks could persist through numerous recompilations and cross-compilations of the compiler (Wheeler, 2005). He then presented that no level of source-level verification or scrutiny will protect systems from such embedded malware code. In fact, he described that the problem could exist at lower levels beyond the compiler such as assembly level code or even hardware microcode which would be harder to detect (Wheeler, 2005). Thompson implemented his attack on a Bell Labs UNIX C compiler and successfully launched the attack on another Bell Labs group computer systems (Wheeler, 2005). His attacks were never detected within Bell Labs and the malicious compiler was never released outside of Bell Labs (Wheeler, 2005). This research began to propel other research into Trojan malware and investigation into detection methods for executable code.

Botnets

Botnets represent the greatest threat to the internet according to multiple researchers (Barakat & Khattab, 2010). Bots represent an infected host that has a connection via a network to a botmaster (Barakat & Khattab, 2010). A botnet represents a number of infected hosts end-hosts under the command and control (C&C) of a botmaster (Barakat & Khattab, 2010). Botnets set out to infiltrate and connect more vulnerable machines to the botmaster. Recruiting additional bots is usually done

exploiting various software vulnerabilities or by propagating malware to eventually exploit a host (Vogt, Aycock, & Jacobson, 2007). All botnets are controlled by at least one command and control (C&C) channel tied to at least one botmaster but in many cases there are several primary and secondary botmasters (Vogt et al., 2007). Communication channels are established in order to receive commands and funnel information back to the botmaster (Vogt et al., 2007). These channels have become more sophisticated over time by encrypting traffic and exfiltrating key information about the end-points, configurations, platforms, networks and organization (Vogt et al., 2007). The main purpose of such a secure communication channel is to provide a command and control (C&C) medium for the botmaster's commands (Vogt et al., 2007). Botnets are constantly recruiting new bots by exploiting different software vulnerabilities for end-points, replicating itself using the same malware to other hosts and using advanced propagation techniques to spread across various networks (Barakat & Khattab, 2010). Newly recruited hosts often download the latest version of the "bot code" and run this code on the end-point typically in the background. However, this new software establishes connections to primary and secondary C&C servers. Once established and going undetected usually by running as a known service infected machines communicate and execute the commands of the botmaster (Vogt et al., 2007). Most C&C channels operate at the application layer and can establish IRC chat protocol to further hide their activities (Barakat & Khattab, 2010). Establishing IRC chat protocols and using open source P2P protocols can put organizations end-points and networks at great risk of being taken over by botnets and super-botnets (Barakat & Khattab, 2010). Additionally, sophisticated botnets can also implement encryption and/or digitally sign instructions to make it almost impossible for

network security and defensive operations to detect and stop a botnet attack (Barakat & Khattab, 2010).

Backdoors

Landwehr et al. (1994) provide some additional definitions of security flaws that then could be used as an attack vector. The researchers defined security flaws as "any conditions or circumstances that can result in denial of service, unauthorized disclosure, unauthorized destruction of data, or unauthorized modification of data" (Landwehr et al., 1994). These researchers developed a taxonomy for security flaws and detailed over 50 actual security flaws. The goal for developing such a taxonomy was to organize or classify computer security flaws by type in order to prevent the exploitation of unintended security flaws and purposeful misuse of flaws to compromise computer systems (Landwehr et al., 1994). This research was similar to the Research in Secured Operating Systems (RISOS) project and Protection Analysis project conducted by Information Sciences Institute of the University of Southern California (Landwehr et al., 1994). Moreover, the researchers wanted to bring attention to the inherent security flaws in software and the consequence of these security flaws in operating systems (Landwehr et al., 1994).

Exploits

In 2000, McGraw and Morrisett published a paper that presented the growing malware problem. This paper detailed a historical perspective of malware and the various approaches to detect and remove malicious files (McGraw & Morrisett, 2000). McGraw and Morrisett (from Cornell University) chaired a group of over twenty

malware researchers from across the world to elevate a world-wide malware discussion and a call to action (McGraw & Morrisett, 2000). The groups discussed such issues as the increasing complexity of computer systems and the networks that deliver connectivity. The group also discussed the ease computer extensibility and the susceptibility of these “networked” computer systems to be attacked. The group also concluded that any networked computing system is susceptible to malware or hostile code. However, the researchers also pointed out that an ever-present network, like the internet, provide attack vectors with ease (McGraw & Morrisett, 2000). The group also came to the realization that attackers no longer have to gain physical access to computer systems to propagate attacks. Networks, not physical access, become the highway to drive malware attacks (McGraw & Morrisett, 2000). Networks combined with rising system complexity provide more avenues for attack and complex systems make it easier to hide or mask malicious code (McGraw & Morrisett, 2000).

Ransomware

Ransomware has become quite popular for cyber-criminals over the past few years. This type of malware has grown by some accounts by 500% each year since 2013 (Symantec, 2014). Ransomware has been classified as a Trojan variant or virus depending upon the particular malware variant infecting the system (Khakhutskyy, 2016). Although ransomware is thought to be fairly recent, ransomware has been around for nearly thirty years (Hampton & Baig, 2015). Ransomware is used by cyber criminals infect and encrypt data/files such that access to this information is only granted once ransom or financial arrangements have been made to provide the “key” to unlock these files (Gazet, 2010). However, these types of attacks are not new. The PC CYBORG

(AIDS) as an example was delivered to many computers via a floppy disk in 1989 (Hampton & Baig, 2015). The PC CYBORG trojan then encrypted files and instructed users via a socially engineered message to pay a license fee with a \$189 check to a company in Panama (Hampton & Baig, 2015). Much of the 1990's ransomware was driven by amateur hackers in order to test and demonstrate technical capability (Hampton & Baig, 2015).

Ransomware began to rise in the early 2000's as malware developers started to sell their "bot-nets" (Bechtel, 2014). Malware developers started to see a market for their "bot-nets" and began to profit from direct information theft and advertising revenue (Bechtel, 2014). The early 2000's also saw an increase in theft of banking credentials or sensitive passwords (Condon, 2012). By the late 2000's, malware developers started to work together and share compromised assets for sale to the highest bidder (Hampton & Baig, 2015). It was these networks of "bots" that enabled cyber criminals to launch large-scale attacks on organizations (Condon, 2012). It was about this time that nation state organizations also started to realize the benefit of such networks (Carlson, Davis, & Leach, 2014). Cyber criminals were now positioned to offer cyber-attacks to steal intellectual property, run sophisticated phishing campaigns and propagate networks further into targeted organizations (Hampton & Baig, 2015).

Until 2011, ransomware attacks had been isolated and unsophisticated (Condon, 2012). In late 2011, ransomware began to directly attack end-users in mass (Hampton & Baig, 2015). By 2012, ransomware launched a major cyber-attack attacking the more connected internet user base and prey on the hype of computer viruses (Hampton & Baig, 2015). In 2012, ransomware developers launched large-scale end-user attacks with "Fake

Anti-virus” (Kharraz et al., 2015). These fake attacks tricked end-users into believing they had been infected by a serious virus and needed to pay for non-existent AV software to remediate the remedy and restore the compromised system (Krebs, 2012). Information security companies and researchers alerted the public to the scam shortly after the attack, However, due to the sheer number of these attacks, the credit card companies responded to payments made to the “Fake AV” and this first ransomware attack was virtually shut down overnight (Krebs, 2012).

More sophisticated ransomware attacks began to take place around 2005. Malware known as “lockers” began to emerge and stage denial of service attacks on infected systems (Young & Yung, 2016). These early “lockers” attacked boot operations and would not allow the machine to initialize until the “ransom” request was paid (Young & Yung, 2016). Early “lockers” did not attack our touch the file system and remained in memory after boot-up. Most security companies responded to these “locker” attacks by extending Anti-Virus software to remove malicious software (Young & Yung, 2016). More advance “locker” attacks were seen in late 2005 with the PGPCoder/GPCode locker. The PGPCoder/GPCode encryption locker was the first instance of ransomware where files and content was encrypted and released for payment (Young & Yung, 2016). Malware attackers released various versions of GPCode and many of these versions were released were greatly flawed. The flawed GPCode had issues poorly implemented encryption routines, breakable encryption keys and recovery of deleted content (Hampton & Baig, 2015). Malware developers began to strengthen GPCode over time and improve the many issues with encryption and encryption keys (Young & Yung, 2016). The GPCode malware was not the only limiting factor for ransomware. Directly attacking

end-users required many points of contact including the end-user, end-user payment gateways, cyber-attacker payment gateway and ultimately the malware developer (Hampton & Baig, 2015). This payment of ransom process was complex and risky. The many points of contact could slow the pace of payment and encumber the extortion process by allowing law enforcement to intervene to stop or track payments (Hampton & Baig, 2015).

In 2013, ransomware met with the perfect storm for delivering and executing ransomware around the world. The perfect storm was represented by three components: 1) strong and unbreakable encryption technology, 2) anonymous delivery and exchange of encryption keys and 3) untraceable methods to execute ransom payments (Hampton & Baig, 2015). CTB-Locker was the first ransom malware to take advantage of the perfect storm. CTB stands for “Curve, TOR, and Bitcoin” (Hampton & Baig, 2015). The “curve” represented the elliptic curve cryptography implemented to encrypt the targeted files/content. The Onion Routing (TOR) protocol enabled anonymous communication for key exchange. Bitcoin provided secure and untraceable crypto-currency transactions in order pay ransom. The CTB-Locker model still provides the means for new generation of ransom malware to deny access, secure key exchange and make untraceable payments (Young & Yung, 2016). According the Symantec ransomware grew at 500% in 2013 likely due to the perfect storm (Symantec, 2014). Large, medium and small organizations were hit with the CTB-Locker ransomware. However, large corporate organizations had backup solutions that offer protection for a number of threats. Many medium and small organizations did not and still do not have the financial or technical resources to develop the necessary backup solutions to address these threats.

Today's malware researchers offer that the ransomware can be easily defeated with collaboration and sharing of malware (Kharraz et al., 2015). However, many information security experts suggest that history should be a lesson for improving malware over time (Kharraz et al., 2015).

Ransomware continues to own the media and prey on the public's fear of losing access to their information (Hampton & Baig, 2015). The security industry and academic community is always playing catchup with advanced malware threats such as ransomware (Wüchner et al., 2014). Currently, ransomware has the technical and financial model for much larger payoffs (Kharraz et al., 2015). It is imperative that ransomware be monitored and analyzed for new and improved releases (Hampton & Baig, 2015). The ransomware financial model is simply too lucrative to ignore. At some point, ransomware will migrate and target large corporate networks such as banks, hospitals and critical infrastructure. Sooner rather than later, large enterprise organizations will have address the threat of ransomware within their operational environment.

Advanced Persistent Threats (APTs)

Advanced Persistent Threat (APT) is a term used to describe a new type of malware attack (Tankard, 2011). The term Advanced Persistent Threat (APT) was originally identified as a specific type of malware by the United States Air Force in 2006 (Rekdal & Bloemerus, 2013). APT's have been described as well-funded, technically advanced and well-organized with financial motivations (Rekdal & Bloemerus, 2013). APT attacks are typically unique and utilize multiple attack vectors to gain access to networks (Rekdal & Bloemerus, 2013). APTs employ advanced techniques to avoid

detection and remain on infected systems for long periods of time before activation (Rekdal & Bloemerus, 2013). In 2011, high profile APT attacks gained notoriety by attacking some of the who's who for Government and technology organizations (Rekdal & Bloemerus, 2013). Commercial organizations such as Sony, RSA Security, Lockheed Martin, Citigroup, Fox Broadcasting and Public Broadcasting Service (PBS) were faced with new types of attacks (Nicho & Khan, 2014). United States Government organizations such as the National Aeronautics and Space Agency (NASA), Federal Bureau of Investigation (FBI) and Department of Treasury were also targeted (Nicho & Khan, 2014). Europe also faced similar struggles with this new type of malware. European organizations such as the European Space Agency, the British and French treasuries were also targeted by APTs (Nicho & Khan, 2014).

APT malware has quickly become a major issue for information security and leaders around the world (Molok, Ahmad, & Chang, 2012). APTs employ stealthy techniques to breach networks and establish long term surveillance within the network (Molok et al., 2012). Malware developers for APTs are concerned with breaching systems and gaining unauthorized access to systems over time (Nicho & Khan, 2014). The first phase for an APT is typically breaching and evading detection for extended periods of time (Molok et al., 2012). The second phase for APT's is to gather intelligence and perform reconnaissance within the breached network (Molok et al., 2012). APT's intelligence gathering and reconnaissance is sophisticated and literature suggests very targeted to the organization (Symantec, 2014). During the intelligence gathering phase, APTs try to gain insight into operational aspects of the organization such as information assets, business functions, approval authorities and "normal"

communications (Molok et al., 2012). After the intelligence has been gathered, APTs begin to capture the necessary credentials, gain access or escalate privileges in order to achieve the target objective (Rekdal & Bloemerus, 2013). APT attackers tend to target large organizations such as financial services, government and defense contractors (Symantec, 2014). In 2010, Google, Adobe and other large U.S. organizations were reportedly breached by Chinese APT attacks that stole intellectual property, email accounts, and other organizational information (Panda Security, 2014). The APTs then sent the stolen information to Taiwanese IP addresses (Panda Security, 2014). APT's remain a powerful malware and represent a major threat to businesses and Governments. However, APT's remain the least studied and the least understood of all malware (Nichols & Khan, 2014). It is understood that APT's adapt constantly and take advantage of polymorphic techniques to evade detection for long periods of time (Kaur, 2014). Several studies and authors suggest that the sophistication and targeting of APT malware may represent the next wave of military conflict (Dunlap, 2011).

Analyzing and Detecting Malware

In 1995, Lo et al. developed detection tools for various types of malware including computer viruses, worms, Trojans, and logic bombs (Lo, Levitt, & Olsson, 1995). This research was one of the first detection approaches that included Static Analysis for the malware in question (Lo et al., 1995). The detection method used was called the Malicious Code Filter (MCF). MCF was used to detect malicious code and security related vulnerabilities in software (Lo et al., 1995). MCF could be used to perform off-line analysis to determine indications of compromise. The researchers proposed MCF to slice the program into small functions or "code pieces" in order to

analyze the overall program. Each sub-function could be analyzed to determine the maliciousness. Program slicing techniques were used to evaluate and collect the program properties (Lo et al., 1995). This approach enabled researchers to evaluate a number of sub-functions of a larger program to determine overall malicious behavior. The researchers also proposed an approach to defeat “program slicing” and the potential countermeasures needed to maintain detection rates (Lo et al., 1995).

In 1998, Lee and Stolfo, working at Columbia University developed a general and systematic methodology for Intrusion Detection using data mining techniques. The researchers leveraged pattern recognition and machine learning techniques in order to model program execution properties and user behavior (W. Lee & Stolfo, 1998). The techniques employed were able to analyze system call data with network tcpdump data in order to detect potential anomalies from both programs and user behavior (W. Lee & Stolfo, 1998). The researchers were able to construct concise and accurate classifiers by using the association rules algorithm and the frequent episodes algorithm (W. Lee & Stolfo, 1998). These two algorithms were used to compare both intra-and inter-audit record patterns (W. Lee & Stolfo, 1998). Using this approach, researchers could analyze both standard and abnormal program or user behaviors. The discovered patterns then could be used to facilitate additional feature selection to improve detection rates.

In 2001, Wagner and Dean developed static and behavioral analysis methods for host-based intrusion detection. The researchers working at University of California Berkeley developed methods to examine program specifications and behaviors exhibited at time of execution (Wagner & Dean, 2001). Static analysis for the first time was being used for intrusion detection and the researchers recognized that attacks should have

atypical behavioral characteristics (Wagner & Dean, 2001). Wagner & Dean developed a specification for a program based upon the results from static analysis. Secondly, the authors used execution monitoring to determine whether the program executed as expected. The primary challenge for the researchers was to develop an intrusion detection technique with low false positive rates (Wagner & Dean, 2001). The researchers were able to develop a static analysis and execution monitoring approach to look for unexpected execution of functions. This approach was able to demonstrate positive results for host-based intrusion detection. The authors highlight three distinct advantages for analysis using both static and dynamic analysis: 1) achieves a high degree of automation, 2) provides protection against a broad class of malware attacks based on corrupted code, and 3) limited false positives or “false alarms” (Wagner & Dean, 2001).

In 2003, Linn and Debray working at the University of Arizona, researched techniques used to obfuscate executable code to avoid static disassembly. Static analysis provides the ability to expose machine code into human readable functions that then can be used to reverse engineer software executables (Linn & Debray, 2003). Static analysis, while helpful in malware research, creates deep insight into the logic of executables and the library of functions used by the software by detailing the step-functions as the code executes. This insight provides malware analysts the ability to look into the executable and look for malicious functions as they execute. Static analysis also offers the opportunity for others to reverse engineer software and steal intellectual property from legitimate software providers (Linn & Debray, 2003). As part of the reverse engineering, executable disassembly provides a translation from machine code to assembly code (Linn & Debray, 2003). Code and function obfuscation techniques are

used by software developers in order to disrupt reverse engineering and program disassembly by making programs harder to disassemble correctly (Linn & Debray, 2003). Linn and Debray concluded the paper with a discussion on two static disassembly algorithms that break obfuscation and various techniques used to impede these algorithms.

In 2005, Gheorghescu, working at Microsoft Corporation, performed research that would automate malware classification. The research conducted provided advanced malware classification methods to aid in the detection of malware. Classification and naming of viruses is helpful for sharing discovered malware. However, the anti-virus industry did not adhere to a standard naming convention causing issues with processing new malware samples (Gheorghescu, 2005). Standard naming conventions could significantly speed the determination of files being malware or benign. The researchers introduced an innovative classification system for desktop computers. The classification compared new and unknown samples with known database of malware within minutes. This approach would also track samples based on evolution of the malware sample (Gheorghescu, 2005). Gheorghescu's approach used three matching algorithms to process malware samples and based on the results made determinations of good or bad. The research also presented methods for malware-handling tasks including sample clustering, outbreak detection, automatic virus naming, and phylogeny tree (Gheorghescu, 2005).

In 2006, Baecher et.al., working at the University of Mannheim, developed and presented a platform for processing a large-scale collection of self-replicating malware collected in the wild (Baecher, Koetter, Holz, Dornseif, & Freling, 2006). At this point in

time, there was little empirical data (quantitative or qualitative) to describe self-replicating malware. The inability to harness empirical data hampered many counter-measures for malware including network-based and host-based intrusion detection (Baecher et al., 2006). The Nepenthes platform, provided an emulation platform to expose and capture attack data. The Nepenthes platform also provided a means to capture empirical data about self-replicating malware and a means to analyze thousands of samples of previously unknown malware (Baecher et al., 2006). The data collected by the Nepenthes platform provided the empirical data capture to vendors such as host-based IDS/anti-virus systems (Baecher et al., 2006).

Kolter and Maloof (2006), describe the use of machine learning and data mining to detect and classify malicious executables found in the wild. The researchers, from Stanford University and Georgetown, gathered nearly 2000 benign and 1,700 malicious executables to perform advanced classification research. The researchers extracted more than 255 million distinct n-grams from these two sample sets. N-grams are used to extract substrings of a file for a fixed length n. These n-grams can be efficiently collected and analyzed for signatures. The researchers processed and extracted feature for 3,700 files (Kolter & Maloof, 2006). The researchers evaluated a variety of inductive methods, including naive Bayes, decision trees, support vector machines, and boosting to process n-gram samples (Kolter & Maloof, 2006). The researchers found that boosted decision trees outperformed other algorithms with an area under the receiver operating characteristic (ROC) curve of 0.996. The researchers proposed an analysis methodology that should scale to larger collections of samples. The conducted studies examined three key parameters: the size of n-grams, the size of words and the number of selected features

(Kolter & Maloof, 2006). Due to limited computational resources the study was unable to evaluate exhaustively all methods for all settings. However, once the researchers applied detectors to 291 malicious executables the true-positive rate of 98% and a desired false-positive rate of less than 5%. This was an important finding as the methodology suggested that the approach could be used in operational systems for detecting unknown malicious samples (Kolter & Maloof, 2006).

In 2006, Lee and Mody presented methods for systems to automatically classify malware into families or categories. Their approach monitored runtime behavior of applications and captured the series of functions as they were executed (T. Lee & Mody, 2006). Based upon these environmental parameters the system was able to categorize applications. The system accurately classified malware based on execution properties and behavioral characteristics operating in a Microsoft-based computer system (T. Lee & Mody, 2006). The researchers constructed a knowledge base of application groups by sampling a large population of applications. Based upon the set of known functions and behaviors exhibited by prior classifications, the system was able to classify new applications into known application groups and render a verdict of whether the application was malware (T. Lee & Mody, 2006).

In 2007, Bilar, from Wellesley College (Massachusetts), presented a paper to discuss detection mechanisms for malicious code through statistical analysis of operation code (opcode) distributions (Bilar, 2007). The researcher analyzed, disassembled and performed opcode frequency distribution for 67 malware executables (Bilar, 2007). These results were compared to those of 20 non-malicious samples. Bilar (2007) found that the malware opcode distributions did significantly differ from that of non-malicious

samples. In addition, unique opcodes seem to be a stronger predictor as many of the malware samples had unique frequency distribution patterns (Bilar, 2007). The researchers found that sixty opcodes accounted for 99.8% of opcodes found in both malicious and benign samples. However, 14 malicious opcodes accounted for 92% of the total extracted opcodes and the top 5 malicious opcodes accounted for 65% of the extracted opcodes. This research would be used as the basis for using machine learning for detection of malware.

In 2007, Martignoni, Christodorescu and Jha discussed the growing threat of malware due to the sophistication of malware evading signature-based detection (Martignoni, Christodorescu, & Jha, 2007). The malware developers can easily evade detection by "packing" the malicious payload in layers of compression or encryption (Martignoni et al., 2007). The researchers describe state-of-the-art malware detection using both static and dynamic techniques to uncover the packed malware. These techniques are slow due to off-line nature of the analysis and prove to be highly ineffective due to the metamorphic nature of the malware (Martignoni et al., 2007). The researchers proposed a new technique known as OmniUnpack. The OmniUnpack approach closely monitors the execution of a program in real-time and detects the "unpacking" when the program has removed the various layers of packing (Martignoni et al., 2007). OmniUnpack improved "packed" malware detection by analyzing the unpacked malicious payload during runtime execution. Experimental results demonstrated quite effective detection results with low overhead.

In 2008, Ye, Wang, Li, Ye, & Jiang discussed the failure of the traditional signature-based anti-virus systems to detect polymorphic/metamorphic malware (Ye et

al., 2008). The researchers also discussed that the traditional signature-based anti-virus systems were also blind to new and previously unseen malicious executables (Ye et al., 2008). The researchers discussed that data mining techniques are needed due to large and growing collections of malware (Ye et al., 2008). The research presented involved analyzing and developing profiles for Windows APIs called by PE files. The researchers discuss the results of their Intelligent Malware Detection System (IMDS) using Objective-Oriented Association (OOA) mining based classification (Ye et al., 2008). IMDS consisted of three major modules: 1) PE parser, 2) OOA rule generator, and 3) rule based classifier. This study outperformed commercial software products such as Norton Anti-Virus and McAfee VirusScan (Ye et al., 2008). This approach also outperformed previous data mining based detection systems using Naive Bayes, Support Vector Machine (SVM) and Decision Tree techniques (Ye et al., 2008). The study demonstrated a solid approach for polymorphic and metamorphic malware detection.

In 2009, Rieck et al. present issues with malware variants (polymorphism) and the use of obfuscation to hinder detection at the file level (Rieck, Trinius, Willems, & Holz, 2009). The researchers acknowledge that the amount and diversity of malware variants render classic security defenses like anti-virus ineffective (Rieck et al., 2009). The research suggested that the sheer volume of attacks from malware including viruses, worms and Trojans make detection more difficult by the day (Rieck et al., 2009). The volume combined with obfuscation and polymorphism techniques require a new approach for detection. The researchers believe that a machine learning approach was needed to solve the polymorphic malware growing problem. The research presented offers a framework for automatically identifying novel classes of malware with similar

behavior (clustering) and assigning unknown malware to these discovered classes (classification) (Rieck et al., 2009). These techniques, clustering and classification, allowed for improved detection and an environment to process thousands of malware binaries. Further, clustering and classification provided improved discovery of novel malware variants.

In 2009, Bayer et.al., working on conjunction with University of California Santa Barbara and University of Vienna, developed automated environment to analyze malware samples in a controlled environment (U. Bayer, Milani-Comparetti, Hlauscheck, Kruegel, & Kirda, 2009). This automated environment would also produce reports that detailed the program's actions during execution. These details were then used to analyze both benign and malware samples. The researchers then explored clustering techniques to identify samples based on execution behavior. The researchers admitted that previous clustering techniques did not scale and failed to generalize the observed behavior (U. Bayer et al., 2009). The new approach proposed a scalable clustering approach to identify and classify malware samples. The researchers performed dynamic analysis to capture execution traces for malware programs. Profiles for malware were generated using these execution traces in a generalized manner (U. Bayer et al., 2009). These profiles were then used as input into a scalable clustering algorithm capable of handling large datasets.

Dai, Guha and Lee (2009) provide unique insight into malware classification by analyzing frequency distribution or unique call sequences for various types of malware. The researchers approach malware detection through extracting dynamic instruction sequences from malware through data mining techniques (Dai et al., 2009). The researchers extracted runtime instruction sequences from unknown executables and

organized instruction sequences into basic blocks in order to evaluate malware (Dai et al., 2009). The extraction techniques used were based on certain instruction sequence patterns based on instruction associations with derived basic blocks (Dai et al., 2009). The researchers used a data mining processes to perform feature extraction, feature selection and to build the classification model (Dai et al., 2009). This approach yielded accurate, reliable and efficient predictive classification model for malware detection.

In 2010, Devesa et al. presented that a significant security threat exists due to the exponential growth in malware (viruses, Trojans or worms) (Devesa et al., 2010). The researchers define malware as any kind of program explicitly designed to harm or disrupt computer system operations (Devesa et al., 2010). In order to mitigate the malware problem, all incoming code must be analyzed to classify these files as malware or benign software. The most common approach is to combine static and dynamic analysis techniques in order to extract execution properties for the unknown files (Devesa et al., 2010). However, due to the escalation in polymorphic and metamorphic malware attacks the manually analyzing thousands of suspicious files each day would be futile (Devesa et al., 2010). The researchers proposed an emulation environment for testing that provided secure and safe execution of suspicious code. The environment properties to classify samples with several machine-learning algorithms (Devesa et al., 2010). The study tested the proposed system real malware samples. The initial results from the study reported reliable results with high performance for the malware sample set.

In 2010, Paulevé et al. working in France, applied Locality Sensitive Hashing (LSH) to dramatically improve performance for processing pattern recognition to be applied to malware detection (Paulevé, Jégou, & Amsaleg, 2010). The researchers

viewed traditional search schemes as computationally expensive and poorly fitting for real data sets (Paulevé et al., 2010). Although several extensions had been proposed to address the limitations, there had not been a comprehensive review of the algorithms or recommendations for “real world data”. The study conducted a comparison of several families of space hashing functions in a real world environment (Paulevé et al., 2010). The comparison included random projections, lattice quantizers, k-means and hierarchical k-means. The finding demonstrated that the unstructured quantizer significantly improved the accuracy of LSH (Paulevé et al., 2010). The study also discussed the previous findings concerning LSH, merits and limitations to the proposed LSH approach.

In 2011, Bailey et al. presented a study to highlight the ineffectiveness of Anti-Virus (AV) commercial products. The researchers used a large collection of malware that spanned a variety of attack vectors (e.g., spyware, worms, spam) to demonstrate the ineffectiveness of AV in an operational environment (Bailey, Oberheide, & Andersen, 2011). Further, the study showed that AV products mischaracterized malware and failed in large part to semantically define malware or classes of malware. The authors proposed a new classification technique that describes malware behavior in terms of system state changes (e.g., files written, processes created) rather than in sequences or patterns of system calls (Bailey et al., 2011). The researchers developed a methodology to automatically categorize malware into classes based on behaviors (Bailey et al., 2011). The study also demonstrated that behavior-based clustering provided a more direct and effective way to classify malware (Bailey et al., 2011).

Alam, Horspool and Traore (2013) extended malware research by examining dynamic binary obfuscation. Dynamic binary obfuscation is used by metamorphic malware to generate a new sequence of opcodes in the memory in order to avoid detection (Alam et al., 2013). Polymorphic and metamorphic malware is very difficult to analyze manually because of the sophistication of such malware (Alam et al., 2013). The researchers developed an automated tool and language named MAIL (Malware Analysis Intermediate Language) to automate and optimize the analysis and detection process (Alam et al., 2013). MAIL provided an element of portability for building metamorphic malware analysis and detection due to the use of annotated control flow graphing used for pattern matching (Alam et al., 2013). The study yielded detection rate for metamorphic malware of 93.92% and a low false positive rate of 3.02% (Alam et al., 2013). This study also discussed the re-use of functions from previous successful malware exploits. The researchers also discussed the use of embedded compilers and obfuscation by metamorphic malware to evade detection (Alam, et al., 2013).

In 2014, Tamersoy and Roundy, presented research for detecting both polymorphic and metamorphic malware. Their researchers stressed the need for new detection methods due to the increasing sophistication of malicious software (Tamersoy et al., 2014). The researchers call for new defensive techniques that are scalable and agile to malware. The researchers propose a new algorithm that identifies malicious executable files by applying reputational data with the file (Tamersoy et al., 2014). The researchers constructed a large dataset from Norton Community Watch and performed analysis for non-infected/infected machines and benign/malicious files resident on those machines. The researchers then identified relationships that exist between machine and

malware (Tamersoy et al., 2014). This algorithm “Aesop” leverages locality-sensitive hashing to measure the strength of these inter-file relationships (Tamersoy et al., 2014). These relationships were used to construct a bi-partite graph to infer benign or malicious (Tamersoy et al., 2014). The study produced impressive results with a 99.61% true positive rate and a 0.01% false positive rate (Tamersoy et al., 2014).

In 2015, Mohaisen and Alrawi presented an approach to automatically detect metamorphic and polymorphic malware based on behavior exhibited during analysis. Automated Malware (AMAL) provides both analysis and labeling (classification and clustering) to improve malware detection (Mohaisen & Alrawi, 2015). AMAL consists of two sub-systems, AutoMal and MaLabel. AutoMal provides tools to collect detailed behavioral artifacts of file, memory, network and registry during execution (Mohaisen & Alrawi, 2015). MaLabel uses the data provided by AutoMal to train and build classifiers. AutoMal can be used with unsupervised learning by leveraging multiple clustering algorithms for sample grouping (Mohaisen & Alrawi, 2015). The research presented achieved a precision rate of 99.6%. The researchers also discussed several benchmarks, costs estimates and measurements highlight and support the merits and features of AMAL.

The foundational studies presented above were intended to provide a historical perspective and context for malware research. Advanced detection studies are provided below. Advanced detection studies provide deeper insight into designing, conducting and evaluating malware experimental research. Additional details regarding advanced detection studies referenced below are presented in Appendix A.

Research Methods

Malware research is typically conducted using quantitative research methods (Rossow et al., 2012). Quantitative studies have been used to observe and collect datasets for a wide array of experiments (Creswell, 2007). In many cases, quantitative malware experiments have been designed to improve detection models, study longitudinal behaviors and validate prior malware research (Rossow et al., 2012). The study was aligned with Rossow's (2012) three basic concerns for conducting malware research: 1) following a research methodology and being transparent with the study details, 2) developing an experimental framework that uses the correct and representative data and 3) the experiment must take due consideration to not harm the research, resources or contaminate findings. Each of these concerns were addressed and discussed in detail later in the study's approach.

The experimental design and approach enabled for this research to advance the study in stages. The staged approach enabled the research to conduct experiments and validate results at various checkpoints. This approach provided flexibility for the researcher and allowed the feature set and clustering algorithms to be utilized and adjusted over time. The developed prototype environment was able to discover polymorphic malware with multiple datasets – training datasets and experimental datasets. The developed prototype environment was designed to inspect and determine the malicious or benign attributes of files and compare results. The focus of this study was to identify and detect Windows-based polymorphic malware. Windows has become primary target for malware attacks and represents the largest population of end-points at the enterprise level. An illustration of the proposed environment is provided below.

This malware research followed the single-subject experimental approach outlined by Neuman and McCormick (1995). Single-subject experimental design has evolved over time to facilitate researchers quest to answer different types of questions under varying experimental circumstances (Neuman & McCormick, 1995). The most common single-subject experimental designs include are reversal designs, multiple-baseline designs and multi-element designs (Neuman & McCormick, 1995). The purpose of using the single-subject design for this experiment was to develop a protocol that worked nicely with the malware research subject area. It was believed that malware research closely resembles that of medical research and thus can be used to evaluate incremental treatments and measure results for drug treatment and patient care protocols. Single-subject experimental design was used for this study as it paralleled baseline testing with incremental treatment testing. This study achieved the desired goal to improve malware detection given multidimensional topological data extracted from static analysis, dynamic analysis and file property. This study further improved detection rates by adding weighted multidimensional features from static analysis, dynamic analysis and file properties to evaluate detection rates. This study was designed to answer two basic research questions:

1) Can detection rates be improved by increasing the quality and quantity of multidimensional features for the machine learning advanced clustering algorithms from file properties, static and dynamic analysis?

2) Which of the machine learning advanced clustering algorithms performed better given the multidimensional features from file properties, static and dynamic analysis?

The single-subject design provided a standard and widely accepted protocol to answer the research questions posed for this study. The study followed the basic single-subject protocol throughout the study:

1. Establish baseline data – established a baseline for detection given a standard feature dataset with a standard clustering algorithm for polymorphic malware through multiple measurements before an intervention. Currently the detection rates for polymorphic detection rates using advanced classifiers range from 68.75% to 81.25% (Amos et al., 2013). Three datasets were generated that allowed detection rates to exceed the baseline detection rates of 68.75% to 81.25%.

2. Manipulate feature set – the standard feature dataset was augmented to evaluate whether detection rates are improved after the intervention of features. A number of studies examined dynamic analysis (Rieck, Trinius, Willems, & Holz, 2011), static analysis (Kerchen et al., 1990) and file properties (Subramanya & Lakshminarasimhan, 2001). The study examined three multidimensional datasets. The three datasets were generated by combining known malware, known benign and unknown samples into single feature dataset. Once the baseline was generated, the feature weighting was manipulated to see if detection rates would improve (Siddiqui, 2008).

3. Controlled procedures and environment – the developed prototype environment and datasets were controlled to ensure the extracted feature dataset remained static over time and the automated feature extraction process remained unaltered (Rossow et al., 2012).

4. Standard measurement approaches – a standard measurement approach was completed after performing baseline detection measurements for known (malware and benign) and unknown samples using multidimensional topological data with baseline clustering. The study established baseline detection rates for each of the three clustering algorithms: 1) Advanced Ensemble Classification (Bootstrap Aggregating (Meta Bagging (MB))), 2) Instance Based k-Nearest Neighbor (IBk) and 3) Deep Learning Multi-Layer Perceptron (DLMLP). Standard measurements included such measures as Accuracy (ACC) and the Correlation Coefficient (CC), True Positive Rate (sensitivity measure), and False Positive Rate (specificity measure). As specified by the standard protocol, these measurements were established as a permanent observational recording (Rossow et al., 2012).

5. Weighting of features – as part of this testing protocol the inputs were manipulated and assessed. Within a single dataset, features were weighted in a structured methodology, evaluated and assessed. The weighting of static features, dynamic to static features and file properties to dynamic to static features were assessed to evaluate improved detection rates for polymorphic malware.

6. Capturing testing results - All testing data were captured in terms of ACC, CC, True Positive Rate (TPR) and False Positive Rate (FPR).

7. Graphing results - All testing data were graphed and presented in terms of ACC, CC, True Positive Rate (TPR) and False Positive Rate (FPR).

8. Evaluating results – results from test data was collected after each test. Each test was analyzed with rigor to understand the ACC, CC, True Positive Rate (TPR) and

False Positive Rate (FPR). The test results from each test was evaluated and changes to inputs were documented to understand detection rate improvement. The test results were generated after each test and provision were made to ensure that the results were accurately reported.

9. Test controls - the specific design for this study required that test controls remain in place from one test to another such that the test data, interpretation of the data test and conclusions reached from testing were reliable and believable.

The experimental design and approach enabled the researcher to advance the study through a defined testing protocol. The testing protocol enabled the researcher to conduct baseline testing and perform experimental testing with the various clustering algorithms. The testing protocol also provided the researcher with a means to select, weight feature sets, conducting testing and document results for the various clustering algorithms. The testing protocol allowed the researcher to validate detection results in stages and evaluate effectiveness. The experimental research methodology was consistent with previous quantitative malware studies (Creswell, 2007). The study's methodology was constructed to improve detection models and validate test results prior to conducting additional research (Rossow et al., 2012). The research design utilized a widely accepted quantitative approach for conducting experimental studies similar to those conducted in healthcare, drug trials and other medical studies (Rizvi & Nock, 2008).

Gaps in Current Literature

In both the foundational research and advanced detection studies there exists a number of gaps for ongoing malware research. As the war against malware (Trojans,

viruses, worms, APT's, etc.) continues to escalate and spiral into unknown dimensions some perspective is needed to understand the threat landscape. It is important to understand and recognize that malware attacks are causing businesses and Governments to lose billions of dollars (McAfee Labs, 2015). Malware research is unfortunately not keeping pace with the adversary and the rapid escalation of polymorphic and metamorphic malware has continued evade today's detection techniques (Hansen, Mark, Larsen, Stevanovic, & Pedersen, 2016).

In reviewing nearly six-hundred (584) articles and studies, roughly one fifth (124) of those studies conducted advanced polymorphic malware research. However, polymorphic malware remains a major threat for ransomware and other types of malware (Kaur, 2014). More research is needed for polymorphic malware as the threat has become more advanced. Polymorphic malware is thought to represent at least half of the new malware released annually (Symantec, 2014). In addition, polymorphic malware growth in 2012 was 392% (Qu & Hughes, 2013) and greater than 500% in 2015 (Gostev et al., 2016). Due to the changing nature of polymorphic malware, additional malware research is needed to identify the evolving characteristics of the malware in order to provide better and different detection approaches. New techniques are needed to identify and classify the evolving polymorphic families of malware (Kaur, 2014).

Another research gap identified is the dataset or "samples" used for conducting malware research. There are many variations for collecting and testing malware datasets. Some of the major malware researchers from industry and academic continue to make samples available for consumption. Some of the major industry contributors include McAfee, Symantec, Virus Total, etc. There are also a number of unaffiliated sites that

offer malware for research as well. The unaffiliated sites include contagio, Virus Sign, VxHeaven and others. The majority of reviewed studies collected and used malware from these community of interest sites. In doing so, researchers leveraged known malware samples for detection purposes and experiments. Researchers were able to leverage these known malware datasets and validate new detection techniques with “live” malware samples. However, few studies mentioned the use of benign samples in order to validate detection. A majority (24 of 33) of the advanced malware detection studies referenced “labeled” test data – meaning that the study knows all the samples submitted. A minority of advanced malware detection studies used both known malware and known benign samples in order to evaluate and validate detection rates. However, in reviewing over thirty advanced malware studies not a single study evaluated a dataset with unknown samples. Several leading researchers suggest that using unknown samples would strengthen the study and offer additional validation for detection techniques (Hansen et al., 2016). It is well understood that assembling a malware dataset for research purposes is difficult. Finding additional benign and unknown samples makes the task of assembling a rich dataset much more difficult. As Hansen (2016) points out, there are significant benefits with constructing datasets that contains known malware, known benign and unknown samples.

Due to the use of polymorphic and metamorphic techniques utilized by malware developers, the number of unique malware files has exploded (Tamersoy et al., 2014). It is the sheer number of these unique files that render signature detection essentially useless (Fraley & Figueroa, 2016). However, there exists some number of unique features that can identify “families” of malware (Wüchner et al., 2014). The challenge

for improving malware detection is to find the right combinations that lead to better results (Wüchner et al., 2014). Malware researchers have been utilizing varying techniques to identify “clusters” of related malware families and use the proximity of the clusters for detection (Tamersoy et al., 2014). Several researchers have offered that the files can be clustered based on the result static or dynamic analysis (Hansen et al., 2016; Kaur, 2014; Qu & Hughes, 2013). Other researchers have demonstrated unique file properties and executable location may be used to detect malware (Mohaisen & Alrawi, 2015). Taking feature extraction outputs from both static and dynamic analysis would allow the construction of a rich dataset. In addition to static and dynamic analysis features, values such as file characteristics, environmental properties and file relationships could be used for additional analysis (Mohaisen & Alrawi, 2015). Further, advanced algorithms such as Meta Bagging (MB), Instance Based k-Nearest Neighbor (IBk) and Deep Learning Multi-Layer Perceptron (DLMLP) use proximity clustering algorithms to understand file-properties affinity and maliciousness (Yedidia, et al., 2003). Thus, if one file is known as malicious with certain features it could be associated with like files and features. Advanced clustering techniques such as General Belief Propagation, Loopy Belief Propagation, Dynamic Bayesian Networks, Hidden Markov Model (HMM), and Markov Random Field (MRF) could be examined and evaluated using the rich dataset offered by combining the results of static and dynamic analysis with file properties. These techniques have been examined by previous studies and should prove to be a solid approach for future study. The benefit of advanced clustering algorithms is that they may provide additional fidelity for analyzing feature clusters

In summary, there are a number of gaps in current literature. The first issue is there is not sufficient research for polymorphic malware. Polymorphic malware continues to grow at rates that far exceed the ability to industry and academia to keep pace. The second issue concerns the datasets used for research. The datasets should have representation from three all types of files: 1) known malware, 2) known benign and 3) unknown samples. Constructing dataset in this manner should yield more “real world” detection results. Lastly, many of the malware studies have utilized advanced algorithms based on either static or dynamic analysis. None of the studies reviewed combined the features of both types of analysis in order to improve detection. By combining the feature dataset and using advanced clustering algorithms should provide improved polymorphic malware detection.

Strengths and Weaknesses of Current Studies

The research conducted for previous studies include both foundational malware research (discussed in the previous section) as well as advanced detection studies in order to establish a baseline for this study. It was important to establish a baseline of topics and emerging malware research. It was also important to expand the understanding of contemporary experimental research leading to advanced malware detection. For the purpose of this discussion, the research referred to in this section is the Advanced Malware Detection Studies presented later in this chapter. In researching, reviewing and evaluating over thirty contemporary malware detection studies that span seven years (2010-2016), researchers presented unique malware detection and classification using a number of experimental approaches. The majority of the studies presented an experimental framework for conducting the malware research. A number of the studies

selected a sufficient sample size for conducting the research. Some of the better studies conducted malware research and thoroughly evaluated detection findings. These findings and results were analyzed to evaluate achieved detection rates in terms of True Positive and False Positive rates.

The thirty-three (33) studies presented in the Advanced Malware Detection Studies were reviewed and evaluated with the intent to leverage those positive aspects of the for this study. These studies were reviewed and evaluated based on a number of factors. The strengths of these studies include: 1) most of the studies (32 of 33) performed the study through an experimental framework, 2) almost all the studies (31 of 33) surveyed were focused on improving malware detection with “live” or real malware samples, 3) a majority of the studies (27 of 33) used dynamic analysis or static analysis to perform feature extraction for their study, 4) almost all of the studies (32 of 33) utilized advanced algorithms to aid or boost detection rates using features extracted from static or dynamic analysis and 5) most of the studies (30 of 33) analyzed detection results with a quantitative evaluation of study results. In most cases, the studies evaluated findings with several statistical measures to evaluate detection effectiveness. These criteria are typically used as statistical quality measures with machine learning. The statistical tests used to evaluate the performance various algorithms can be evaluated with Accuracy (ACC) and the Correlation Coefficient, True Positive Rate (sensitivity measure), and False Positive Rate (specificity measure).

Upon deeper inspection, the studies collectively also had some weaknesses or issues with conducting the research. The weaknesses for advanced malware studies include: 1) malware sample storage - there is little to no discussion regarding the

working with or securing “live” malware for the study or experiment, 2) sample size - the range of malware samples tested was very large - nearly 23 million (from a low of 90 samples to a high of 24 million samples), 3) feature selection - not a single study (0 of 33) combined the features from both static and dynamic analysis – many of the studies performed either static (14) or dynamic analysis (13), 4) malware classification or types of malware used for the experiment was not disclosed or the study only worked with known malware – there were no unknown samples to further test detection rates from a blind study perspective and 5) the use of training datasets for machine learning – there were varied and differing approaches to selecting, maintaining and using training datasets. The weaknesses outlined here should not discount any work conducted. However, addressing these issues can only help to improve future research.

Similar Research Methods

Researchers who choose to pursue quantitative research methods seek to understand complex models and make knowledgeable claims regarding the subject matter (Creswell, 2007). Researchers develop and apply strategies of inquiry or develop evaluation methodologies to analyze complex models or to better understand phenomena (Creswell, 2007). Strategies of inquiry or evaluation methodologies assist with the overall research approach by focusing on the experiments, quasi-experiments or correlational studies (Creswell, 2007). Quantitative research strategies design and execute complex experiments with many variables and specialize in the treatment of interdependent variables (Creswell, 2007). Quantitative studies seek to explain experiments through structural equation models that explain and prototype experimental outcomes (Creswell, 2007).

Malware research is typically conducted using quantitative research methods (Rossow et al., 2012). In the past, malware researchers have relied on quantitative experimental designs to understand how malicious code behaves during execution (Rossow et al., 2012). Quantitative studies have been used to observe and collect datasets for a wide array of malware experiments. In many cases, malware experiments were designed to improve detection models, study longitudinal behavior and validate prior research (Rossow et al., 2012). The previous advanced malware detection quantitative research methods have been reviewed and will be presented later in this section.

Malware research, similar to other types of research, needs to construct the appropriate framework for the experimental study. The research design must address a number of basic concerns. The first concern is the research methodology. According to Rossow (2012), the research methodology must be described sufficient detail in order to enable allow other researchers to reproduce the approach. In addition, the methodology must be transparent with the study's details in order to fully share findings (Rossow et al., 2012). Secondly, the experimental framework must have the correct and representative dataset. Researchers must be careful with selecting and using datasets such that bias is not introduced into the experiment (Rossow et al., 2012). Lastly, the experiment must take due consideration to not harm others (Rossow et al., 2012). Malware research must be very careful as these experiments are dealing with live samples, similar to medical virus research. An outbreak of malware can harm not only the experiment but also those resources around or supporting the research (Rossow et al., 2012).

The thirty-three (33) studies presented in the Advanced Malware Detection Studies section were reviewed to understand: 1) the research methodology, 2) the dataset and 3) the safeguards put in place in order to do no harm. Almost all the studies (32 of 33) performed the study through a quantitative experimental framework. The one study that did not follow an experimental framework conducted an observational study. Almost all studies (31 of 33) performed malware detection research with “live” or real malware samples. A majority of the studies (27 of 33) used either dynamic or static analysis as part of the experiment to extract further data for analysis. Almost all of the studies (32 of 33) utilized advanced algorithms to aid or boost detection rates. However, the majority of the studies did not provide sufficient details to replicate the study. Most of the previous studies (30 of 33) analyzed detection results. In most cases, the findings were evaluated with several statistical measures to evaluate detection effectiveness. Overall, the Advanced Malware Detection Studies provided the necessary framework and were fairly consistent with the experimental methodology and using representative datasets.

Advanced Malware Detection Studies

Over sixty (60) contemporary malware detection studies were surveyed that spanned seven years (2010-2016). These studies surveyed and reviewed to understand the unique detection and classification approaches presented by the researchers. In order to address current malware detection research for the proposed research were analyzed to understand:

- 1) research approach or framework for the study;
- 2) sample size – both labeled and unlabeled samples;

- 3) feature selection - based on static or dynamic analysis;
- 4) use of machine learning (supervised/unsupervised training) or advanced algorithms used to support detection;
- 5) evaluation of detection results.

After reviewing sixty studies, it became clear that number of studies had strengths and weaknesses. However, studies that had severe weakness needed to be eliminated from further review as they provided little or suspect value. Studies that did not provide a solid research approach, a reasonable sample size or set and an evaluation of detection results were eliminated from further review. In many cases, the evaluation 1) research approach, 2) sample set and 3) evaluation of results were pass/fail. Based upon the criteria above - twenty-seven (27) of the studies were eliminated from further review. Some of the studies were eliminated due to the fact that they lacked sufficient detail regarding the experimental research approach or how the experiment was conducted. Other studies were eliminated due to the fact that they were unclear or vague about the malware samples used for the experiment. Some of the eliminated studies had contradictory discussions regarding the experimental research and the results. These studies either did not specify or did not directly address the research approach for conducting the malware study. Studies were also eliminated due to the fact of how the malware detection results were either presented at a high level or not evaluated. In many cases, the studies eliminated presented results but did not quantitatively present an evaluation of the detection results achieved. Lastly, studies were eliminated because of the malware sampling or how the malware was collected for the study. Only the studies that clearly specified research framework, provided sufficient malware samples,

evaluated detection results and remained as part of the next phase of reviewing current research.

The remaining thirty-three (33) studies then reviewed and data extracted for evaluation purposes. These studies were reviewed and evaluated based on a number of factors.

1. Most of the studies (29 of 33) performed the study through an experimental research framework;
2. All remaining studies (33 of 33) surveyed were focused on improving malware detection with “live” malware samples;
3. A majority of the studies (27 of 33) used dynamic analysis or static analysis to perform feature extraction for their study;
4. Almost all of the studies (32 of 33) utilized advanced algorithms to help or boost detection rates and,
5. Most of the studies (30 of 33) analyzed detection results with a quantitative evaluation of study results.

The highlights for the remaining malware experimental studies are presented in Table 1 below.

Table 1 Advanced Detection Studies

Lead Author	Year	Study Type	Dataset	% Malware	Labeled Samples	Static Analysis	Dynamic Analysis
Park, Y. (2010)	2010	Experiment	200	55%	Yes	Yes	Yes
Shamili, A. S. (2010, August)	2010	Quantitative Experiments	897,922	33%	No	None	None
Sun, X. (2010)	2010	Experiment	90	67%	Yes	None	Yes
Wang, Z. H. (2010, May)	2010	Iterative Experiment	5,223	68%	Yes	Yes	None
Ye, Y. (2010)	2010	Experiment	50,000	70%	Yes	None	None
Caballero, J. (2011)	2011	Experiment	313,791	Not Specified	Yes	Yes	None
Jacob, G. (2011)	2011	Experiment	37,572	Not Specified	Yes	None	Yes
Jang, J. (2011)	2011	Experiment	20,000	100%	Yes	Yes	Yes
Kolbitsch, C. (2011)	2011	Experiment	Not Specified	Not Specified	Yes	Yes	Yes
Rosow, C. (2011)	2011	Experiment	100,000	85%	Yes	Yes	None
Zhang, J. (2011)	2011	Experiment	Not Specified	Not Specified	Yes	None	None
Chen, Y. (2012)	2012	Experiment	10,000	60%	Yes	Yes	None
Eskandari, M. (2012)	2012	Experiment	956	52%	Yes	Yes	Yes
Ghiasi, M. (2012)	2012	Experiment	1,211	68%	Yes	None	Yes
Zhuang, W. (2012)	2012	Experiment	75,000	78%	Yes	None	None
Borojerdi, H. R. (2013)	2013	Experiment	360	67%	Yes	None	Yes
Jha, S. (2013)	2013	Experiment	961	95%	Yes	None	Yes
Naval, S. (2013)	2013	Experiment	1,296	37%	Yes	Yes	None
Ponomarev, S. (2013)	2013	Experiment	1,544	54%	Yes	Yes	None
Tsuruta, H. (2013)	2013	Experiment	23,234,538	27%	Yes	Yes	None
Wang, H. T., Wei, T. E., & Lee, H. M. (2013)	2013	Experiment	3,000	96%	Yes	Yes	Yes
Xiao, X. (2013)	2013	Experiment	3,401	53%	Yes	None	Yes
Zolotukhin, M. (2013)	2013	Experiment	1,089	55%	Yes	Yes	None
Adebayo, O. S. (2014)	2014	Empirical Study	1,500	67%	Yes	Yes	None
Dornhaed, H. (2014)	2014	Observation	Not Specified	Not Specified	No	None	None
El Attar, A. (2014)	2014	Experiment	Not Specified	Not Specified	No	None	None
Elaziz, P. E. A. (2014)	2014	Experiment	80,077	Not Specified	No	None	None
Li, J. (2014)	2014	Experiment	Not Specified	Not Specified	No	None	None
Pramono, Y. W. T. (2014, August)	2014	Experiment	1,059,419	Not Specified	No	None	None
Pramono, Y. W. T. (2014, September)	2014	Experiment	Not Specified	Not Specified	No	None	None
Li, Y. H. (2015)	2015	Experiment				None	Yes
Hansen, S. S. (2016)	2016	Experiment	270,837	100%	Yes	None	Yes

Summary

Detecting polymorphic and metamorphic malware continues to be a challenge for the security community. A majority of the today's security research is focused on developing enhanced detection using techniques that collect, study, and mitigate malicious code (Kolbitsch et al., 2009). However, new polymorphic malware and the detection evading techniques render many of the current signature protections useless and therefore leave end-points unprotected (Rodríguez-Gómez et al., 2013). The speed at which polymorphic malware is advancing threatens enterprise computing and internet operations (Symantec Corporation, 2016). Being able to detect polymorphic, metamorphic and zero-day malware requires advanced detection techniques that provide rapid adaptation, scalability and produce low false positive rates (Borojerdi & Abadi, 2013). This study offers an attractive alternative for detecting polymorphic malware specifically targeting Windows operating systems. This study offers a single data set containing known malware, known benign and unknown samples, feature extraction methodology, a prototype environment and detection approach that far exceeded today's accepted baseline. The proposed methodology, experimental framework and experimental design is presented next.

Chapter 3

Methodology

Overview

The purpose of this study was to develop a quantitative experimental prototype using multidimensional topological data with machine learning using advanced clustering to provide improved detection for polymorphic malware. The study ultimately yielded a detection methodology for polymorphic malware with true positive detection rates well above the established baseline range of 68.75% to 81.25% (Amos et al., 2013).

Quantitative methods were chosen in order to provide objective measurements for malware detection (Babbie, 2010). The statistical, mathematical, or numerical analysis of the data collected through the proposed malware experiments were conducted using standard and accepted computational techniques (Babbie, 2010). Previous malware research highlighted the need for correct and representative use of the datasets, proper experimental methodology design such that there is sufficient transparency to enable reproducibility, and safely conducting experiments such that no harm is caused to others while performing the research (Rossow, 2013). This study leveraged past research for selecting supervised and unsupervised machine learning training and experiments (Kaur, 2014).

Malware research relies mainly on observing or analyzing malicious code (H. Yin, Song, Egele, Kruegel, & Kirda, 2007). Malware research methods must be cautious in designing and conducting experiments that observe and analyze the execution of malware within a real or virtual environment (Rossow, 2013). Malware experiments must carefully and thoughtfully select datasets, detection models, and research

methodologies in order to evaluate effectiveness (Hu, Wang, Li, Bai, & Jing, 2014).

Malware experiments must also study the longitudinal behavior and validate various aspects of the research (Rossow, 2013). The experimental methodology utilized was mindful of these requirements and carried out in a series of steps outlined below.

The experimental methodology had four components:

- 1) Secure dataset – The prototype protected the dataset (samples), data (feature extraction from samples), experiment (cluster analysis) and experimental resources with secure storage of malware, benign and unknown samples (over 2M total samples) (Rossow et al., 2012).
- 2) Automated malware feature extraction – the prototype system processed, analyzed, extracted and assembled multidimensional topological features from the sample datasets through an automated means using both static and dynamic analysis (Tamersoy et al., 2014),
- 3) Supervised and unsupervised machine learning – the developed prototype used multidimensional topological features and applied advanced clustering algorithms to demonstrate improved detection rates with training and extended datasets (Kaur, 2014), and
- 4) Results Validation - the developed prototype evaluated detection effectiveness using various accepted quantitative methods (Mohaisen & Alrawi, 2015).

This novel research developed a unique detection methodology using multidimensional topological features in a machine learning environment. The multidimensional topological features approach combined static and dynamic analysis in addition to unique file properties. Extensive research of over 500 studies over the last six

years has not produced a single study that leverages multidimensional topological features using static and dynamic analysis with file properties. In addition, the developed prototype produced repeatable feature extraction for various malware binaries such that new sample datasets could be analyzed over time. The study analyzed and evaluated results with the rigorous quantitative evaluation measures described by Mohaisen and Alrawi (2015).

Research Design

In this study, the researcher sought to understand the role of multidimensional topological features plays on improved malware detection. The study employed a quantitative experimental approach using the Single-Case Experimental Design (SCED). SCED has been chosen for this study as it enables the researcher to study small groups (in this case unique polymorphic malware) and generalize findings to a larger population (malware classes) (Neuman & McCormick, 1995). SCED provided a methodology for establishing a baseline and measuring interventions or changes over time (Neuman & McCormick, 1995). SCED was chosen because each polymorphic malware sample may be unique and can be treated as individual subject for research purposes. Therefore, the features extracted through static and dynamic analysis from the individual malware become criterion for further study. Instead of using control groups, Changing Criterion Design (CCD) was selected for this study as it allows the researcher to add or subtract features to better understand and manipulate detection rates based on adding/subtracting or weighting of features. SCED CCD was chosen as control groups are not be appropriate for this malware study. In drug and disease treatment protocol studies, control groups are commonly used to evaluate treatment over time. Detection is binary

and will not improve or decline over time. The SCED CCD allowed the researcher to introduce incremental changes in terms of multidimensional topological features and measure those changes in terms of detection rates throughout the study (Neuman & McCormick, 1995). Parallel studies in both the Educational and Clinical arenas use SCED research design with changing criterion design to understand behaviors, drugs and treatment for certain conditions (Neuman & McCormick, 1995). SCED CCD allows experimental controls to be demonstrated by a consistent shift in the rate of targeted outcome with each successive test or measurement (Neuman & McCormick, 1995). In theory, each achieved criterion of a behavior can function as the baseline for the next targeted criterion for successive studies (Neuman & McCormick, 1995). Given that each malware samples may represent a unique malware sample and unique features may improve detection the design is appropriate.

Research Procedures

The study randomly selected and assigned three test datasets of 200,000 samples. These datasets were constructed via a pseudo-random program that composed the datasets on a stratified sample basis (malware, benign and unknown). The test datasets developed maintained the distribution profile on a percentage basis: malware (40%), benign (30%) and unknown (30%). The datasets were then used to conduct baseline and further testing with the three clustering algorithms (MB, IBk and DLMLP). The study ran a series of baseline tests with unweighted multidimensional topological features to determine baseline detection rates. Baseline detection rates were documented for each clustering algorithm. The datasets were used as part of SCED CCD testing process for manipulating feature weighting for each clustering algorithm. After establishing the

baseline for each algorithm, incremental tests were conducted with weighted multidimensional topological features to evaluate improved or decreased detection rates.

The study established an effective detection rate for polymorphic malware detection ranges from 68.75% to 81.25% (Amos, Turner, & White, 2013). Each of the three clustering algorithms produced detection rates well above 81.25%. Additional experiments were conducted to tune and document the effects of feature weighting on detection rates. Additional tests were used to better understand the how weighting of multidimensional topological features improved or reduced the effective detection rate.

Baseline Testing. Initial baseline experimental testing was conducted for each clustering algorithm using unweighted multidimensional topological features. These features were derived through extracting static analysis, dynamic analysis and file properties features. Baseline testing enabled the researcher to establish training dataset for each clustering algorithm. Baseline testing supported further investigation of cause and effect to enable the researcher to develop manipulation or intervention strategies (Neuman & McCormick, 1995).

Weighted Occurrence Testing. The first set of experimental tests conducted utilized weighted multidimensional topological features derived from static analysis. These tests are similar to experiments presented by Bilar in 2007. Bilar (2007) performed a series of experiments with weighted features based on frequency of occurrence using static analysis various malware samples. Bilar (2007) used the weighted features in a series of Support Vector Machines (SVM) experiments. The study performed both weighted features and frequency of occurrence testing from static analysis using the three clustering algorithms (MB, IBk and DLMLP).

Dynamic-Static Occurrence Testing. The next set of experiments involved leveraging Bilar's weighted occurrence tests (derived from static analysis) with features derived from dynamic analysis. Polymorphic malware can encrypt functions and have embedded library calls which can be hidden from static analysis (Hansen, Mark, Larsen, Stevanovic, & Pedersen, 2016). Other studies have observed that malware developers have padded or introduced "junk" code to bypass static dynamic thus making the function call tracing very difficult (Jasiul, Śliwa, Gleba, & Szpyrka, 2014). Therefore, linking static and dynamic analysis may serve to enhance detection by linking dynamic occurrence of functions to those function calls observed in static analysis. This study extended Bilar's weighted feature study (2007) by linking opcodes observed in an actual execution environment (dynamic analysis) with those observed through static analysis. The weighting of features was directly correlated to frequency of occurrence in dynamic to static analysis. The experiments performed evaluated the detection results linking the static to dynamic features for all three advanced clustering algorithms (MB, IBk and DLMLP).

Extended Static-Dynamic Occurrence Testing. Polymorphic malware may call libraries or use functions available through the operating system such as Microsoft's dynamic link libraries (DLLs) (Egele, Scholte, Kirda, & Kruegel, 2012). The intended use or abuse of such functions by malware can be linked to file properties of the malware itself (Hansen et al., 2016). Therefore, experiments conducted during this phase, added various features from file properties to those of dynamic-static occurrence tests performed previously. Previous research has shown that there exists a number of "metadata" properties that can be used for malware detection (Kamongi, Kotikela, Kavi,

Gomathisankaran, & Singhal, 2013) and (A. Singh et al., 2012). This study found similar results. This study performed extended static to dynamic occurrence testing with all three advanced clustering algorithms (MB, IBk and DLMLP).

The research process evaluated and demonstrated effective detection rates for polymorphic malware. The detection rates for each algorithm exceeded the established ranges for polymorphic malware of 68.75% to 81.25% (Amos, Turner, & White, 2013). It was anticipated that each of the clustering algorithms with weighted features would deliver detection rates above the baseline test results. Additional weighting and experiments conducted did in fact deliver higher detection rates. The developed prototype environment to extract and assemble test datasets is presented next.

Prototype Environment

In order to consistently, accurately and safely extract multidimensional topological feature information, the study developed an integrated system to perform automated feature extraction for analysis for all datasets. This integrated system delivered the capability to extract features for file properties, static and dynamic analysis. The integrated system then constructed one dataset for each clustering algorithm. The feature extraction framework consisted of five modules: 1) a pre-process module that extracts and generates topological features based on static analysis of machine code and file characteristics, 2) a behavioral analysis module that extracts behavioral characteristics based on file execution (dynamic analysis), 3) a post-processing module that reviewed results from the pre-process and behavioral modules, 4) an input file construction and submission module, and 5) a machine learning module that employs various clustering techniques to be specified at run-time. As with most signature-based or behavior-based

detection mechanisms, careful attention was paid to false positive and false negative rates which can serve to reduce overall detection effectiveness.

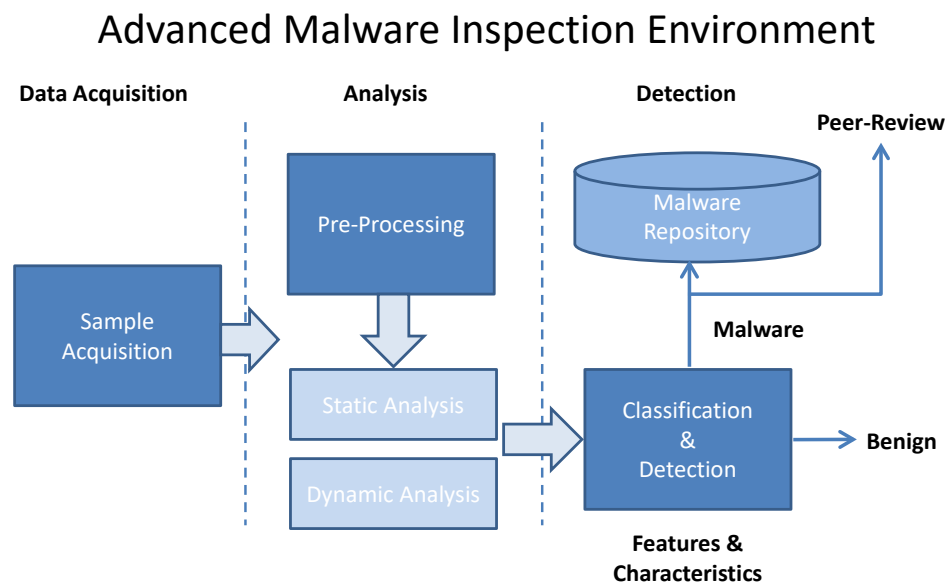


Figure 1. System Overview

Data Acquisition. The first step was to acquire sufficient samples needed to conduct study's experiments. This study needed to acquire three types of samples: 1) known malware, 2) known benign files and 3) unknown files. Known malware and benign files was acquired through industry partnerships and open source repositories available for malware researchers. However, this required more effort and time than anticipated. This study leveraged malware repositories from across industries and communities of interest. The malware samples were acquired through relationships with Virus Total, VirusShare, VXHeaven and Contagio. There are a number of such malware repositories that contain polymorphic and metamorphic samples specifically for malware

research. Symantec had previously agreed to share samples. However, due to personnel changes at Symantec this was no longer possible in time for the study. A majority of the malware and benign samples were acquired through Virus Total and VirusShare. These databases provided deep metadata and samples to track various types of malware. This database also provides other details such as unique files, cloned files and file identifiers. Working with Virus Total and VirusShare, the researcher was able to obtain samples (malware, benign and unknown), metadata/labels and other key malware classification information. Unknown files were acquired through the same means and other open source repositories for malware research. The unknown files were acquired from VirusShare, VXHeaven and Contagio.

Analysis and Feature Extraction. As discussed previously, static and dynamic analysis provides a means to observe and identify malicious behaviors and characteristics. Pre-processing of samples is needed to identify obfuscated malware variants. This step provides the ability to gather multidimensional topological data from malware samples. This step extracts file properties and other detailed features extraction via static and dynamic analysis. The integrated prototype system confirmed the inventory of functions through static analysis. The integrated prototype system evaluated execution and observed execution of behaviors through dynamic analysis. While many families of malware obfuscate and encrypt modules, many commercial vendors have incorporated “malware-like” obfuscation techniques to protect intellectual property. Pre-processing was needed to ensure the dataset contains representative samples for all three data types. As expected, static and dynamic analysis provide sufficient details, features and characteristics for identifying samples. These dynamic malware features consisted of observing malicious

behaviors, triggered behaviors and malicious characteristics.

The integrated prototype system produced an environment to conduct both static and dynamic analysis tools in order to extract multidimensional features from known and unknown samples. The integrated prototype used the static tool known as IDA SDK v6.95 (IDAPro) in order to extract static features and topological information. An example of static analysis is shown in Fig. 3. However, the extraction software development kit (SDK) provides a means to submit and extract feature information. As discussed in the literature review (Bilar, 2007), previous research suggests that there exists a limited set of operations that may point to malware; operations such as MOV, ADD, LEA, SUB, AND, INC, OR, NEG, XOR, ASSIGN (XCHG), STACK (POP), and CONTROL_C (JMP) have been used for malware detection in other studies. Bilar (2007) observed that the total universe of Microsoft 32-bit executables was 398 opcodes. Upon deeper inspection, Bilar (2007) observed that non-executable opcodes account for 192 of the 398 opcodes. Further, Bilar (2007) found that malware accounted for 141 of the 398 opcodes. Analyzing the 141 malware opcodes further, Bilar (2007) found that 72 opcodes accounted for over 99% of the malware opcodes, 14 opcodes accounted for over 92% malware opcodes, and 5 opcodes accounted for over 65% malware opcodes found (Bilar, 2007). Thus, based on Bilar's research (2007) the number of opcodes needed for improved detection was somewhere between the 14 and 72 opcodes. The opcodes captured during feature extraction was used for the static feature component of the multidimensional topological data.

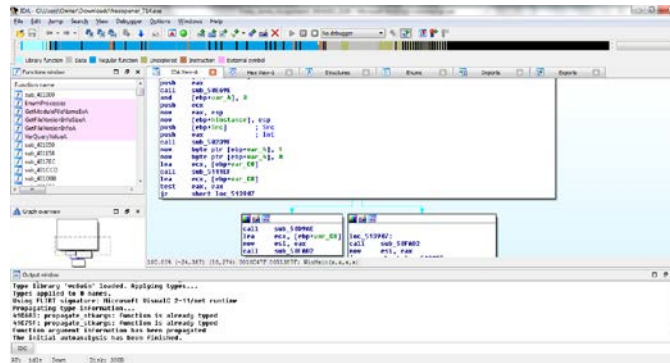



Fig. 3. Static Analysis - IDAPro

As discussed previously, Dynamic Analysis is needed to collect additional features during execution. Dynamic Analysis was performed on the basis of system-wide quantitative data flows from previous research (Wüchner et al., 2014). System-wide data flow analysis allows the researcher to capture the actual processes executed and the context of the system calls within a specific time frame within an operating environment (Wüchner et al., 2014). System calls were captured and aggregated as part of the integrated prototype environment. Dynamic Analysis features were captured for the purpose of augmenting the dynamic feature component of the multidimensional topological data. As previous researchers found, features such as File Size (FSize), File Type (Ftype), Malware Type (Malware_Type), Virtual Machine Aware (VM_Aware), Compiler Type (Comp_Type), Internal or External Libraries (Libraries) and Encryption Function (EncrypFunction) have been proven in the past to be good indicators of malware during execution (Ulrich Bayer et al., 2010; Devesa et al., 2010; Wüchner et al., 2014). The dynamic analysis information was augmented with both the file and static feature information for a complete data set for a single sample. An example output from an open source dynamic analysis tool – Cuckoo – is shown in Fig. 4. During the dynamic analysis step, a sample was placed into an operational virtual machine environment and system calls were captured. The dynamic analysis features were combined with file and static analysis data and stored into a single

repository for the dataset being processed. Some programming/scripting enabled the automated data collection and dataset construction.



Category	Started On	Completed On	Duration	Cuckoo Version
FILE	2014-10-11 19:39:45	2014-10-11 19:41:08	R1 seconds	1.2.0a6

Machine	Label	Manager	Started On	Shutdown On
cuckoo1	cuckoo1	VirtualBox	2014-10-11 19:39:46	2014-10-11 19:41:08

File Details

File name	putty.exe
File size	49536 bytes
File type	PE32 executable (GUI) Intel 80386, for MS Windows
CRC32	135f5655
MD5	7a05fc535ff6de7e0208a29fa2ffcd
SHA1	44ac2504a02af64ee342adaa3ea70b868185906f
SHA256	abcc2a2082813024839cf8c462ccffdc0f2c03a51a438d0209a035c8cc027
SHA512	6da636ab478a7f40127c706272ff9b2862a5e950f434e8509b7c1ff2da6c87003a76495c9b62c56d80334cf8aa10c8c343575a79aa853e42c3305361411e#
Sysprep	None
PEID	None matched
Yara	None matched
VirusTotal	Normal View/Total Scan Date: 2014-10-11 13:59:29 Detection Rate: 0/54 (0.0%)

Fig. 4. Dynamic Analysis - Cuckoo

Binary Feature Extraction. The study utilized three types of multidimensional topological data: 1) file features, 2) static analysis features, and 3) dynamic analysis feature for detection. The study treated input features as independent variables for the study. The emphasis of the study was to improve detection rates for polymorphic malware. Currently, detection rates range from polymorphic detection rates using advanced classifiers range from 68.75% to 81.25% (Amos et al., 2013). These detection rates used Bayes and Multilayer Perceptron techniques to establish a baseline for effective detection of various types of polymorphic malware (Amos et al., 2013). Due to the polymorphic nature of the malware the study (dependent variable) controlling for time (control variable) for polymorphic malware samples that have been collected from the malware analysis community. The independent variables file features, static analysis features and dynamic analysis features were used as a single input of features for the sample in question. In other words, the topological file properties/features extracted from

the executable/sample, the static analysis features – the topological properties/features extracted from code analysis for the executable/sample and dynamic analysis features – the observational behavioral features extracted from the execution of the sample were all used for detection purposes. The detection rate dependent variable was defined as the effective detection rate for malware in terms of Accuracy (Number of correct assessments)/Number of all assessments) and the control variable can be defined as time (control variable) – as polymorphic samples mutate changes occur and the intervening variable defined as external calls/libraries.

Feature Aggregation and Detection. All information collected during file property, static and dynamic analysis was compiled into a single database. The researcher explored various tools and databases (e.g. Elasticsearch) to store extracted features from the various modules based on a unique identifier. Elasticsearch enabled the construction of linked features and enabled analysis to be conducted multiple times. Ultimately, the database (all features linked to the MD5 hash) was converted into an Attribute-Relation File Format (ARFF). The ARFF file was used as an input file for the Weka Open Source Machine Learning tool. Weka was used to perform cluster analysis on the ARFF input file for each dataset. Weka is a collection of machine learning algorithms for data mining tasks. The algorithms were directly applied to the three datasets through a programming application interface (API) or graphical user interface (GUI). Weka was used to perform a number of functions and to perform advanced analysis on the experimental datasets. Weka was used to conduct additional analysis regarding feature isolation and hidden feature dependencies. Weka was used to advance the understanding of the relationships between the features for each of the three datasets. Weka algorithms available were used

to perform advanced functions such as data mining pre-processing, data classification, regression analysis, clustering, association rules, attribute selection and data visualization.

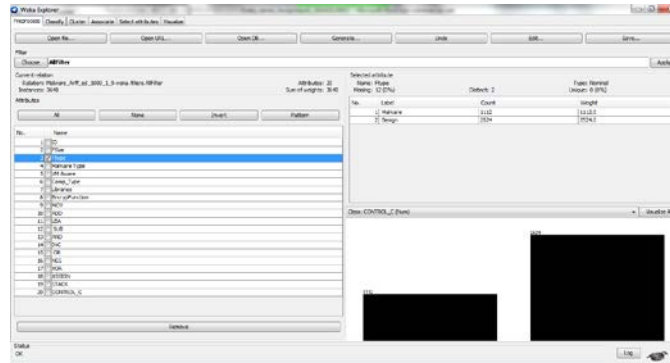


Fig. 5. Weka User Interface

The extracted multidimensional topological features in the form of an ARFF file were then submitted to Weka using the various clustering algorithms. Weka was used to perform detection based on both supervised and unsupervised learning. A malicious feature matrix was constructed and used to train the three clustering algorithms in Weka. These algorithms were then used for detection to identify known malware, benign and unknown files. These features were correlated with attributes in the Weka machine learning environment. The study considered and evaluated the three cluster algorithms to explore, weight and strengthen possible detection techniques. A number of other algorithms have been researched and explored in Weka. This study focused on MB, IBk and DLMLP. These were selected based on a number of factors including recent studies, new approaches and new techniques.

Other machine learning algorithms considered included Belief Propagation (BP), General Belief Propagation, Loopy Belief Propagation, Dynamic Bayesian Networks, Hidden Markov Model (HMM), and Markov Random Field (MRF). It was believed

that the combination of MB, IBk and DLMLP provided the study with sufficient and broad types of algorithms for detection purposes. IBk provided advantages over LSH by leveraging connectivity-based and proximity-based clustering. DLMLP improved BP techniques. New research has found that artificial neural networks (ANN) can leverage belief propagation for clustering or classification but do so inefficiently (Gruber et al., 2017). ANN's build networks of neurons, share information between neurons and propagate results throughout the network using weights or beliefs (Gruber et al., 2017). DLMLP improves the ANN approach by creating localized networks to share "beliefs" (Gruber et al., 2017). The DLMLP approach builds localized or shallow nets and distributes beliefs locally (Gruber et al., 2017). These shallow networks work more efficiently as new "beliefs" do not need to be propagated throughout an entire network (Gruber et al., 2017).

In addition to IBk and DLMLP, an ensemble technique such as Bootstrap Aggregating (Meta Bagging) was also evaluated. Meta Bagging may provide a model in which the ensemble voting can be done with equal weights or weighted attributes for various features extracted. This approach may allow the evaluation model to promote variance based on training data to improve malware detection. Once satisfied with the approach (through a series of techniques), the approach was tested against a larger dataset to evaluate the efficacy for detection.

Threats to Validity

Researchers who use quantitative measures and experimental methods to explore hypothetical questions are categorized as quantitative research (Hoepfl, 1997). Quantitative research emphasizes the measurement and analysis of causal relationships

between variables (Denzin & Lincoln, 2006). Quantitative research uses numbers, charts and graphs to explain or describe relationships that exist between variables (Bogdan & Biklen, 1998). Quantitative research use charts and graphs to illustrate relationships between variables, populations and results of tests (Bogdan & Biklen, 1998). The quantitative research process allows the researcher to understand the problem, develop a methodology to study the problem and if needed generate hypotheses to be tested in terms of numbers that can be quantified and summarized (Bogdan & Biklen, 1998). Quantitative research utilizes a mathematical process to explore, analyze and explain findings (Charles, 1995). Generally, quantitative research "...supported by the positivist or scientific paradigm, leads us to regard the world as made up of observable, measurable facts" (Glesne & Peshkin, 1992, p. 6). Quantitative research attempts to segment and control the experimental environment such that the research can be generalizable to wider populations and similar situations (Winter, 2000). Quantitative researchers must construct an experiment to study the problem and use standard measures if the research is to be useful (Crocker & Algina, 1986). In simple terms, quantitative research design must ensure the validity of the test and measurements (Crocker & Algina, 1986). Special attention must be made to ensure replicability or repeatability of the experimental results (Crocker & Algina, 1986).

Internal Validity

Internal validity is concerned with the rigor of the study design (Crocker & Algina, 1986). Internal validity can be determined by the degree of control over the potential extraneous variables impacting the quantitative study (Crocker & Algina, 1986). Researchers must control such issues as confounding variables in order to minimize the

potential for alternative explanations for results (Crocker & Algina, 1986). Researchers must also control such issues as variable treatment for testing inputs such that there is more confidence that the effects observed are due from research manipulation (Crocker & Algina, 1986). Campbell & Stanley (1963) have provided the research world with the most authoritative source for understanding threats to internal validity.

Threats to internal validity compromise the study's confidence in establishing relationships exists between the independent and dependent variables (Morse, Barrett, Mayan, Olson, & Spiers, 2008). Campbell & Stanley (1963) identified eight threats to internal validity: history, maturation, testing, instrumentation, regression, selection, experimental mortality, and an interaction of threats. Each are described and addressed below.

History. History becomes a threat to validity when other factors impact the study, subject or findings by the passage of time (Campbell & Stanley, 1963). For the purpose of this study, the research conducted and the malware samples do not suffer from this threat as malware remains in effect for long periods of time. History is not an internal validity concern to be addressed for this study.

Maturation. Maturation becomes an issue when changes occur for subjects or testing procedures and impact the outcomes or findings for the study (Campbell & Stanley, 1963). History and maturation are both major concerns for longitudinal studies (Campbell & Stanley, 1963). For the purpose of this study, the malware samples and procedures will remain static throughout the study. Therefore, maturation is not an internal validity concern to be addressed for this study.

Testing. The testing process may introduce issues with studies and test results because of repeated testing and not the intervention or manipulation of variables by the researcher (Campbell & Stanley, 1963). Testing issues often occur as a result of researchers administering pretests and posttests. For the purpose of this study, the malware samples are extracted through a single repeatable process (Campbell & Stanley, 1963). The features are extracted and subjected to testing. The feature dataset remains static but not influenced by repeated testing. Therefore, testing is not an internal validity concern to be addressed for this study.

Instrumentation. Instrumentation issues arise when instrument calibration or measurement protocols change and the results seemingly change due to measurement rather than to a true treatment effect (Campbell & Stanley, 1963). Measurement and evaluation protocols must remain constant for over the course of all experiments (Campbell & Stanley, 1963). For the purpose of this study, all testing protocols and clustering algorithms used for detection will remain the same across all tests. Only features sets will change as part of the experimental testing. Therefore, instrumentation is not an internal validity concern to be addressed for this study.

Regression. A regression threat occurs when subjects have been selected on the basis of previous test scores (low and high) (Campbell & Stanley, 1963). Issues with regression occur for studies that have test scores move closer to the mean as a result of repeated testing (Campbell & Stanley, 1963). For the purpose of this study, repeated testing will have no effect on testing outcomes. Malware samples are subjected to a standard single extraction process. The features are extracted and subjected to testing.

The feature dataset remains static but not influenced by repeated testing. Therefore, regression is not an internal validity concern to be addressed for this study.

Selection. The selection threat is of utmost concern for many experimental studies (Campbell & Stanley, 1963). Studies that cannot be randomly assigned subjects to test groups or treatment groups have issues with measurement before and after treatment intervention (Campbell & Stanley, 1963). For the purpose of this study, each of the test groups have both random selection and random assignment. The test groups remain static throughout the study. The study design and testing protocol specifically addresses the internal validity issue of selection. Therefore, selection is not an internal validity concern to be addressed for this study.

Experimental mortality. The issue with experimental mortality occurs when testing groups suffer from attrition, withdrawals, or dropouts (Campbell & Stanley, 1963). Experimental mortality becomes problematic with ongoing testing by losing subjects from comparison groups subsequent to random selection (Campbell & Stanley, 1963). For the purpose of this study, each of the test groups will remain static after random selection and random assignment. The test groups will remain static and will not suffer from mortality throughout the study. Therefore, experimental mortality is not an internal validity concern to be addressed for this study.

Interaction of Threats. The final issue threatening internal validity is the interaction of the threats across the study (Campbell & Stanley, 1963). Most experimental studies are concerned with the interaction of selection and maturation (Campbell & Stanley, 1963). Internal validity can be impacted when subjects are selected and assigned to groups based on the subject maturation (Campbell & Stanley,

1963). For the purpose of this study, random selection and random assignment to parallel groups address the interaction of threat issues directly and effectively. Random selection and random assignment address most internal validity issues except experimental mortality. Differential selection is controlled for this study as random assignment creates groups that are equivalent with respect to known and unknown variables (Campbell & Stanley, 1963). Lastly, influences of maturation on other variables can be ruled out because of static and consistent test groups. Other issues considered for interaction of threats include the reactive or interaction effect of testing subject multiple times. Reactive validity issues concern pretesting procedures and the potential increase or decrease of a subject's sensitivity or responsiveness to the experiment (Lana, 1959; Willson & Putnam, 1982). Again, malware samples are subjected to a standard single extraction process. The features are extracted and the data is subjected to testing. The feature dataset remains static but not influenced by repeated testing. Therefore, reactive issues are not an internal validity concern needed to be addressed for this study.

External Validity

Campbell and Stanley (1963) provided the foundational groundwork for examining issues related to external threats to validity. Smith & Glass (1987) classified threats to external validity in a number of related issues concerning 1) population validity, 2) ecological validity and 3) external validity of operations (Smith & Glass, 1987). Population validity and selection treatment concerning the sampling and selection process that impact generalizability (Smith & Glass, 1987). The ecological validity issues concern experimenter effects, multiple-treatment interference, reactive arrangements, time and treatment interaction, history and treatment interaction (Smith & Glass, 1987).

Lastly, issues for external validity of operations concern specificity of variables and pretest sensitization (Smith & Glass, 1987). Each of these are discussed below.

Population validity. Population validity refers to the extent to which findings can be generalized from the study sample to that of the larger target population and potentially to subgroups within the larger target population (Smith & Glass, 1987).

Utilizing sufficiently large and random samples will increase the population validity of results. For the purpose of this study, sample size far exceeds the minimum sample size requirement. Further random selection and random assignment reduces the concern for this external validity threat. Unfortunately, population validity is a threat in virtually all experimental studies. The collection of malware samples from various malware collaboration sites may under-represent new or emerging malware. Most researchers are forced to use samples from accessible populations. In most cases these samples represent the best group of participants available for a study. Therefore, the degree of representativeness depends on how large the accessible population is relative to the target population. In this study, the collection of samples exceeds one million and should be sufficiently large to satisfy this concern.

Ecological validity. Ecological validity refers to the extent that findings from one study can be generalized to another experimental setting (Rossow et al., 2012; Smith & Glass, 1987). The experimental setting must accurately and in sufficient detail allow other researcher to understand settings, conditions, variables, and contexts extent that findings from one study can be generalized to another experimental setting (Rossow et al., 2012; Smith & Glass, 1987). Therefore, ecological validity represents the extent to which findings are independent from the setting or location in the study was performed

extent that findings from one study can be generalized to another experimental setting (Rossow et al., 2012; Smith & Glass, 1987). For the purpose of this study, ecological validity is not a concern. The environment, settings, hardware, and software (including versions) was disclosed as part of the study.

Temporal validity. This external validity issue refers to the ability for the research and findings to be generalized across time (Smith & Glass, 1987). Researchers conducting experimental research rarely mention temporal validity because most studies are conducted within a relatively short period of time (Smith & Glass, 1987). For the purpose of this study, the timeframe is sufficiently short such that issues with experimental design and data collection are minimized. It should be noted that only malware sample and benign samples created within the past eighteen months was used for the study. Also, the single case experimental design was selected for this study as the protocol facilitates the time to select, conduct and analyze results. Therefore, temporal validity is not a concern for this study.

Multiple-treatment interference. This external validity issue occurs when the same research subjects are exposed to multiple interventions (Smith & Glass, 1987).. Multiple treatment interference also occurs when subjects are selected for multiple studies with multiple interventions (Smith & Glass, 1987).. Thus, test results or outcomes may not be generalizable for the target population because of the sequence of interventions that was administered prior to or during the experiment (Smith & Glass, 1987). For the purpose of this study, the sample groups are each distinct with a population of 200,000. However, the sampling technique to be used is random selection without replacement across three types of samples. The total sample size is over 2 million

and assuming the selector is not repeated the probability of being selected is remote - malware (1:1 million), benign samples (1:750,000) and unknown (1:300,000). In addition, the single case experimental design selected for this study facilitates the manipulation of variables over time. However, the feature dataset remains static and not influenced by repeated testing. Therefore, multiple-treatment interference is not an external validity concern to be addressed for this study.

Researcher bias. This external validity issue arises due to processes or procedures the researcher introduces while conducting the experiment (Smith & Glass, 1987). Researcher bias is an external validity issue as the study's finding may be dependent upon the researcher and thus not generalizable for the intended target population (Smith & Glass, 1987). The more unique the researcher's process, protocol and procedures are the more they interfere with findings (Smith & Glass, 1987). The single case experimental design was specifically selected for this study as the testing protocol facilitates the manipulation of variables and testing over time. SCED is an accepted testing protocol for unique subjects such as polymorphic malware. The testing protocol and procedures are consistent with each test. The dataset remains static and not influenced by repeated testing. Therefore, researcher bias is minimized for this study.

Reactive Arrangements. This external threat to validity occurs when subject response changed due to the fact that they know they are being studied (Smith & Glass, 1987). Reactive arrangements reduce external validity because findings may become less generalizable as the results have been compromised or interventions not properly measured (Smith & Glass, 1987). For the purpose of this study, certain malware samples are sophisticated enough to know they are being analyzed. For example, advanced

malware will check environmental conditions prior to executing in a virtual environment like dynamic analysis. In this case, the additional static analysis and file property analysis provide missing pieces as part of the standard feature extraction process. Therefore, reactive arrangements have been dealt with as part of the feature extraction process. External issues dealing with reactive arrangements have been adequately compensated for and is not an external validity concern to be addressed for this study.

Order bias. This external validity issues occurs when observed findings become dependent upon the order in which the multiple interventions are administered (Smith & Glass, 1987). In this case, findings resulting from the administering interventions in a particular order may not be confidently generalized to situations in which the interventions are applied differently (Smith & Glass, 1987). The single case experimental design was specifically selected for this study as the testing protocol facilitates performing various tests in a structured protocol. However, the dataset, clustering algorithms and testing procedures are not structured such that order dictates findings or outcomes. Therefore, order bias is not a concern for this study.

Matching bias. This external validity issues arises in cases where subjects tested do not match those in the accessible population (Smith & Glass, 1987). Thus, matching bias is a threat to external validity to the extent that findings cannot be generalized to general population (Smith & Glass, 1987). This issue is a concern when the subjects studied fall out of the sampling frame (Smith & Glass, 1987). For the purpose of this study, the samples collected and selected become the subject studied. Therefore, study findings will directly correlate to sample frame and reduces the concern for this external validity threat.

Specificity of variables. This issue is a threat to external validity to almost every study. Researchers attempt to define and control a number of variables within the experiment in order to generalize findings (Smith & Glass, 1987). However, the number of variables that impact generalizability is large and include 1) a specific type of individual, 2) at a specific time, 3) at a specific location, 4) under a specific set of circumstances, 5) based on a specific operational definition of the independent variable, 6) using specific dependent variables, and 7) using specific instruments to measure all the variables. Experiments with unique participants, testing time frames, developed context, testing conditions, and other variables, the less likely the experiment will produce findings that are generalizable. In order to counter threats of specific variables, the researcher must operationally define variables and be able to represent the contextual aspects of the study setting and use extreme caution when generalizing findings. The single case experimental design was specifically selected for this study as the testing protocol facilitates the study setting, baseline testing, manipulation of variables and producing findings over time. SCED is an accepted testing protocol for unique subjects such as polymorphic malware. The setting, setting variables, testing protocol and procedures are consistent with each test. Therefore, the consistency of variables held constant and the threat posed by specificity of variables is minimized for this study.

Treatment diffusion. This external validity threat arises as subjects are exposed to multiple interventions and these manipulations become diffused or impact other treatments threatening the researcher's ability to generalize findings (Smith & Glass, 1987). In essence, the manipulation of testing variable contaminate one or more of the treatment conditions and prevents the study from being replicated in the future (Smith & Glass,

1987). SCED is an accepted testing protocol for manipulating treatments for experiments. The establishing of baselines, applying treatments and measuring the impact of treatments is consistent with each test. Therefore, the external validity of treatment diffusion is minimized for this study.

Pretest x treatment interaction. This external validity threat arises when the administering of pretests changes the participants' responsiveness (sensitivity increases or decreases) and renders the observed findings for the pretest group generalizable to only that group and not to the untested group or larger population (Smith & Glass, 1987). In order to utilize generalizable pretest findings, researchers must carefully choose pretest conditions and treatments, characteristics of the research participants, the duration of the study, and nature of the independent and dependent variables. SCED was specifically selected for this study as the testing protocol facilitates performing various baseline testing (pre-tests) in a structured protocol. The proposed baseline tests will not influence further testing post intervention or manipulation. The baseline tests are quantitative in nature and not attitudinal so future influence from pretest is non-existent. Therefore, pretest x treatment interaction is not a concern for this study.

Selection x treatment interaction. This external validity threat arises when treatment groups differ non-treatment groups and thus findings cannot be generalizable to the larger population (Smith & Glass, 1987). The selection-treatment interaction threat occurs more often when randomization is not used for selecting intervention and test groups (Smith & Glass, 1987). However, this external threat to validity can still prevail when randomization is used as it does not guarantee that the group selected is representative of the target population (Smith & Glass, 1987). For the purpose of this

study, there are a number of groups selected for testing from a sample population. The sample set is large (over 1 million samples) and the multiple individual test groups far exceeds the required minimum sample size. Therefore, the likelihood that the population studied do not represent the larger population is remote is not an external validity concern for this study.

An experiment is deemed to be valid as long as relationships can be established and the results possess internal and external validity (Campbell & Stanley, 1963). The proposed experiment has addressed internal validity concerns from an experimental design, sampling and a replicative perspective. Internal validity is maintained when conditions are changed or manipulated and the results are measured or observed (Campbell & Stanley, 1963). Thus, internal validity provides assurances that results are in fact due to direct treatment and not due to other circumstances (Campbell & Stanley, 1963). As such, the proposed research has also addressed internal and external validity concerns from a number of perspectives. The proposed study has a controlled approach and environment and experimental design addresses generalizability of findings. There has been considerable amount of thought given to how the results and findings can be extended to broader populations, groups, environments, and contexts outside the experimental setting (Rossow et al., 2012; Smith & Glass, 1987). Consequently, the proposed experimental design and approach should satisfy the issues raised for internal and external validity. The proposed experimental design has addressed the number of issues raised concerning internal and external validity as defined by Campbell and Stanley (1963). A number researchers have contributed to addressing threats to internal and external validity not only from analyzing the results but more importantly addressed

by quantitative research and the experimental design (Smith & Glass, 1987). The proposed study kept internal and external validity in mind by selecting an accepted experimental design (SCED), designing the experiment with a testing protocol, adopting a sampling strategy and insuring any possible outside influences are controlled or mitigated such that to the greatest extent possible concerns with validity have been addressed ensuring the validity of study.

Sample

The sample used for this study is an unrestricted probability sampling or simple random sampling or pseudo-random sample. The study conducted a random selection and random assignment of files for the three datasets. However, the dataset is considered to be stratified dataset. The sample set (files) included three data types or strata on a percentage basis: 1) known bad samples – known malware (polymorphic/metamorphic), 2) known good samples – known benign samples and 3) unknown samples – files that are unknown to the researcher but was determined post- test. Although, malware detection is a straightforward bi-partite problem (malware or non-malware) the study also included a “blind” dataset for validating detection results.

Known Malware Dataset. The study had a sample set of over 1 million malware samples of polymorphic malware targeting 32-bit and 64-bit malicious executables. Although, there are a number of malware types, this study will group the distinct malware types into a single class - malware. A majority of malware samples were acquired from Virus Total, Virus Sign, and VxHeaven. The samples used represented various malware types including: backdoors, Trojans, bots, worms, and viruses. According to Symantec (2016), there are approximately 1 million new malware samples released and discovered

daily. In conducting a simple sizing analysis, a sample size of at least 666 yields 99% confidence level and 5% confidence interval for a population of one million. A total collection of 1,009,108 malware samples were acquired for the study. Therefore, the study dataset far exceeded the required number of malware samples needed for developing a valid sample size and representative dataset.

Known Benign Dataset. The known benign samples were collected from Virus Total and Virus Sign. These files were tested as “clean” from the providers and probably were collected from clean installation disks and other known clean distribution sites. Special attention was made to collect vendor software with valid digitally signed installation files. In conducting a simple sizing analysis, a sample size of at least 666 yields 99% confidence level and 5% confidence interval for a population of 1 million. A total collection of 756,322 known benign samples were acquired for the study. Therefore, the study dataset far exceeded the required number of known benign samples needed for developing a valid sample size and representative dataset.

Unknown Dataset. The unknown samples were collected from Virus Total, Virus Sign and VirusVx. A majority of these samples were collected from Virus Total. Users can submit files to sites like Virus Total as unknown and eventually these files would be determined as malware or benign. Until the files are determined to be malware or benign they remain in an unknown status until a determination has been made. Again, in conducting a simple sizing analysis, a sample size of at least 666 yields 99% confidence level and 5% confidence interval for a population of 1 million. A total collection of 748,976 of unknown samples were acquired for the study. Therefore, the

study dataset far exceeded the required number of known benign samples needed for developing a valid sample size and representative dataset.

In summary, the dataset for the study was developed as a simple stratified random sample. The study performed a pseudo-random selection via a python script for selecting and generating datasets on a percentage basis. The random assignment from a stratified sample of three sample subsets: 1) known malware (40%), 2) known benign (30%) and 3) unknown (30%). The study's sample size far exceeded the 666 samples required for a confidence level of 99%. The three data subsets had the following sample size: 1) malware – over 1 million samples, 2) known benign – over 750,000 samples and 3) unknown samples – just under 750,000 samples.

Data Analysis

Data analysis was conducted throughout the experiment. The focus of the experiments was to evaluate detection rates given the various features, weighting of the features and the advanced cluster algorithms leveraged during testing. There were a number of data analysis tools that were used to evaluate detection performance.

Sensitivity and specificity were two statistical measures that were used to evaluate malware detection performance (Kolter & Maloof, 2006). Sensitivity is the statistical measure used to understand and evaluate the proportion of True Positives in an experiment (Kolter & Maloof, 2006). The True Positive Rate (TPR) represents the proportion of correctly identified malware samples within a given test. Specificity is another important statistical measure and is used to represent the True Negative Rate (TNR) (Kolter & Maloof, 2006). Specificity measures the proportion of true negatives (non-malware) correctly identified within an experiment (Kolter & Maloof, 2006). The

two measures – Sensitivity and Specificity provide insight into detection rates from a true positive rate (sensitivity) and true negative rate (specificity). For any experiment, there is usually a trade-off between the two measures in terms of usability of results. Too many false positives or false negatives can erode the confidence of the detection approach. Depending upon detection goal, one of the measures may be more important than the other. In some cases, false positives (false alarms) significantly impact detection rates and experimental outcomes. On the other hand, false negatives (non-detection) allow researchers to evaluate characteristics and issues with missed detections. In a perfect world, sensitivity and specificity would both be 100%. However, achieving 100% for both is usually not achievable. Both measures can be graphed in order to analyze the tradeoffs between sensitivity and specificity. Receiver Operating Characteristic (ROC) depicts the curve between sensitivity and specificity. An example of ROC is provided below.

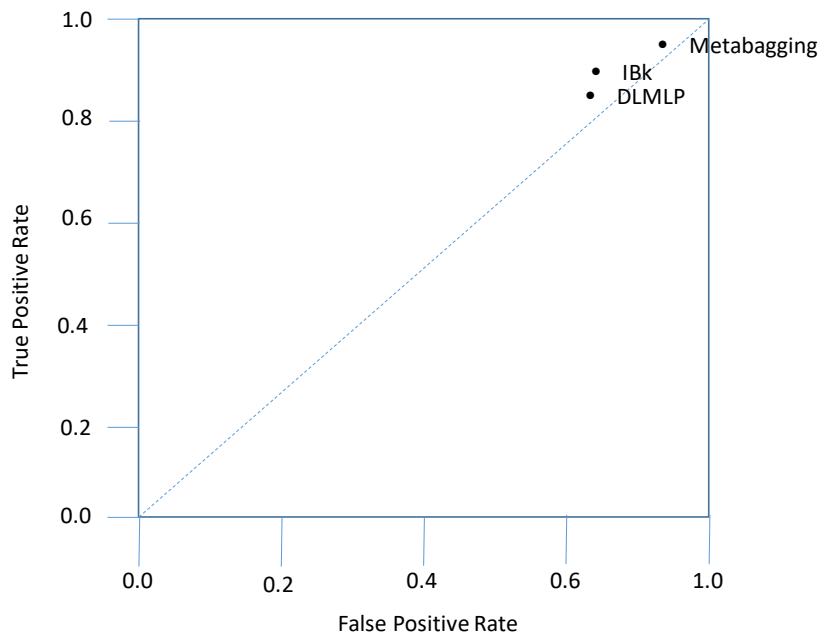


Figure 2. ROC Results

In order to evaluate other aspects of detection results, other statistical measures can be used to evaluate detection effectiveness that go further than sensitivity and specificity. Accuracy (ACC) and the Correlation Coefficient (CC) are additional statistical quality measures with machine learning used to explore deeper understanding of detection rates (Kolter & Maloof, 2006). In order to gain additional insight into detection rate with Accuracy and Correlation Coefficient the study must produce the results in a confusion table format. The confusion table allows the results to be fully examined. An example of the confusion table is provided below.

Table 2 Confusion Matrix

	TRUE ACCEPT	FALSE REJECT
TRUE (T)	True Positive	True Negative
FALSE (F)	False Positive	False Negative

The confusion matrix presented in Table 2 is an easy concept to evaluate detection outcomes and was generated for each test. Each entry in the table was used to evaluate the detection algorithm effectiveness. As part of the confusion matrix or truth table as it is also referred to, each of the rates were measured. The four entries in the confusion matrix were used to evaluate testing outcomes: 1) True Positive Rate, 2) True Negative Rate, 3) False Positive Rate and 4) False Negative Rate. The True Positive Rate is defined as the number of malware that was correctly classified as malware. In other words, the True Positive Rate (TPR) represents the proportion of positive instances of malware correctly detected. The True Negative Rate (TNR) represents the number of benign samples that were correctly detected as Benign. The TNR represents the proportion of negative instances detected correctly. The False Positive Rate (FPR)

represents the number of non-malware samples that were classified as malware. The FPR is in essence the proportion of negative instances incorrectly detected as positive (malware). The False Negative Rate (FNR) represents malware that was classified as Benign. Thus, the FNR is the proportion of positive instances wrongly classified as non-malware. The equations for each statistical measure are provided below:

$$TPR = \frac{TP}{TP + FN}$$

$$TNR = \frac{TN}{TN + FP}$$

$$FPR = \frac{FP}{FP + TN}$$

$$FNR = \frac{FN}{FN + TP}$$

Once the confusion matrix had been fully populated, the additional statistical measures of Accuracy and Correlation Coefficient were then applied. The equations for these two measures are provided below:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

$$CC = \frac{(TP \times TN) - (FP \times FN)}{(TP + FN)(TP + FP)(TN + FN)}$$

The malware detection effective rate was analyzed based on the results from the testing for the three clustering algorithms. These measurements were largely concerned

with correctly classified sample based on the statistical measures described above. The proposed study used these measures to evaluate detection results based on the various datasets for detection purposes.

Additional analysis was performed using Sensitivity, Specificity, Accuracy and Correlation Coefficient information for the various tests. Developing the confusion matrix provided a solid understanding of the results for each test. The various types of analysis were conducted to compare and evaluate detection rates across the three algorithms used during testing. Also, the results were evaluated using various sample sets, feature set and weighting of features. There were three tests conducted per algorithm. The testing approach used a benchmark test with two other tests for weighted feature testing per algorithm. The formatting of test results for the various experiments are presented in the next section.

Data Formats for Results

Once the base data (TPR, TNR, FPR and FNR) had been captured, additional data analysis was conducted using the ACC, CC, and ROC curve methodology. The detection model had the possibility of two classes – positive (malware) or negative (benign). The malware detection effective rate used analyzed the results from the three algorithms in terms of correctly classified sample based on the statistical measures described above. The study evaluated detection results based on mapping those results into a single classification set (p, n) – positive and negative. Given a confusion matrix, there are four possible outcomes given the set: 1) detected as a positive and correctly classified therefore classified as a True Positive (TP), 2) detected as a negative and correctly classified and correctly classified therefore classified as a True Negative (TN), 3)

detected as a positive but should have been classified as a negative therefore this is a False Positive (FP), and 4) detected as a negative but should have been classified as a positive therefore this is a False Negative (FN). The confusion matrix illustrates the set of instances in a two-by-two confusion matrix. The following example show the various rates achieved by clustering algorithm (MB, IBk and DLMLP).

Table 3 Example Confusion Matrix

MB	TRUE ACCEPT	FALSE REJECT
TRUE (T)	0.9630	0.0010
FALSE (F)	0.0031	0.0060
IBk	TRUE ACCEPT	FALSE REJECT
TRUE (T)	0.9780	0.0012
FALSE (F)	0.0080	0.0017
DLMLP	TRUE ACCEPT	FALSE REJECT
TRUE (T)	0.9970	0.0000
FALSE (F)	0.0000	0.0000

Further, the following example demonstrates the preliminary results for MB, IBk and DLMLP algorithms each with 20 unweighted features:

Table 4 Example Unweighted Testing Results

	True Positive	False Positive	True Negative	False Negative	ACC	CC
IBk	0.9630	0.0031	0.0010	0.0060	0.9741	0.0010
DLMLP	0.9780	0.0012	0.0080	0.0017	0.9971	0.0078
MB	0.9970	0.0000	0.0000	0.0000	0.9970	0.0000

From this data, a graph can be generated to demonstrate the effective rates achieved comparing the results across the three clustering algorithms. The graph below demonstrates the baseline testing and the achieved results across the three algorithms

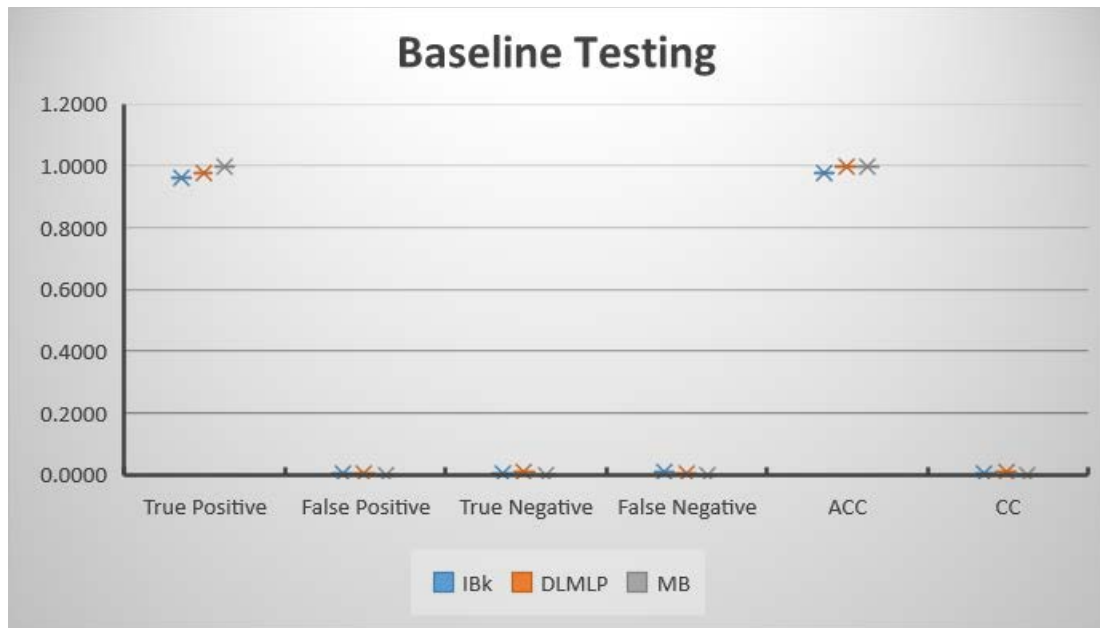


Figure 3. Comparative Baseline Testing Results.

Additionally, based on the result from the confusion matrix, the ROC graph can be constructed using a two-dimensional graph. The TP rate is plotted on the Y axis and the FP rate is plotted on the X axis. Overall the ROC graph enables the visualization of relative trade-offs between benefits (true positives) and the associated costs with detection (false positives) (Fawcett, 2006). The graph below demonstrates the ROC with the three classifiers (MB, IBk and DLMLP).

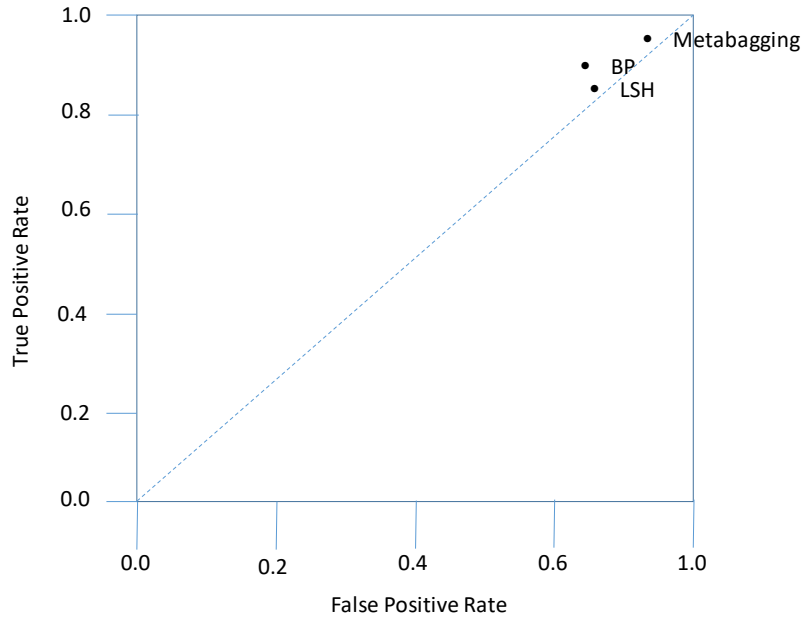


Figure 4. ROC Analysis.

Data analysis will provide a means to evaluate achieved results. Each experiment will provide a series of data sets for each experiment: Sensitivity, Specificity, Accuracy, Correlation Coefficient and ROC. These standard statistical measures provide deeper insight into the TPR, TNR, FPR and FNR produced by each experiment. The malware detection effective rate will analyze the result from the three clustering algorithms in terms of correctly classified sample based on the statistical measures described above.

Resource Requirements

The study has a number of resource requirements. The most critical resource for conducting the proposed study was the dataset. A number of unaffiliated websites (contagio, Virus Sign, VxHeaven, Virus Total, and others) contributed samples in order to achieve the 1 million malware sample target. Malware samples collected from the various websites were checked via the MD5 hash to avoid duplication. It is common for these websites share and collaborate on samples. Malware researchers from industry and

academia frequently find interesting samples and share finding across the community. In doing so, researchers leverage known malware samples for detection purposes and experiments.

The second issue was that of constructing a dataset with sufficient number of non-malware (unknown and benign) samples. Fortunately, Virus Total provided a majority of benign samples for the research. Virus Total also was able to provide unknown samples. Both sample types – unknown and benign were needed in order to evaluate the algorithm detection effectiveness. However, these samples were identified by the SHA256 hash versus MD5. Sample Identifiers (SampleIDs) were then generated for tracking purposes. All samples were collected, uniquely identified, analyzed and screened. All samples were analyzed to ensure that they contained contemporary characteristics and behaviors that protect intellectual property for commercial software. Duplicate samples were removed. These non-malware behaviors and characteristics are often mimicked by malware developers in order to bypass detection. In order to conduct a realistic evaluation, malware, benign and unknown samples were used for testing. Over 1 million benign samples were collected, analyzed and screened. Benign samples historically had been more difficult to obtain than the malware data set. For the collected 1 million benign sample collected only roughly 75% passed the screening criteria. Over .5 million unknown samples were collected, analyzed and screened. Roughly 50% of the unknown samples passed screening criteria. These samples were then used to create basis for the generation of the datasets used for testing. These non-malware samples were then used to evaluate overall detection effectiveness (False Positives and False Negatives) during testing.

A related data set issue was simply the storage of the malware data set. Once the various data sets had been collected, the malware and non-malware samples were stored in a secure and protected repository. The secure storage of malware data set needed to ensure non-detonation of samples within the experimental storage environment. The malware samples were kept in a .zip file with password to prevent detonation or infect the analysis platform/environment. Another issue is the size and number of malware samples. The objective of the research is to analyze millions of samples in order to evaluate efficacy and validate the proposed detection approach. This study acquired over 2 million total samples. The storage will require over two terabytes of storage for malware and non-malware samples.

Lastly, the tools required to perform the analysis had issues. IDAPro did not ultimately provide the necessary API for bulk analysis. Open Source tools such as Capstone provided better programmatic access and was readily available. These tools provided static and dynamic analysis and was used to extract features from the sample repository. The number of modules were developed to extract the various static and dynamic features. The seven feature extraction modules included: 1) Compiler Extractor (Embedded Compiler, Compiler Type, Compiler Used), 2) Encryption Extractor (Encrypted Indicator, Encrypted Type), 3) Library Call Extractor (Call Indicator, Call Counter, Library Call) , 4) Malware Extractor (Malware Indicator, Malware Type), 5) Virtual Machine Aware Extractor (VM Aware Indicator), 6) OpCodes Extractor (MOV, ADD, LEA, SUB, AND, INC, OR, NEG, XOR, XCHG, POP, and JMP) and 7) File Properties Extractor (File Type and File Size). The dynamic analysis tool (Cuckoo) was leveraged for some feature extraction and detection. The tools used for these upgrades

did impact schedule but did not affect other tools such as IDAPro and Weka. Managing the overall analysis environment was challenging and but was technically achievable.

The researcher had a mitigation plan or alternative plan for all issues raised. In terms of malware samples, the researcher had collected a majority of the malware samples. Virus Total was able to supply both benign and unknown samples. There were a number of blogs which offer malware samples as well. However, the researcher had a strong desire to work with mainstream malware research data sets. In corresponding with both Symantec and Virus Total, there is a high degree of confidence that at least one of the firms would deliver the number of benign samples needed. The researcher has also found a number of malware research sites that provide “benign” samples as well. Therefore, the non-malware data set may have to be constructed using samples from Virus Total. Additional analysis was required to ensure that the samples represent the types of samples needed for the research. The additional analysis has been factored as part of the proposed effort. Mitigating the secure storage of malware was a process that checks and changes file extensions to prevent file execution. In addition, shipments of malware were done via third-party compression software such as WinZip. Each set of Zip files transferred will also be password protected in order to prevent file execution. In terms of meeting storage requirements for the project, a high performance 24 TB Network-Attached Storage (NAS) device was purchased to house both malware and non-malware data sets. The only remaining concern for storage was a backup for experimental data set. The researcher did explore cloud-based backup solutions.

In order to construct a solid analysis environment, the researcher began with the latest releases of a number of static analysis tools (IDAPro, Capstone and objdump),

Open Source dynamic analysis tool (Cuckoo) and the Open Source machine learning tool (Weka). The researcher had sufficient understanding of the computing requirements for the various tools as well as the underlying operating system compatibility issues. This approach allowed and enabled the researcher to begin the assembly of the analysis environment and provided a programmatic means to investigate integration issues with the latest releases. The assumption is that the latest releases will have a vibrant technical support network and issues raised were supported by the community of interest. Most of the other resources are in place or can be easily secured. A full description of the resources needed is provided below.

Table 5 Experimental Resource Requirements

Type	Resource	Description	Possess?	Risk Assessment
Dataset	Malware Samples	Semantec Agreement	No	Other Sources are available
Hardware	Server	Dell Server i7 64bit w/Linux 16GB RAM	Yes	New Server/No Warranty
Hardware	Workstation	Toshiba i3 64bitw/Windows7	Yes	Current Laptop/No Warranty
Hardware	Storage	4 TB Storage	No	To be Purchase
Hardware	Network	Internet Connectivity	Yes	Wifi Enabled
Software	IdaPro	v6.6	Yes	Education Version/Limited Version
Software	Volatility	v2.3	Yes	Education Version/Limited Version
Software	Yara	v3.2	Yes	Researcher owned
Software	Weka	v3.7.9	Yes	Open Source
Software	Cuckoo	v1.1	Yes	Open Source
Software	Python	v2.7	Yes	Education Version/Limited Version
Software	Microsoft Office	2010 or better	Yes	Education Version/Limited Version

Most of the resources needed fall into the software category. The researcher had developed risk mitigation for each item. The only issue that presented itself was Weka during experimental time period. WEKA had a major upgrade and release that took time to configure. Yara and Volatility are tools for analysis. Volatility is an advanced memory analysis framework for deeper inspection of memory. This software can be used to analyze memory dumps after malware has been detonated through dynamic analysis.

Yara is a tool that assists malware researchers to identify and classify malware samples. Contingencies for most issues were use or leverage academic or demonstration licenses offered by the software vendor. No issue arose during integration or testing.

Summary

This research developed a unique detection methodology using multidimensional topological features in a machine learning environment. The multidimensional topological features approach combined static and dynamic analysis in addition to unique file properties. The study was designed to develop a quantitative experimental prototype using multidimensional topological data with machine learning using advanced clustering that provided improved detection for polymorphic malware above 81.25% for polymorphic malware. The study utilized quantitative methods to provide objective measurements for experimental results as described by Babbie (2010). The proposed data collection, data analysis and presentation experimental results were conducted using standard and accepted computational techniques (Babbie, 2010). The study was designed with correctly assembled and representative datasets, proper experimental methodology design such that the experiment can be reproducible and safely conducting experiments such that the malware cannot infect the research environment (Rossow, 2013). The experimental methodology was designed to establish, capture, analyze and evaluate results with the rigorous quantitative evaluation measures offered by Mohaisen and Alrawi (2015).

Chapter 4

Results

Research Goals

The goal of this study was to develop an experimental prototype system to provide improved detection for polymorphic malware. Today's effective detection rate for polymorphic malware detection ranges from 68.75% to 81.25% (Amos et al., 2013). The prototype system developed performed various feature extraction and assembled the datasets. This study conducted various tests within testing SCED protocol. The test results were then quantitatively examined and evaluated detection rates for polymorphic malware. This study leveraged previous quantitative experiments for supervised machine learning for malware research (Boro, et al., 2012; Pradesh, 2014) to better understand how a limited set of multidimensional topological information can be used for malware detection.

Review of the Methodology

The single-subject experimental design provides a standard framework and has been widely accepted protocol for research questions posed for this study. The study followed the basic single-subject protocol throughout the study. The protocol follows the following steps.

1. Establish baseline test data – established a testing baseline for detection given a standard feature dataset with a standard clustering algorithm for polymorphic malware through multiple measurements before an intervention. The number of standard features included file properties, static analysis and dynamic analysis.

2. Manipulate feature set – the standard feature dataset was manipulated to evaluate whether detection rates improved or deteriorated after the intervention. The study examined three multidimensional datasets – one dedicated for each algorithm. The three datasets were generated by combining known malware, known benign and unknown samples into single feature dataset of 200,000 samples.

3. Controlled procedures and environment – the developed prototype environment and datasets were controlled to ensure the extracted feature dataset remained static over time and the automated feature extraction process remained unaltered (Rossow et al., 2012).

4. Standard measurement approaches – a standard measurement approach for documenting and capturing baseline testing data and subsequent test results. Detection measurements were conducted in the same fashion for all tests. The study established baseline detection rates for each of the three clustering algorithms. Standard measurements were documented to evaluate Accuracy (ACC), Correlation Coefficient (CC), True Positive Rate (sensitivity measure), and False Positive Rate (specificity measure). As specified by the standard protocol, these measurements were established as a permanent observational recording (Rossow et al., 2012).

5. Weighting of features – as part of this testing protocol the inputs were manipulated and assessed. Within a single dataset, features were removed based on an attribute selection algorithm (e.g. greedy stepwise) and then evaluated for each algorithm.

6. Capturing testing results - All test results were captured in terms of ACC, CC, True Positive Rate (TPR) and False Positive Rate (FPR).

7. Graphing results - All test results were graphed and presented in terms of ACC, CC, True Positive Rate (TPR) and False Positive Rate (FPR).

8. Evaluating results – All test results were evaluated after each test. Each test was analyzed with rigor to understand the ACC, CC, True Positive Rate (TPR) and False Positive Rate (FPR).

9. Test controls – All tests were conducted using the required test controls. Test controls regulated the test environment, test data, and test results.

The experimental design and approach enabled the researcher to conduct experiments in a staged and systematic manner. The standard testing protocol provided a means to plan, prepare, execute, document and analyze the test results. The testing protocol allowed the researcher to document the detection results for the various testing algorithms (Rossow et al., 2012). The results were collected in stages and results were evaluated for effectiveness at the end of each experiment (Rossow et al., 2012). The experimental research methodology was consistent with previous quantitative malware studies (Creswell, 2007). The study's methodology provides a means to introduce new measures and systematically evaluate test results prior to conducting additional testing. The quantitative experimental approach for conducting this study leveraged other testing techniques conducted in healthcare, drug trials and other medical studies (Rizvi & Nock, 2008).

Experimental Outcomes

The following test cases were executed as part of the SCED protocol for each algorithm (MB, IBk and DLMLP).

Test Case 1

As with all test cases, the SCED testing protocol was used to structure and guide the execution of the various experiments. Test Case 1 utilized Dataset 1 with the Metabagging (MB) cluster algorithm. Dataset 1 was generated through a pseudo-random selection of samples representing a stratified sample. The stratified sample contained three sample types: malware, benign and unknown. Dataset 1 was selected and generated on a percentage basis: Malware (40%), Benign (30%) and Unknown (30%). Dataset 1 was automatically selected and generated from a 2.5 million sample population. However, the generation of the final file required some manual intervention to generate the final Comma Separated Values (CSV) file. Once the CSV file was generated, Weka was used for data validation purpose and to convert the CSV file to ARFF file. Once completed, the dataset was found to be “clean” or validated. Dataset 1 was then saved into a controlled directory for validated ARFF files. Dataset 1 was validated prior to the beginning of any baseline testing. The 26 features were validated and included all attributes from file properties, static analysis and dynamic analysis.

1. Establish baseline test data – Dataset 1 was validated through processing the features through Weka. Upon file import, it was discovered that Dataset 1 had several features/attributes mistyped. The fields are typed as part of validated upon import of the CSV file into Weka. A number of numeric features (typically counts) were incorrectly typed or imported as nominal values. In order to use these features for detection these typing errors needed to be corrected. Errors introduced through automated parsing was fixed through searching and eliminating leading spaces and/or special characters embedded in the field. These errors were removed from the entire dataset and all features

were checked for leading spaces and embedded special characters. Validation was conducted on all 200,000 rows and 26 features. Once completed, the dataset was found to be “clean” or validated. Dataset 1 was then saved into a controlled directory for validated ARFF files. Dataset 1 was validated prior to the beginning of any baseline testing. The 26 features were validated and included all attributes from file properties, static analysis and dynamic analysis.

2. Manipulate feature set – Once the Dataset 1 was validated, Dataset 1 was used to produce three distinct datasets - Dataset 1-1, Dataset 1-2, and Dataset 1-3. These three datasets would be used to perform testing for the metabagging algorithm in various ways. Dataset 1-1 was used for MB Baseline experimental testing. Dataset 1-2 would be used for MB Reduced Features and Dataset 1-3 MB Information Gain. Dataset 1 was used as the basis for all MB training and test datasets.

Dataset 1 was then used to generate Dataset 1-1 (baseline – no manipulation). Dataset 1-1 was used as a “full-feature” dataset and all 26 features were used with the MB algorithm to establish a detection baseline. The baseline testing did not filter and did not use any feature analysis to improve any test results. Upon completion of baseline testing, the dataset was exposed to further feature analysis.

Dataset 1 was also used to generate Dataset 1-2 Reduced Features dataset. Dataset 1-2 was generated by analyzing the features in Dataset 1 to better understand relationships and possible feature interdependencies. The full Dataset 1 – was analyzed and all 26 features were processed by the Greedy Stepwise algorithm to understand data relationships and interdependencies. This algorithm analyzes various features from the dataset based on analyzing relationships that exist between features. Previous studies

have analyzed and discovered that features or attributes can have relationships of various types including irrelevant, weakly relevant or strongly relevant feature relationships (John, Kohavi, & Pfleger, 1994). The output of the Greedy Stepwise algorithm suggested that of the 26 original features within Dataset 1, could be reduced to 6 features/attributes. The Greedy Stepwise algorithm produced a .793 Merit of Subset value. The features selected by this algorithm included: 1) File Size, 2) File Type, 3) Compiler Type, 4) Library Calls, 5) Encryption, and the Op Code 6) XCHG. The next step was to reimport the entire Dataset 1 and remove extraneous features/attributes – this became Dataset 1-2. Dataset 1-2 removed all features except for File Size, File Type, Compiler Type, Library Calls, Encryption, and XCHG. Dataset 1-2 was then processed with the MB algorithm.

The third and final dataset (Dataset 1-3) was analyzed utilizing another feature analysis tool to evaluate features and relationships. The Information Gain algorithm seeks to amplify certain features within a dataset to potentially improve model outcomes (C. Lee & Lee, 2006). In this case, the intent was to use information gain to provide a reduced feature set that potentially increases the detection rate. The information gain algorithm was used to analyze the entire Dataset 1 – all 26 original features. The information gain algorithm suggested that the top seven attributes/features included: 1) Compiler Type (1.50537), 2) Malware Type (1.35976), 3) MOV (1.15211), 4) File Size (1.07808), 5) Encrypt Type (0.89771), 6) File Type (0.88218), and 7) Library Calls (0.79424). Dataset 1-3 used the full dataset and reduced the features to only the top seven features to potentially improve detection results. Therefore, Dataset 1-3 used these seven features for detection within MB. Further, Compiler Type could be used as a

leading detection indicator or attribute. Dataset 1-3 then used Compiler Type as a selected attribute and was then processed with the MB algorithm.

After running various experiments with Datasets 1-1, 1-2 and 1-3, it was determined that the results being produced delivered higher than anticipated results. After researching the issues, it was determined that the models were potentially experiencing training data saturation as noted by Zhu (2012). These higher than expected results produced were attributed to using a subset of Dataset 1 for training purposes. The initial training approach was to use a subset of the dataset for training the classifier with widely accepted 10 fold cross validation (Kohavi, 1995). This approach is a standard supervised learning practice and has been used many times to build the internal estimation training models within machine learning (Kohavi, 1995). Based on the previous research (Zhu, 2012), a new training dataset was generated and MB testing was repeated to evaluate testing outcomes.

Previous research has shown that datasets too closely related can produce higher than expected results (Zhu, 2012). Therefore, a new training dataset was generated similar to Dataset 1. The training dataset that was selected and generated in the same manner as Dataset 1. However, the training dataset was allowed to have overlapping samples from Dataset 1. Based on a re-run of the MB Baseline test, the newly generated training dataset seemed to correct the training set saturation issue. Based on the MB test results, additional training datasets were generated in a pseudo-random stratified fashion for use with the other two algorithms. Training Dataset 1 was used for MB testing. Training Dataset 2 and Training Dataset 3 were generated and used for the other cluster algorithm experiments.

3. Controlled procedures and environment – the developed prototype environment and datasets were controlled to ensure the extracted feature dataset remained static over time and the automated feature extraction process remained unaltered (Rossow et al., 2012). Dataset 1-1, Dataset1-2 and Dataset 1-3 remained unaltered during any and all testing. Additional precautions were taken to ensure that the Datasets 1-2 and 1-3 remained static after removal of extraneous features. All datasets were exported in ARFF format after validation and manipulation using naming standard conventions identifying the dataset.

4. Standard measurement approaches – a standard measurement approach was used to document and capture testing results. Test results for Baseline testing (Dataset 1-1), Reduced Feature Selection (Dataset 1-2), and Data Amplification (Dataset 1-3) were collected and captured. Detection measurements were conducted for all tests. Per the testing protocol, the study established a baseline detection rate for MB. Subsequent tests results were captured and analyzed. Results were analyzed using standard measurements including Accuracy (ACC), Correlation Coefficient (CC), True Positive Rate (sensitivity measure), and False Positive Rate (specificity measure). As specified by the standard protocol, these measurements were captured as a permanent observational recording (Rossow et al., 2012).

5. Weighting of features – as part of this testing protocol the inputs were manipulated and assessed. Using the baseline dataset (Dataset 1-1), features were removed based on an attribute selection algorithm (e.g. greedy stepwise) and then evaluated for MB. Likewise, using the baseline dataset (Dataset 1-1), features were removed based on the information gain for data amplification. The three tests for MB

included MB Baseline (Dataset 1-1), MB Reduced Feature Selection (Dataset 1-2), and MB Data Amplification (Dataset 1-3).

6. Capturing testing results - All results were captured in terms test and measurement. Each test (MB Baseline, MB Reduced Feature Selection, and MB Data Amplification) was captured with the associated measurement - ACC, CC, True Positive Rate (TPR) and False Positive Rate (FPR). The results are provided below:

Table 6 MB Baseline Classification Results

Experimental Results		
MB Baseline	Dataset 1-1	
Correctly Classified Instances	198,866	99.433%
Incorrectly Classified Instances	1,134	0.567%

Table 7 MB Baseline Experimental Results

MB Baseline	
Measurement	Result
True Positive Rate (TPR)	0.99983
True Negative Rate (TNR)	0.99983
False Positive Rate (FPR)	0.00017
False Negative Rate (FNR)	0.00017
Accuracy (ACC)	0.99983
Correlation Coefficient (CC)	0.00001

Table 8 MB Reduced Feature Classification Results

Experimental Results		
MB Reduced Feature Selection	Dataset 1-2	
Correctly Classified Instances	199,994	99.997%
Incorrectly Classified Instances	6	0.003%

Table 9 MB Reduced Feature Selection Experimental Results

MB Reduced Feature Selection	
Measurement	Result
True Positive Rate (TPR)	0.99991
True Negative Rate (TNR)	0.99995
False Positive Rate (FPR)	0.00005
False Negative Rate (FNR)	0.00009
Accuracy (ACC)	0.99992
Correlation Coefficient (CC)	0.00001

Table 10 MB Data Amplification Classification Results

Experimental Results		
MB Data Amplification	Dataset 1-3	
Correctly Classified Instances	199,997	99.9985%
Incorrectly Classified Instances	3	0.0015%

Table 11 MB Data Amplification Classification Experimental Results

MB Data Amplification	
Measurement	Result
True Positive Rate (TPR)	0.99999
True Negative Rate (TNR)	0.99998
False Positive Rate (FPR)	0.00002
False Negative Rate (FNR)	0.00001
Accuracy (ACC)	0.99998
Correlation Coefficient (CC)	0.00001

7. Graphing results - All test results were graphed and presented in terms test (MB Baseline, MB Reduced Feature Selection, and MB Data Amplification) and measurement for True Positive Rate (TPR) and False Positive Rate (FPR) are shown below. Accuracy essentially followed the same trendline as TPR. CC as a constant .0001 for all three experiments MB Benchmark, MB Reduced Features and MB Amplified Features.

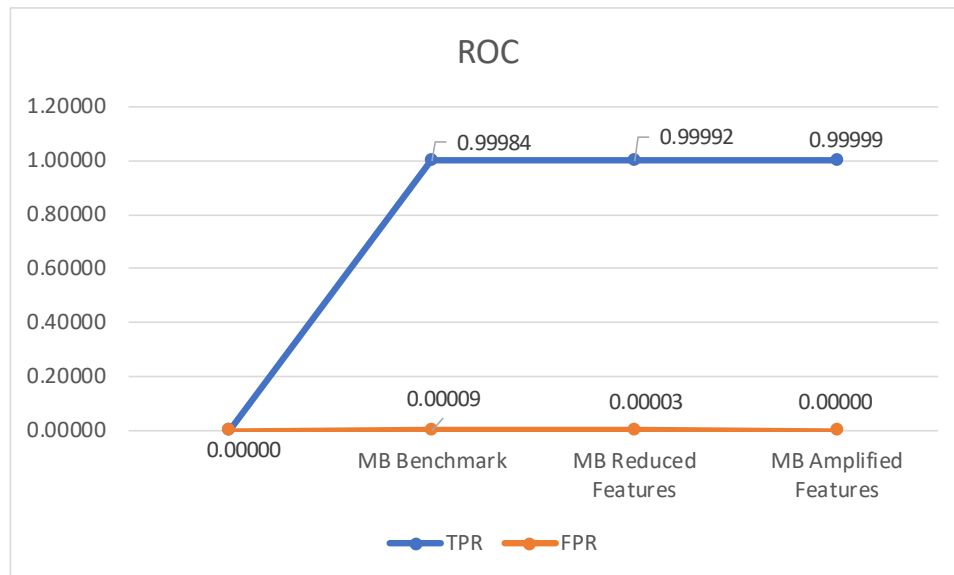


Figure 5. MB Graph Results

8. Evaluating results – All test results were evaluated after each test. Each test was analyzed with rigor to understand the ACC, CC, True Positive Rate (TPR) and False Positive Rate (FPR).

Table 12 MB Experimental Results

<i>Measurement</i>	MB Baseline	MB Reduced Feature Selection	MB Data Amplification
True Positive Rate (TPR)	0.99983	0.99991	0.99999
True Negative Rate (TNR)	0.99983	0.99995	0.99998
False Positive Rate (FPR)	0.00017	0.00005	0.00002
False Negative Rate (FNR)	0.00017	0.00009	0.00001
Accuracy (ACC)	0.99983	0.99992	0.99998
Correlation Coefficient (CC)	0.00001	0.00001	0.00001

9. Test controls – All tests were conducted using the required test controls. Test controls regulated the MB test environment, MB test data (Training Datasets 1-1, 1-2, 1-3 and Test Datasets 1-1, 1-2, 1-3), and test results (MB Baseline, MB Reduced Feature Selection and MB Data Amplification).

Test Case 2

As with all test cases, the SCED testing protocol was used to structure the experiment. Test Case 2 utilized Dataset 2 with the IBk Nearest Neighbor clustering algorithm (IBk). The Instance Based k-Nearest Neighbor (IBk) algorithm was chosen to replace LSH as the algorithm improves upon the “locality” aspects of the clustering algorithm. The IBk is a k-nearest-neighbor classifier that utilizes a similar distance metric

used in the LSH algorithm. The calculated Euclidean distance function is used by IBk as a critical search parameter within the algorithm (Manikandan et al., 2016). Dataset 2 was generated through a pseudo-random selection of samples representing all three types of samples: malware, benign and unknown. Dataset 2 was automatically generated based on the same random selection algorithm that generated Dataset 1. Dataset 2 was selected and generated on a percentage basis: Malware (40%), Benign (30%) and Unknown (30%). Dataset 2 was automatically selected and generated from a 2.5 million sample population. However, the generation of the final file required some manual intervention to generate the final Comma Separated Values (CSV) file. Once the CSV file was generated, Weka was used for data validation purpose and to convert the CSV file to ARFF file. Once completed, the dataset was found to be “clean” or validated. Dataset 2 was then saved into a controlled directory for validated ARFF files. Dataset 2 was validated prior to the beginning of any baseline testing. The 26 features were validated and included all attributes from file properties, static analysis and dynamic analysis.

1. Establish baseline test data – Like Dataset 1, Dataset 2 was validated through processing the features through Weka. As was seen in Dataset 1, several of the numeric features within Dataset 2 were typed incorrectly as nominal. These errors were highlighted upon importing the Dataset 2 CSV file into Weka. Data validation was conducted and data errors were corrected in exactly the same fashion as Dataset 1. Again, these errors were introduced into the dataset through automated parsing during the extraction process. These errors were fixed through searching and eliminating leading spaces and/or special characters embedded in the field. These errors were removed from the entire dataset and all features were validated. Once completed, the dataset was found

to be “clean” or validated. Per the testing protocol, the dataset was validated prior to the beginning of baseline testing. The features (26) included attributes from file properties, static analysis and dynamic analysis. As noted above, a Training Dataset 2 was generated, validated and used as part of the classifier training protocol.

2. Manipulate feature set – The manipulation of the feature set produced three test datasets – Dataset 2-1, Dataset 2-2, and Dataset 2-3. Similarly, the training dataset was developed from the Training Dataset 2 for each test. After both datasets (training and test) were validated, Training Dataset 2-1 and Test Dataset 2-1 (baseline – no manipulation) were processed with all 26 features using the IBk algorithm to establish a detection baseline. The baseline testing did not filter and did not use feature any analysis to improve test results. Upon completion of baseline testing, the dataset was exposed to further feature analysis.

The second set of datasets – Training Dataset 2-2 and Test Dataset 2-2 were generated by analyzing the features to better understand relationships and possible feature interdependencies. The full Dataset 2-1 – all 26 features were analyzed with the Greedy Stepwise algorithm. The output of the Greedy Stepwise algorithm suggested that of the 26 original features, the dataset could be reduced to 7 features/attributes. The Greedy Stepwise algorithm produced a Merit of best subset of 0.896. The predictive attributes for this dataset included 1) File Size, 2) File Type, 3) Compiler, 4) Compiler Type, 5) Library Calls, 6) Encryption and the Op Code 7) XCHG. The next step was to reimport the both the Training Dataset 2 and Dataset 2 and remove extraneous features/attributes. The classifier training was conducted using Training Dataset 2-2. Experimental testing was conducted using Dataset 2-2. In both cases, Training Dataset 2-2 and Test Dataset 2-

2 removed all features except for File Size, File Type, Compiler, Compiler Type, Library Calls, Encryption, and XCHG. Dataset 2-2 was then processed with the IBk algorithm.

The third and final dataset (Dataset 2-3) was analyzed utilizing another feature analysis tool to evaluate features and relationships for Dataset 2. The Information Gain algorithm seeks to amplify certain features within a dataset to potentially improve model outcomes (C. Lee & Lee, 2006). In this case, the intent was to use information gain to provide a reduced feature set that potentially increases the detection rate. The information gain algorithm was used to analyze the entire Dataset 2 – all 26 original features. In this case, the intent was to use information gain to provide a reduced feature set that potentially increases the detection rate. The information gain algorithm suggested that the top seven attributes/features included 1) Compiler Type (1.55744), 2) Malware Type (1.40863), 3) XCHG (1.26408), 4) File Size (1.09264), 5) Encrypt Type (0.88897), 6) File Type (0.86818), and 7) Library Calls (0.77946). Dataset 2-3 used the full dataset and reduced the features to only the top seven features to potentially improve detection results. Therefore, Dataset 2-3 used these seven features for detection within the IBk classifier. Further, Compiler Type could be used as a leading indicator attribute. Dataset 2-3 then used Compiler Type with the IBk classifier to deliver experimental results.

3. Controlled procedures and environment – the developed prototype environment and datasets were controlled to ensure the extracted feature dataset remained static over time and the automated feature extraction process remained unaltered (Rossow et al., 2012). Training Datasets (Training Dataset 2-1, Training Dataset 2-2 and Training Dataset 2-3) and Test Datasets (Test Dataset 2-1, Test Dataset 2-2 and Test Dataset 2-3)

remained unaltered during any and all testing. Additional precautions were taken to ensure that the Test and Training Datasets 2-2 and 2-3 remained static after removal of extraneous features. All datasets were exported in ARFF format after validation and manipulation using naming standard conventions identifying the dataset.

4. Standard measurement approaches – a standard measurement approach was used to document and capture testing results. Test results for Baseline testing (Dataset 2-1), Reduced Feature Selection (Dataset 2-2), and Data Amplification (Dataset 2-3) were collected and captured. Detection measurements were conducted for all tests. Per the testing protocol, the study established a baseline detection rate for IBk. Subsequent tests results were captured and analyzed. Results were analyzed using standard measurements including Accuracy (ACC), Correlation Coefficient (CC), True Positive Rate (sensitivity measure), and False Positive Rate (specificity measure). As specified by the standard protocol, these measurements were captured as a permanent observational recording (Rossow et al., 2012).

5. Weighting of features – as part of this testing protocol the inputs were manipulated and assessed. Using the baseline dataset (Dataset 2-1), features were removed based on an attribute selection algorithm (e.g. greedy stepwise) and then evaluated for IBk. Likewise, using the baseline dataset (Dataset 2-1), features were included IBk Baseline (Dataset 2-1), IBk Reduced Feature Selection (Dataset 2-2), and IBk Data Amplification (Dataset 2-3).

6. Capturing testing results - All results were captured in terms test and measurement. Each test (IBk Baseline, IBk Reduced Feature Selection, and IBk Data

Amplification) was captured with the associated measurement - ACC, CC, True Positive Rate (TPR) and False Positive Rate (FPR).

Table 13 IBk Baseline Classification Results

Experimental Results		
IBk Baseline	Dataset 2-1	
Correctly Classified Instances	199,872	99.936%
Incorrectly Classified Instances	128	0.064%

Table 14 IBk Baseline Experimental Results

IBk Baseline	
Measurement	Result
True Positive Rate (TPR)	0.99984
True Negative Rate (TNR)	0.99991
False Positive Rate (FPR)	0.00009
False Negative Rate (FNR)	0.00016
Accuracy (ACC)	0.99986
Correlation Coefficient (CC)	0.00001

Table 15 IBk Reduced Feature Selection Classification Results

Experimental Results		
IBk Reduced Feature Selection	Dataset 2-2	
Correctly Classified Instances	199,923	99.962%
Incorrectly Classified Instances	77	0.038%

Table 16 IBk Reduced Feature Selection Experimental Results

IBk Reduced Feature Selection	
Measurement	Result
True Positive Rate (TPR)	0.99992
True Negative Rate (TNR)	0.99997
False Positive Rate (FPR)	0.00003
False Negative Rate (FNR)	0.00008
Accuracy (ACC)	0.99993
Correlation Coefficient (CC)	0.00001

Table 17 IBk Data Amplification Classification Results

Experimental Results		
IBk Data Amplification	Dataset 2-3	
Correctly Classified Instances	200,000	100.0000%
Incorrectly Classified Instances	0	0.0000%

Table 18 IBk Data Amplification Experimental Results

IBk Data Amplification	
Measurement	Result
True Positive Rate (TPR)	0.99999
True Negative Rate (TNR)	1.00000
False Positive Rate (FPR)	0.00000
False Negative Rate (FNR)	0.00001
Accuracy (ACC)	0.99999
Correlation Coefficient (CC)	0.00001

7. Graphing results - All test results were graphed and presented in terms of test (IBK Baseline, IBK Reduced Feature Selection, and IBK Data Amplification) and measurement for True Positive Rate (TPR) and False Positive Rate (FPR). The results are shown below. Accuracy essentially followed the same trendline as TPR. CC was a

constant of .0001 across all three experiments IBk Benchmark, IBk Reduced Features and IBk Amplified Features.

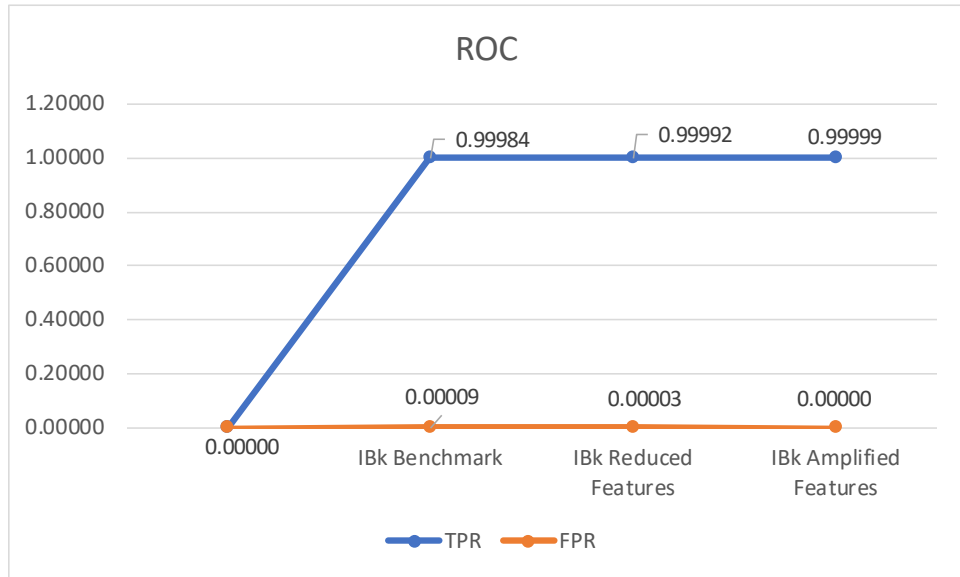


Figure 6. IBk Graph Results

8. Evaluating results – All test results were evaluated after each test. Each test was analyzed with rigor to understand the ACC, CC, True Positive Rate (TPR) and False Positive Rate (FPR).

Table 19 IBk Experimental Results

<i>Measurement</i>	IBK Baseline	IBk Reduced Feature Selection	IBk Data Amplification
True Positive Rate (TPR)	0.99984	0.99992	0.99999
True Negative Rate (TNR)	0.99991	0.99997	1.00000
False Positive Rate (FPR)	0.00009	0.00003	0.00000
False Negative Rate (FNR)	0.00016	0.00008	0.00001
Accuracy (ACC)	0.99986	0.99993	0.99999
Correlation Coefficient (CC)	0.00001	0.00001	0.00001

9. Test controls – All tests were conducted using the required test controls. Test controls regulated the IBk test environment, IBk test data (Training Datasets 1-1, 1-2, 1-3 and Test Datasets 1-1, 1-2, 1-3), and test results (IBk Baseline, IBk Reduced Feature Selection and IBk Data Amplification).

Test Case 3

As with all test cases, the SCED testing protocol was used to structure the experiment. Test Case 3 utilized Dataset 3 with the Deep Learning Multilevel Perceptron (DLMLP) clustering algorithm. The DLMLP algorithm was chosen to replace the BP algorithm as it improves upon the dimensions of belief for training and test datasets. The main idea in replacing the BP algorithm with DLMLP was recently presented by Gruber et al. (2017). ANN's can leverage belief propagation for clustering or classification inefficiently (Gruber et al., 2017). ANN's build networks of neurons, share information between neurons and propagate results throughout the entire network (Gruber et al.,

2017). Gruber (2017) proposed using shallow neural networks with deep learning to achieve similar or better propagation results. The deep learning approach established shallow neural networks and combined other algorithms such as MLP to achieve similar propagation (Gruber et al., 2017). The goal was to utilize the DLMLP algorithm for improved detection. Given that the goal of this research was to establish new detection using advanced algorithms. DLMLP was selected for model efficiencies and to leverage nascent algorithms that improved upon other algorithms. Dataset 3 was generated through a pseudo-random selection of samples representing all three types of samples: malware, benign and unknown. Dataset 3 was automatically generated based on the same random selection algorithm that generated other datasets. Dataset 3 was selected and generated on a percentage basis: Malware (40%), Benign (30%) and Unknown (30%). Dataset 3 was automatically selected and generated from a 2.5 million sample population. However, the generation of the final file required some manual intervention to generate the final Comma Separated Values (CSV) file. Once the CSV file was generated, Weka was used for data validation purpose and to convert the CSV file to ARFF file. Once completed, the dataset was found to be “clean” or validated. Dataset 3 was then saved into a controlled directory for validated ARFF files. Dataset 3 was validated prior to the beginning of any baseline testing. The 26 features were validated and included all attributes from file properties, static analysis and dynamic analysis.

1. Establish baseline test data – Like previous datasets, Dataset 3 was validated through processing the features through Weka. As was seen in previous datasets, several of the numeric features within Dataset 3 were typed incorrectly as nominal. These errors were highlighted upon importing the Dataset 3 CSV file into Weka. Data validation was

conducted and data errors were corrected in exactly the same fashion as previous datasets. Again, these errors were introduced into the dataset through automated parsing during the extraction process. These errors were fixed through searching and eliminating leading spaces and/or special characters embedded in the field. These errors were removed from the entire dataset and all features were validated. Once completed, the dataset was found to be “clean” or validated. Per the testing protocol, the dataset was validated prior to the beginning of baseline testing. The features (26) included attributes from file properties, static analysis and dynamic analysis. As noted above, a Training Dataset 3 was generated, validated and used as part of the classifier training protocol.

2. Manipulate feature set – The manipulation of the feature set produced three test datasets – Dataset 3-1, Dataset 3-2, and Dataset 3-3. Similarly, the training dataset was developed from the Training Dataset 3 for each test. After both datasets (training and test) were validated, Training Dataset 3-1 and Test Dataset 3-1 (baseline – no manipulation) were processed with all 26 features using the DLMLP algorithm to establish a detection baseline. The baseline testing did not filter and did not use feature any analysis to improve test results. Upon completion of baseline testing, the dataset was exposed to further feature analysis.

The second set of datasets – Training Dataset 3-2 and Test Dataset 3-2 were generated by analyzing the features to better understand relationships and possible feature interdependencies. The full Test Dataset 3-1 – all 26 features were analyzed with the Greedy Stepwise algorithm. The output of the Greedy Stepwise algorithm suggested that of the 26 original features, the dataset could be reduced to 7 features/attributes. The Greedy Stepwise algorithm produced a Merit of best subset of 0.807. The predictive

attributes for this dataset included 1) File Size, 2) File Type, 3) Compiler, 4) Compiler Type, 5) Library Calls, 6) Encryption and the Op Code 7) XCHG. The next step was to reimport the both the Training Dataset 3 and Test Dataset 3 and remove extraneous features/attributes. The classifier training was conducted using Training Dataset 3-2. Experimental testing was conducted using Dataset 3-2. In both cases, Training Dataset 3-2 and Test Dataset 3-2 removed all features except for File Size, File Type, Compiler, Compiler Type, Library Calls, Encryption, and XCHG. Dataset 2-2 was then processed with the DLMLP algorithm.

The third and final dataset (Training and Test Dataset 3-3) was analyzed utilizing another feature analysis tool to evaluate features and relationships for Dataset 3. The Information Gain algorithm seeks to amplify certain features within a dataset to potentially improve model outcomes (C. Lee & Lee, 2006). In this case, the intent was to use information gain to provide a reduced feature set that potentially increases the detection rate. The information gain algorithm was used to analyze the entire Dataset 3 – all 26 original features. In this case, the intent was to use information gain to provide a reduced feature set that potentially increases the detection rate. The information gain algorithm suggested that the top seven attributes/features included 1) Compiler Type (1.69143), 2) Malware Type (1.51408), 3) XCHG (1.37264), 4) File Size (1.10192), 5) Encrypt Type (0.91679), 6) File Type (0.78658), and 7) Library Calls (0.71794). Dataset 3-3 used the full dataset and reduced the features to only the top seven features to potentially improve detection results. Therefore, Dataset 3-3 used these seven features for detection within the DLMLP classifier. Further, Compiler Type could be used as a

leading indicator attribute. Dataset 3-3 then used Compiler Type with the DLMLP classifier to deliver experimental results.

3. Controlled procedures and environment – the developed prototype environment and datasets were controlled to ensure the extracted feature dataset remained static over time and the automated feature extraction process remained unaltered (Rossow et al., 2012). Training Datasets (Training Dataset 3-1, Training Dataset 3-2 and Training Dataset 3-3) and Test Datasets (Test Dataset 3-1, Test Dataset 3-2 and Test Dataset 3-3) remained unaltered during any and all testing. Additional precautions were taken to ensure that the Test and Training Datasets 3-2 and 3-3 remained static after removal of extraneous features. All datasets were exported in ARFF format after validation and manipulation using naming standard conventions identifying the dataset.

4. Standard measurement approaches – a standard measurement approach was used to document and capture testing results. Test results for Baseline testing (Dataset 3-1), Reduced Feature Selection (Dataset 3-2), and Data Amplification (Dataset 3-3) were collected and captured. Detection measurements were conducted for all tests. Per the testing protocol, the study established a baseline detection rate for the DLMLP classifier. Subsequent tests results were captured and analyzed. Results were analyzed using standard measurements including Accuracy (ACC), Correlation Coefficient (CC), True Positive Rate (sensitivity measure), and False Positive Rate (specificity measure). As specified by the standard protocol, these measurements were captured as a permanent observational recording (Rossow et al., 2012).

5. Weighting of features – as part of this testing protocol the inputs were manipulated and assessed. Using the baseline dataset (Dataset 3-1), features were

removed based on an attribute selection algorithm (e.g. greedy stepwise) and then evaluated for DLMLP. Likewise, using the baseline dataset (Dataset 3-1), features were removed based on the information gain for data amplification. The three tests for DLMLP included DLMLP Baseline (Dataset 3-1), DLMLP Reduced Feature Selection (Dataset 3-2), and DLMLP Data Amplification (Dataset 3-3).

6. Capturing testing results - All results were captured in terms test and measurement. Each test (DLMLP Baseline, DLMLP Reduced Feature Selection, and DLMLP Data Amplification) was captured with the associated measurement - ACC, CC, True Positive Rate (TPR) and False Positive Rate (FPR).

Table 20 DLMLP Baseline Classification Results

Experimental Results		
DLMLP Baseline	Dataset 3-1	
Correctly Classified Instances	199,903	99.952%
Incorrectly Classified Instances	97	0.048%

Table 21 DLMLP Baseline Experimental Results

DLMLP Baseline	
Measurement	Result
True Positive Rate (TPR)	0.99994
True Negative Rate (TNR)	0.99995
False Positive Rate (FPR)	0.00005
False Negative Rate (FNR)	0.00006
Accuracy (ACC)	0.99994
Correlation Coefficient (CC)	0.00001

Table 22 DLMLP Reduced Feature Selection Classification Results

Experimental Results		
DLMLP Reduced Feature Selection	Dataset 3-2	
Correctly Classified Instances	199,996	99.998%
Incorrectly Classified Instances	4	0.002%

Table 23 DLMLP Reduced Feature Selection Experimental Results

DLMLP Reduced Feature Selection	
Measurement	Result
True Positive Rate (TPR)	0.99999
True Negative Rate (TNR)	0.99998
False Positive Rate (FPR)	0.00002
False Negative Rate (FNR)	0.00001
Accuracy (ACC)	0.99999
Correlation Coefficient (CC)	0.00001

Table 24 DLMLP Data Amplification Classification Results

Experimental Results		
DLMLP Data Amplification	Dataset 3-3	
Correctly Classified Instances	199,999	99.9995%
Incorrectly Classified Instances	1	0.0005%

Table 25 DLMLP Data Amplification Experimental Results

DLMLP Data Amplification	
Measurement	Result
True Positive Rate (TPR)	0.99999
True Negative Rate (TNR)	1.00000
False Positive Rate (FPR)	0.00000
False Negative Rate (FNR)	0.00001
Accuracy (ACC)	0.99999
Correlation Coefficient (CC)	0.00001

7. Graphing results - All test results were graphed and presented in terms test (DLMLP Baseline, DLMLP Reduced Feature Selection, and DLMLP Data Amplification) and measurement for True Positive Rate (TPR) and False Positive Rate (FPR). The results are shown below. Accuracy essentially followed the same trendline as TPR and was not graphed for that reason. CC was a constant of .0001 across all three experiments DLMLP Benchmark, DLMLP Reduced Features and DLMLP Amplified Features and did not provide substantial value.

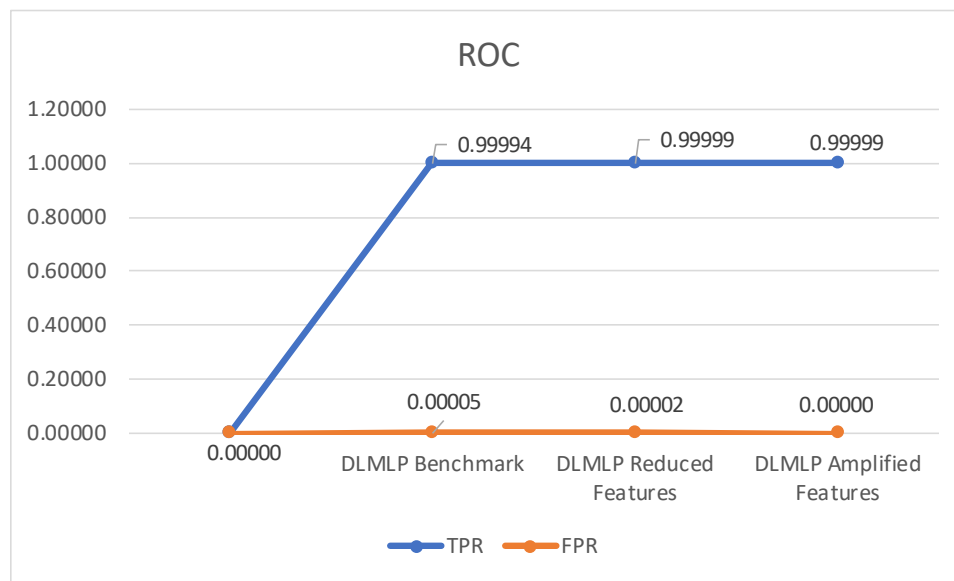


Figure 7. DLMLP Graph Results

8. Evaluating results – All test results were evaluated after each test. Each test was analyzed with rigor to understand the ACC, CC, True Positive Rate (TPR) and False Positive Rate (FPR).

Table 26 DLMLP Experimental Results

<i>Measurement</i>	DLMLP Baseline	DLMLP Reduced Feature Selection	DLMLP Data Amplification
True Positive Rate (TPR)	0.99994	0.99999	0.99999
True Negative Rate (TNR)	0.99995	0.99998	1.00000
False Positive Rate (FPR)	0.00005	0.00002	0.00000
False Negative Rate (FNR)	0.00006	0.00001	0.00001
Accuracy (ACC)	0.99994	0.99999	0.99999
Correlation Coefficient (CC)	0.00001	0.00001	0.00001

9. Test controls – All tests were conducted using the required test controls. Test controls regulated the DLMLP test environment, DLMLP test data (Training Datasets 3-1, 3-2, 3-3 and Test Datasets 3-1, 3-2, 3-3), and test results (DLMLP Baseline, DLMLP Reduced Feature Selection and DLMLP Data Amplification).

Data Analysis

The purpose of this study was to evaluate the effective malware detection rates using multidimensional topological features with advanced cluster algorithms. Previous research (Fraley & Figueroa, 2016) provided some insight into detection rates using multidimensional topological features with using advanced clustering algorithms.

However, previous research was limited to a few thousand files/samples. This study greatly expanded both the training and test datasets. Having larger datasets provided deeper insight into how each of the various algorithms perform given a limited set of topological features. This study also provided insight into how algorithms perform given a full (26) and optimized features (6 or 7). Experiments were performed with full data, reduced features and amplified features. This process provided an understanding of how detection performed given some algorithm tuning. This study established baselines for all three cluster algorithms (MB, IBk and DLMLP). The MB classifier out performed other algorithms (LSH and BP) in previous studies (Fraley & Figueroa, 2016). However, given the larger dataset, both the IBk and DLMLP classifiers out performed MB in almost every respect.

This study also provided a test case for detecting malware using multidimensional topological data. The baseline testing included processing the full set of features that included file properties, static analysis and dynamic analysis. The features include File Size, File Type, Malware Type, VM Aware, Compiler, Embedded Compiler, Compiler Type, Library Import/Export, Library Call, Encryption, Encrypt Type, and Op codes – ADD, AND, INC, LEA, MOV, NEG, OR, SUB, XOR, POP, JMP and XCHG. The testing from this study supports the concept that effective malware detection can be achieved using multidimensional topological data. Further, experimental testing has shown that a reduced feature set (6 or 7) delivered better and more effective detection rates than the full data set for this dataset. The features that delivered optimal detection rates include: 1) File Size, 2) File Type, 3) Compiler Type, 4) Library Calls, 5) Encryption, 6) Malware Type and the Op Codes 6) XCHG or MOV.

In terms of algorithm performance, IBk and DLMLP delivered impressive true positive test results given this dataset. The IBk classifier scored almost a perfect detection rate with the Amplified Feature training and test dataset (Dataset 2-3). The IBk true positive detection rate (.99999) has been validated and repeated several times to ensure the results. This far exceeded the initial study expectations. In addition, the study considered training dataset saturation and developed separate training datasets in order to not artificially increase detection rates. The near perfect detection rate should also validate the selection of this algorithm for replacing LSH for the study.

Evaluating algorithm performance in terms of false positive rates, both IBk Amplified Data and DLMLP Amplified Feature delivered impressive results. The false positive rates for both of these classifiers was near zero. It is difficult to improve false positive rates of zero. Again, several tests were repeated in order to validate these rates for both IBk and DLMLP.

It should be noted that both the IBk and DLMLP algorithms are expensive in terms of observed processing time. The MB algorithm would typically process and deliver test results (200,000 samples) in under 240 seconds. The IBk and DLMLP by contrast would process and deliver results in excess of 2,800 and 10,000 seconds respectively. The infrastructure was not tuned for processing any algorithm. Memory allocation and processor allocations remained constant and unchanged throughout the testing process.

Lastly, ACC and CC were not discriminators for the various algorithms. It was originally thought that ACC and CC would provide greater insight into algorithm performance over and above true positive and false positive rates. However, because of

the high true positive and low false positive rates, the evaluation of algorithms based on ACC and CC was not as useful as anticipated.

Findings

There were a number of substantial findings from this study. The study does answer the two basic research questions posed previously.

- 1) Can detection rates be improved by increasing the quality and quantity of multidimensional features for the machine learning advanced clustering algorithms from file properties, static and dynamic analysis?

Research findings suggest that reduced multidimensional topological feature set using file features, static and dynamic analysis delivered better overall results than the full dataset (7 feature vs 26 features). The reduced and amplified datasets delivered better results. While all three algorithms delivered above target results, less or selected features provided better detection results (99.99%) across all three algorithms.

- 2) Which of the machine learning advanced clustering algorithms performed better given the multidimensional features from file properties, static and dynamic analysis?

It should be stated that the experiments conducted for this study demonstrated that multidimensional topological data can be used to improve malware detection. The 26 selected features did help to establish and improve overall detection rates. The experimental approach demonstrated impressive detection rates of 99.99% (MB, IBk and DLMLP) for all experimental datasets. The lowest detection rate delivered in experimental testing was 99.43% (MB Baseline). The experimental detection rates

produced delivered far above the 81.25% target for polymorphic malware (Amos et al., 2013).

It should be noted that this study was concerned with recent malware (within the previous 18 months) and used a malware dataset specifically targeting Microsoft operating systems. Lastly, feature selection and data amplification both reduced the number of features processed by the algorithm. In performing the research for both the IBk and DLMLP algorithms, the theoretical aspects of both algorithms should perform well with an abundance of data. It was thought that the 26 multidimensional features would provide sufficient data features to enhance classifier performance. However, both algorithms performed incredibly well given a reduced and amplified dataset. Both produced near perfect true positive rates (99.99%) and near zero false positive rates.

Summary of Findings

The goal of this study was to develop an experimental approach for improved detection for polymorphic malware. The experimental detection rates delivered for this study were as high as 99.99% and the low was 99.43%. These experimental testing results far exceeded the current detection rate of 81.25% for polymorphic malware (Amos et al., 2013). The hypothesis for the study was that detection could be achieved by combining file properties, static and dynamic analysis features. Experimental testing with this dataset delivered effective detection using file properties, static and dynamic analysis features. This study leveraged the SCED protocol for performing various experimental tests. The test results were then quantitatively examined and evaluated. This study leveraged previous quantitative experiments for supervised machine learning

for malware research (Boro, et al., 2012; Pradesh, 2014) to better understand how multidimensional topological information can be utilized for malware detection.

The study's target was to develop an approach that would deliver an effective detection rate of higher than 81.25% using multidimensional topological data for malware detection. The prototype system performed various feature extraction and assembled the datasets for training and testing. The study sought to establish baselines for three advanced cluster algorithms (MB, IBk and DLMLP) and then manipulate feature weighting to exceed the target effective rate. In all cases (MB, IBk and DLMLP) the feature weighting delivered better detection performance.

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Conclusions

There were a number conclusions reached in for this study. The study demonstrated that multidimensional topological data can be effectively used for detection. Secondly, the study demonstrated that the various algorithms used delivered solid detection results using multidimensional topological data. The study also demonstrated that high malware detection rates can be achieved using reduced or amplified multidimensional topological features. A brief explanation for each conclusion is provided below.

This study produced experimental evidence that multidimensional topological data can be used for improved malware detection. The initial premise for the study was that by combining multidimensional features from file type, static and dynamic analysis with machine learning effective malware detection rates could be improved. Experimental testing with three algorithms successfully demonstrated detection rates greater than 99.99% with the experimental datasets. In addition, the algorithms did not suffer from an increase in false positive rates which is usually the case with algorithms that produce high detection rates. Given the high true positive rates and low false positive rates it can be safely stated that multidimensional topological data can be used to improve detection rates.

The study also produced highly effective malware detection rates by leveraging the advanced algorithms (MB, IBk and DLMLP). All three algorithms produced and

delivered effective detection results given the experimental datasets: MB (99.985%), IBk (100%) and DLMLP (99.9995%). All three algorithms produced and delivered low false positive rates given the experimental datasets: MB (0.00002%), IBk (0.0%) and DLMLP (0.0%). It can be safely concluded that these algorithms produced effective detection for the experimental datasets.

The study also produced evidence that very high malware detection rates can be achieved with reduced or amplified features. IBk and DLMLP both produced 99.999% effective detection rates with using a minimum number of features (6 or 7). Both the IBk and DLMLP algorithms are designed to process volumes of data. It was assumed that these algorithms would produce better results with more data. The opposite was found to be true for this study. Both algorithms produced better effective detection rates with the reduced features or amplified dataset. In addition, neither algorithm suffered from increased in false positive rate which normally is associated with algorithms that deliver higher detection rates. Therefore, it can be safely concluded that given the high true positive rate and the low false positive rate for both IBk and DLMLP it can be safely concluded that high malware detection rates can be achieved with reduced or amplified features leveraging multidimensional topological data with advanced algorithms.

As seen above, there are three valuable conclusions reached for this study. The conclusions reached for this study include leveraging multidimensional topological for detection, utilizing advanced algorithms to improve detection, and using reduced or amplified features produced higher than anticipated detection rates. These three conclusions are supported by both the experimental protocol and test results from various experiments presented earlier.

Implications

There are two major implications that can be gleaned from this research. First, malware detection can be accurately detected using multidimensional topological data. This means that a combination of file properties, static and dynamic analysis did provide impressive detection rates for this study. The implication is that detection for polymorphic malware targeting Microsoft platforms can be readily achieved through extracting multidimensional features and subjecting those features to a clustering algorithm. This implication may extend or augment current endpoint detection techniques and/or approaches.

The second implication is that endpoint protection could be greatly improved using reduced or amplified multidimensional topological features. The experimental results with these datasets demonstrate that by using a few features (reduced or amplified features) with advanced algorithms did produce impressive malware detection rates – approaching 100%. Further, the implication of using only a few features (6 or 7) for detection purposes for executable files may change or enhance how endpoint protection is currently performed. Endpoint detection for malware could embed a feature extraction and machine learning modules to process minimal features.

These two major implications for malware detection may change endpoint strategies. Machine learning is being incorporated into many commercial products. The study demonstrates that endpoint protection could and should use multidimensional data for advanced protection. Secondly, endpoint protection could use a few or minimal number of features (6 or 7) for detection purposes. The study also illustrates the case that reduced features or amplified features could dramatically improve malware detection at

the endpoint. These two implications could easily improve or enhance the way endpoint detection is conducted today across the enterprise.

Recommendations

There are three areas in which additional research is recommended. Two areas deal with an understanding the various functions or opcodes inherent in the executable. This study extracted and counted the associated opcode for each sample through a dynamic analysis process. For each sample, all opcodes were extracted and counted on an execution basis. This study focused on Bilar's (2007) top twelve Op Codes. The total universe of opcodes is approximately 398 depending on the code set. Just as Bilar (2007) observed, there were samples that executed a high number of opcodes. However, in some cases the number of instances a particular opcode was executed was incredibly high in comparison to other samples. As an example, one sample executed ADD nearly 10,000 times. Likewise, other opcodes such as LEA was executed over 7,000 times. The average for most opcodes was under 1,000. While not the focus of this study, further research could evaluate why the execution of some or all opcodes were performed so many times. As it turns out, the particular sample in question was indeed malware. Additional research needs to be conducted to further understand whether this is typical for poorly written malware thus making it noisy or is it something else more menacing like performing reconnaissance activities for detection at the endpoint. It is interesting from a research point of view as to why these opcodes are executed so many times.

Similarly, additional research needs to be conducted for two specific Op Codes: MOV and XCHG. This study focused Bilar's top 12 Op Codes. Bilar's research demonstrated that these 12 Op Codes represent 95% of the malicious Op Codes. Bilar's

(2007) research profiled ADD, AND, INC, LEA, MOV, NEG, OR, SUB, XOR, POP, JMP and XCHG. In performing attribute analysis, reduced feature selection and data amplification, MOV and XCHG were the only two Op Codes that were selected from an attribute perspective. Feature analysis routinely highlighted MOV and XCHG as interesting attributes. This study used the MOV and XCHG attributes in both reduced feature selection and amplified features to perform advanced detection. These two attributes produced highly effective detection rates for several experiments. These two attributes provided improved detection across all algorithms in this study. Additional research should be conducted to understand the nature of both Op Codes for malware and benign samples.

The last research area is single-layer packed and multi-layer packed executables. A substantial number of samples for this study used packing algorithms to avoid or evade further analysis. Packing algorithms commonly used for packing malware and hiding execution routines (Jeong, Choo, Lee, Bat-Erdene, & Lee, 2010). As previously discussed packing is also utilized to protect intellectual property for legitimate purposes. Packed executables essentially encode the data sections so that dynamic analysis cannot view results (Jeong et al., 2010). As these executables are loaded into memory, packed executables dynamically change the size and content of the data (Jeong et al., 2010). There were a number of samples in this study that exhibited single-layer packing, re-packing and multi-layer packing. This made feature extraction more difficult or in some cases impossible. This study did not find a single benign sample that used multi-layer packing. Therefore, additional research should be conducted to understand the nature of single-layer packed and multi-layer packed executables for malware detection. It is

believed that classification of packing algorithms could lead to better detection for advanced malware.

Summary

Malware represents some of the most serious security concerns for today's Internet. Security breaches and cyber-attacks can be directly attributed to malware or multi-stage cyber-attacks. Malware can compromise networks and computers in the form of botnets, viruses, worms, ransomware and advanced persistent threats (APTs). These cyber-attacks are launched using targeted and advanced malware techniques to steal personal, proprietary or financial information. The high number of attacks and the associated negative notoriety make malware one of the most popular areas for advanced research. Much of today's advanced research has been concentrated on developing techniques to collect, study, and mitigate malware. This research focused on detecting "real" malware and samples found "live" on the internet. As improved detection becomes a reality – mitigation or elimination of malware for end-points can be greatly enhanced. Unfortunately, current host-based detection approaches that leverage signature-based detection is largely ineffective for new polymorphic malware. Polymorphic malware avoids or evades signature detection by using advanced obfuscation or encryption techniques. The goal of this research was to address these malware detection shortcomings. New research was conducted to develop new dynamic detection approaches. New approaches demonstrated that using machine learning with advanced algorithms can correctly and efficiently identify potential malware threats.

This study demonstrated a novel malware detection approach that provides improved detection for polymorphic malware. The research should enhance and augment

traditional end-point detection approaches. This experimental study extracted key features from file properties, static and dynamic analysis. Machine learning using advanced algorithms correctly determined the likelihood of files (samples) to be benign (good) or malicious (bad). The prototype environment analyzed malware executables (program) in a controlled environment in order to better understand behaviors, function calls and the inclusion of dynamic libraries. Features were extracted and assembled for further analysis by three different cluster algorithms. Experiments were conducted to better understand the malware threat landscape in terms of file properties, static and dynamic analysis. Foundational and experimental reviews of previous research literature was conducted and summarized.

Detecting polymorphic and metamorphic malware continues to be a challenge for the security community. A majority of the today's security research is focused on developing enhanced detection using techniques that collect, study, and mitigate malicious code (Kolbitsch et al., 2009). However, new polymorphic malware and the detection evading techniques render many of the current signature protections useless and therefore leave end-points unprotected (Rodríguez-Gómez et al., 2013). The speed at which polymorphic malware is advancing threatens enterprise computing and internet operations (Symantec Corporation, 2016). Being able to detect polymorphic, metamorphic and zero-day malware requires advanced detection techniques that provide rapid adaptation, scalability and produce low false positive rates (Borojerdi & Abadi, 2013). This results from this study offers an attractive alternative for detecting polymorphic malware specifically targeting Windows operating systems. This study performed analysis on over a sample population of 2.5M files. This study assembled a

single data set containing known malware, known benign and unknown samples, performing feature extraction and developed a prototype environment for detection that far exceeded today's accepted baseline (Amos et al., 2013). This study successfully demonstrated a feature extraction methodology, a prototype detection environment and conducted a number of experiments within an accepted testing protocol.

This study developed a unique detection methodology using multidimensional topological features in a machine learning environment. The ability to extract unique multidimensional topological features utilizing file properties, static and dynamic analysis is a new approach for feature extraction. The study uses this information to develop a quantitative experimental prototype using multidimensional topological data with machine learning utilizing advanced algorithms. This approach and every experiment conducted for this study provided ample evidence based on the experimental dataset delivered improved detection for polymorphic malware far above 81.25%. All experiments used quantitative methods to provide objective measurements for experimental results as described by Babbie (2010). Experimental data collection, data analysis and presentation of results was conducted using standard and accepted computational techniques (Babbie, 2010). The study was carefully designed and executed using the SCED testing protocol. The study paid particular attention to assembly of representative datasets and the execution of the various experiments within the testing protocol. The testing methodology followed an approach that can easily be reproduced. All experiments were conducted in a safe environment such that the malware could not escape and infect the research environment (Rossow, 2013). The experimental prototype

was designed to assemble, capture, analyze and evaluate results with the rigorous quantitative evaluation measures discussed by Mohaisen and Alrawi (2015).

The goal of this study was to develop and demonstrate an experimental approach for improved detection for polymorphic malware that exceeded the effective detection rate of 81.25% for polymorphic malware (Amos et al., 2013). The experimental detection rate delivered across all three algorithms exceeded 99.99%. A number of the experiments (IBK & DLMLP) delivered effective detection results as high as 99.999%. The lowest effective rate observed through testing 99.43% for MB. The hypothesis for the study was that detection could be achieved by combining file properties, static and dynamic analysis features. The study successfully demonstrated through multiple experiments that multidimensional topological can deliver effective malware detection. Experimental testing with this dataset delivered effective detection using file properties, static and dynamic analysis features. This study leveraged the SCED protocol for performing and documenting various experimental tests. The test results were then quantitatively examined, evaluated, analyzed and presented testing outcomes. This study leveraged previous quantitative experiments for supervised machine learning for malware research (Boro, et al., 2012; Pradesh, 2014) to better understand how multidimensional topological information can be utilized for malware detection.

The study's target was to develop an approach that would deliver an effective detection rate of higher than 81.25% using multidimensional topological data for malware detection. The developed prototype system performed various feature extraction and assembled the datasets for training and testing. The study sought to establish baselines for three advanced cluster algorithms (MB, IBk and DLMLP) and

then manipulate feature weighting to exceed the target effective rate. In all cases, each algorithm at baseline testing delivered higher than target detection rates. Additionally, feature weighting through reduced feature selection and amplified features delivered substantially better detection performance over baseline results.

Although this study was successful in addressing the research goal there were a number of important conclusions, implications and recommendations that stemmed from this research. In terms of conclusions, this study produced three important conclusions: 1) multidimensional topological features did effectively demonstrate effective malware detection with machine learning algorithms, 2) advanced algorithms using multidimensional topological features delivered impressive detection results (MB (99.985%), IBk (100%) and DLMLP (99.9995%)) and 3) advanced algorithms (IBk and DLMLP) with reduced or amplified features delivered near perfect (99.9999%) malware detection results. These three conclusions provide greater insight into how multidimensional topological can be used to enhance endpoint detection.

There were two major implications from this study. The first implication is that polymorphic malware targeting Microsoft platforms can be readily detected through machine learning with multidimensional topological data. This is the first study to evaluate and successfully demonstrate malware detection using multidimensional topological data with machine learning. The second implication is that endpoint protection could potentially be augmented with minimal features (6 or 7) and machine learning for malware detection. This implication could change or expand how detection is conducted at the endpoint. Both implications could potentially expand enterprise

malware detection by commercial security endpoint products as well as network boundary devices scanning and evaluating executable transport at the network layer.

There are three areas in which additional research was recommended. Two of the three areas deal with executable functions or Opcodes. This study extracted and studied over 2M samples through a dynamic analysis process. Leveraging prior research (Bilar, 2007), only the top twelve Op Codes were collected, analyzed and counted. In some samples, these Op Codes were executed in abnormally high numbers (e.g. ADD nearly 10,000 times). Abnormally high counts may be another indication of maliciousness for detecting malware. The second area requiring additional research is for two specific Op Codes: MOV and XCHG. In over 2M samples, using reduced feature selection and data amplification these two Op Codes were found to provide additional indications of maliciousness for this dataset. However, additional research would have to be conducted to understand the nature of both Op Codes for malware and benign samples. The third research area involves packed executables. This study observed over 25% of the sample analyzed used either single-layer packed or multi-layer packed executables. Packing algorithms are commonly used for packing malware and hiding execution routines (Jeong et al., 2010). Packed executables encode aspects of execution to hide routines so that dynamic analysis cannot view results. This study did not find a single benign sample that used multi-layer packing. Therefore, additional research should be conducted to understand the frequency of single-layer packed and multi-layer packed executables for malware detection. (Kim & Hong, 2014). (Zhu et al., 2012)

References

- Ahmadi, M., Ulyanov, D., Semenov, S., Trofimov, M., & Giacinto, G. (2016). Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification. *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, 183–194.
- Alam, S., Horspool, R. N., & Traore, I. (2014). MARD: A framework for metamorphic malware analysis and real-time detection. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, 480–489.
- Alam, S., Horspool, R., & Traore, I. (2013). MAIL: Malware Analysis Intermediate Language: a step towards automating and optimizing malware detection. *Proceedings from the 6th International Conference on Security of Information Networks*.
- Amos, B., Turner, H., & White, J. (2013). Applying machine learning classifiers to dynamic Android malware detection at scale. *IEEE Wireless Communications and Mobile Computing Conference (IWCMC)*, 1666–1671.
- Babbie, E. (2010). *The Practice of Social Research. The Practice of Social Research*.
- Bächer, P., Holz, T., Kotter, M., & Wicherski, G. (2005). Tracking Botnets: Using Honeynets to Learn More about Bots. *The HoneyNet Project and Research Alliance*, 1–116.
- Baecher, P., Koetter, M., Holz, T., Dornseif, M., & Freling, F. (2006). The nepenthes platform: An efficient approach to collect malware. *Recent Advances in Intrusion Detection*, 165–184.
- Bailey, M., Oberheide, J., & Andersen, J. (2011). Automated classification and analysis of internet malware. *Recent Advances in Intrusion Detection*, 1–18.
- Barakat, A., & Khattab, S. (2010). A comparative study of traditional botnets versus super-botnets. *Informatics and Systems (INFOS), 2010 The 7th International Conference on*, 1–5.
- Bayer, U., Kirda, E., & Kruegel, C. (2010). Improving the efficiency of dynamic malware analysis. *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10*, 1871.
- Bayer, U., Milani-Comparetti, P., Hlauscheck, C., Kruegel, C., & Kirda, E. (2009). Scalable, Behavior-Based Malware Clustering. *16th Symposium on Network and Distributed System Security*, 120–129.
- Bechtel, K. (2014). Malware's Journey from Hobby to Profit-Driven Attacks, 10–26.
- Bilar, D. (2007). Opcodes as predictor for malware. *International Journal of Electronic Security and Digital Forensics*, 1(2), 156.
- Bogdan, R., & Biklen, S. (1998). Qualitative research in education: An introduction to theory and methods. *Foundations of Qualitative Research in Education*, 1–48.
- Bohra, A., Neamtiu, I., Gallard, P., Sultan, F., & Iftodet, L. (2004). Remote repair of operating system state using backdoors. *Proceedings - International Conference on*

- Autonomic Computing*, 256–263.
- Bontchev, V. (1996). Possible macro virus attacks and how to prevent them. *Computers & Security*, 15(7), 595–626.
- Bontchev, V. (1998). Methodology of computer anti-virus research. *Doctoral Thesis, Faculty of Informatics, University of Hamburg*, 1–233.
- Boro, D., Nongpoh, B., & Bhattacharyya, D. K. (2012). Anomaly based intrusion detection using meta ensemble classifier. *Proceedings of the Fifth International Conference on Security of Information and Networks - SIN '12*, 143–147.
- Borojerdi, H. R., & Abadi, M. (2013). MalHunter : Automatic Generation of Multiple Behavioral Signatures for Polymorphic Malware Detection. *IEEE Systems Journal*, (3rd International Conference on Computer and Knowledge Engineering (ICCKE 2013)), 1–7.
- Bossert, G., Hiet, G., & Inria, S. (2014). Towards Automated Protocol Reverse Engineering Using Semantic Information Categories and Subject Descriptors. *ACM Symposium on Information, Computer and Communications Security (2014)*, 51–62.
- Brenner, S. (2008). *Cyberthreats: The Emerging Fault Lines of the Nation State*. New York.: Oxford University Press.
- Campbell, D., & Stanley, J. (1963). *Experimental and Quasi-Experimental Designs for Research*. American Educational Research Association.
- Campo-Giralte, L., Jimenez-Peris, R., & Patino-Martinez, M. (2009). PolyVaccine: Protecting Web Servers against Zero-Day, Polymorphic and Metamorphic Exploits. *2009 28th IEEE International Symposium on Reliable Distributed Systems*, 91–99.
- Carlson, S. M., Davis, A. C., & Leach, J. G. (2014). Less Is More, (February).
- Cesare, S., & Xiang, Y. (2010). A Fast Flowgraph Based Classification System for Packed and Polymorphic Malware on the Endhost. *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 721–728.
- Cesare, S., & Xiang, Y. (2011). Malware Variant Detection Using Similarity Search over Sets of Control Flow Graphs. *2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, 181–189.
- Cesare, S., & Xiang, Y. (2013). Malwise — An Effective and Efficient Classification System for Packed and Polymorphic Malware. *IEEE Transactions on Computers*, 62(6), 1193–1206.
- Cesare, S., & Xiang, Y. (2014). Control Flow-Based Malware Variant Detection. *IEEE Transaction on Dependable and Secure Computing*, 11(4), 304–317.
- Cesare, S., Xiang, Y., & Zhou, W. (2007). Malwise – An Effective and Efficient Classification System for Packed and Polymorphic Malware. *IEEE Transactions on Computers*, 1–14.
- Cesare, S., Xiang, Y., & Zhou, W. (2013). Malwise-an effective and efficient classification system for packed and polymorphic malware. *IEEE Transactions on Computers*, 62(6), 1193–1206.
- Charles, C. (1995). *Introduction to educational research*. (Longman, Ed.) (Second Edi).

San Diego, CA.

- Chaumette, S., Ly, O., & Tabary, R. (2011). Automated extraction of polymorphic virus signatures using abstract interpretation. *Network and System Security*.
- Chen, K., Zhang, Y., & Lian, Y. (2013). Vulnerability-based backdoors: Threats from two-step trojans. *Proceedings - 7th International Conference on Software Security and Reliability, SERE 2013*, 169–177.
- Chen, T., & Robert, J.-M. (2004). Evolution of Viruses and Worms. *Statistical Methods in Computer Security*, 1–16. Retrieved from <http://vx.netlux.org/lib/atc01.html>
- Chu, B., Holt, T., & Ahn, G. (2010). Examining the Creation, Distribution, and Function of Malware On Line. *Department of Justice Abstract*, 1–183.
- Cohen, F. (1985). Computer viruses, (January), 1–152.
- Cohen, F. (1989a). Computational aspects of computer viruses. *Computers & Security*, 8(4), 297–298.
- Cohen, F. (1989b). Models of practical defenses against computer viruses. *Computers & Security*, 8(2), 149–160.
- Cohen, F. (1992). A formal definition of computer worms and some related results. *Computers Security*, 11(7), 641–652.
- Condon, F. (2012). Towards a Scalable Framework for Android Application Analytics. *UC Berkley 2013 Security Symposium*, 1–10.
- Creswell, J. W. (2007). *Research Design: Qualitative, Quantitative and Mixed Method Approaches*. SAGE Publications.
- Crocker, L., & Algina, J. (1986). *Introduction to classical and modern test theory*. Orlando, FL: Holt, Rinehart and Winston.
- Dai, J., Guha, R., & Lee, J. (2009). Efficient Virus Detection Using Dynamic Instruction Sequences. *Journal of Computers*, 4(5), 405–414.
- David, Y., & Yahav, E. (2013). Tracelet-based code search in executables. *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation - PLDI '14*, 349–360.
- Denning, J. (1989). The Science of Computing: The Internet Worm. *American Scientist*, Vol. 77(No. 2), 126–128.
- Denzin, N. K., & Lincoln, Y. S. (2006). The Sage Handbook of Qualitative Research, 2nd ed. Edited by Norman K. Denzin, and Yvonna S. Lincoln. *Library*, 28(August), 467–468.
- Devesa, J., Santos, I., Cantero, X., Penya, Y. K., & Bringas, P. G. (2010). Automatic behaviour-based analysis and classification system for malware detection. *Computer*, 2 AIDSS(November 2015), 395–399.
- Dunlap, C. J. (2011). Perspectives for Cyber Strategists on Law for Cyberwar. *Strategic Studies Quarterly*, (June 2010), 81–99.
- Eichin, M. W., & Rochlis, J. a. (1989). With microscope and tweezers: an analysis of the Internet virus of November 1988. *Proceedings. 1989 IEEE Symposium on Security*

- and Privacy, (November), 1–17.
- Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters* 27 (2006) 861–874, 27(6), 861–874.
- Fraley, J. B., & Figueroa, M. (2016). Polymorphic malware detection using topological feature extraction with data mining. In *SoutheastCon 2016* (pp. 1–7).
- Gazet, A. (2010). Comparative analysis of various ransomware virii. *Journal in Computer Virology*, 6(1), 77–90.
- Gheorghescu, M. (2005). An Automated Virus Classification System. *Virus Bulletin Conference*, (October), 294–300.
- Glesne, C., & Peshkin, A. (1992). *Becoming qualitative researchers: An introduction*. White Plains, NY: Longman.
- Gordon, S., & Chess, D. (1998). Attitude Adjustment : Trojans and Malware on the Internet Attitude Adjustment : Trojans and Malware on the Internet, (October 1998).
- Gostev, A., Unuchek, R., Garnaeva, M., Makrushin, D., & Ivanov, A. (2016). It Threat Evolution in Q1 2016. *Kaspersky 2015 Report*, Kaspersky L.
- Gruber, T., Cammerer, S., Hoydis, J., & Brink, S. ten. (2017). On Deep Learning-Based Channel Decoding. *arXiv Preprint arXiv:1701.07738*, 1–6.
- Guri, M., Kedma, G., & Sela, T. (2013). Noninvasive detection of anti-forensic malware. *Proceedings of the 2013 8th International Conference on Malicious and Unwanted Software: "The Americas", MALWARE 2013*, 1–10.
- Hampton, N., & Baig, Z. A. (2015). Ransomware: Emergence of the cyber-extortion menace. *The Proceedings of the 13th Australian Information Security Management, 2015*, 47–56.
- Hansen, S. S., Mark, T., Larsen, T., Stevanovic, M., & Pedersen, J. M. (2016). An Approach for Detection and Family Classification of Malware Based on Behavioral Analysis. *2016 International Conference on Computing; Networking and Communications (ICNC)*, (February), 1–5.
- Hoepfl, M. C. (1997). Choosing Qualitative Research : A Primer for Technology Education Researchers. *Journal of Technology Education*, 9(1), 47–63.
- Hu, C., Wang, X., Li, N., Bai, H., & Jing, X. (2014). Approach for malware identification using dynamic behaviour and outcome triggering. *IET Information Security*, 8(2), 140–151.
- Jeong, G., Choo, E., Lee, J., Bat-Erdene, M., & Lee, H. (2010). Generic unpacking using entropy analysis. *Proceedings of the 5th IEEE International Conference on Malicious and Unwanted Software, Malware 2010*, 98–105.
- John, G., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference*, 121–129.
- Kamongi, P., Kotikela, S., Kavi, K., Gomathisankaran, M., & Singhal, A. (2013). VULCAN: Vulnerability Assessment Framework for Cloud Computing. *2013 IEEE 7th International Conference on Software Security and Reliability*, 218–226.

- Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., & Savage, S. (2008). Spamalytics: an empirical analysis of spam marketing conversion. *ACM Conference on Computer and Communications Security*, 3–14.
- Kaur, R. (2014). Efficient Hybrid Technique for Detecting Zero-Day Polymorphic Worms, (September 2011), 95–100.
- Kaur, R., & Singh, M. (2014). A Survey on Zero-Day Polymorphic Worm, *16*(3), 1520–1549.
- Kauranen, K., & Makinen, E. (1990). A note on Cohen’s formal model for computer viruses. *ACM SIGSAC Review*, 8(2), 40–43.
- Kayacik, H., Zincir-Heywood, a N., & Heywood, M. I. (2005). Selecting Features for Intrusion Detection : A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets. *Proceedings of the Third Annual Conference on Privacy Security and Trust PST2005*, 3–8.
- Kephart, J., & Arnold, W. (1994). Automatic extraction of computer virus signatures. *4Th Virus Bulletin International Conference*, 178–184.
- Kephart, J. O. (1993). Measuring and Modeling Computer. *Proceedings of IEEE Computer Society Symposium on Security and Privacy*, 2–15.
- Kephart, J., Sorkin, G., Arnold, W., Chess, D., Tesauro, G., & White, S. (1995). Biologically inspired defenses against computer viruses, 985–996.
- Kephart, J., Sorkin, G., Swimmer, M., & White, S. (1999). Blueprint for a computer immune system. *IBM Thomas J. Watson Research Center*, 1–17.
- Kephart, J., & White, S. (1991). Directed-graph epidemiological models of computer viruses. *Proceedings., 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, 343–359.
- Kerchen, P., Lo, R., Crossley, J., Elkinbard, G., Levitt, C., & Olsson, R. (1990). Static Analysis Virus Detection Tools For Unix Systems. *Proceedings of the 13th National Computer Security Conference*, 350–365.
- Khakhutskyy, V. (2016). Behavior-based Malware Detection with Quantitative Data Flow Analysis, (c), 2–4.
- Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., & Kirda, E. (2015). Cutting the gordian knot: A look under the hood of ransomware attacks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9148, 3–24.
- Kim, K., & Hong, S. (2014). Study on Enhancing Vulnerability Evaluations for BYOD Security. *Perspectives on Security*, 8(4), 229–238.
- Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *International Joint Conference on Artificial Intelligence*, 14(12), 1137–1143.
- Kolbitsch, C., Comparetti, P. M., Kruegel, C., Kirda, E., Zhou, X., Wang, X., ... Antipolis, S. (2009). Effective and Efficient Malware Detection at the End Host. *System*, 4(1), 351–366.

- Kolbitsch, C., Holz, T., Kruegel, C., & Kirda, E. (2010). Automated Extraction of Proprietary Gadgets from Malware Binaries. *2010 IEEE Symposium on Security and Privacy*, 29–44.
- Kolo, E. (2016). Kolobyte. *Kolobyte Blog*, February.
- Kolter, J., & Maloof, M. (2006). Learning to detect and classify malicious executables in the wild. *Journal of Machine Learning Research*, 2721–2744.
- Krebs, B. (2012). Massive Profits Fueling Rogue Antivirus Market. *Washington Post*.
- Lana, R. (1959). Pretest-treatment interaction effects in attitudinal studies, *56*(4), 293.
- Landwehr, C. E., Bull, A. R., McDermott, J. P., & Choi, W. S. (1994). A taxonomy of computer program security flaws. *ACM Computing Surveys*, 26(3), 211–254.
- Lee, C., & Lee, G. G. (2006). Information gain and divergence-based feature selection for machine learning-based text categorization. *Information Processing and Management*, 42(1 SPEC. ISS), 155–165.
- Lee, T., & Mody, J. (2006). Behavioral Classification. In *15th European Institute for Computer Antivirus Research (EICAR 2006) Annual Conference*.
- Lee, W., & Stolfo, S. J. (1998). Data Mining Approaches for Intrusion Detection. *Proceedings Of the Seventh Usenix Security Symposium*, 79–93.
- Li, X., Duan, H., Liu, W., & Wu, J. (2010). The growing model of Botnets. *1st International Conference on Green Circuits and Systems, ICGCS 2010*, 414–419.
- Linn, C., & Debray, S. (2003). Obfuscation of executable code to improve resistance to static disassembly. *Proceedings of the 10th ACM Conference on Computer and Communication Security - CCS '03*, 290–299.
- Liu, Y., Chen, W., & Guan, Y. (2012). Monitoring Traffic Activity Graphs with low-rank matrix approximation. *37th Annual IEEE Conference on Local Computer Networks*, 59–67.
- Lo, R. W., Levitt, K. N., & Olsson, R. a. (1995). MCF: A malicious code filter. *Computers & Security*, 14(6), 541–566.
- Mandiant Research. (2014). Beyond the Breach. *Mandiant Threat Report*, 1–79.
- Manikandan, P., Ramyachitra, D., Kalaivani, S., & Ranjani Rani, R. (2016). An improved instance based K-nearest neighbor (IIBK) classification of imbalanced datasets with enhanced preprocessing. *International Journal of Applied Engineering Research*, 11(1), 642–649.
- Martignoni, L., Christodorescu, M., & Jha, S. (2007). OmniUnpack: Fast, generic, and safe unpacking of malware. *Proceedings - Annual Computer Security Applications Conference, ACSAC*, 431–440.
- McAfee. (2014). McAfee Labs Threats Report, (June).
- McAfee. (2016). McAfee Labs Threat Report, (June).
- McAfee Labs. (2015). McAfee Labs Threats Report, (May).
- McGraw, G., & Morrisett, G. (2000). Attacking malicious code: A report to the Infosec Research Council. *IEEE Software*, 17(5), 33–41.

- Mohaisen, A., & Alrawi, O. (2015). High-fidelity, Behavior-based Automated Malware Analysis and Classification. *Computers & Security*, 1–12.
- Molok, N. N. A., Ahmad, A., & Chang, S. (2012). Social networking: a source of intelligence for advanced persistent threats. *International Journal of Cyber Warfare and Terrorism (IJCWT)*, 2(1)(July), 1–13.
- Moore, D., Shannon, C., & Brown, J. (2002). Code-Red: A Case Study on the Spread and Victims of an Internet Worm. *Proceedings of the Second ACM SIGCOMM Workshop on Internet Measurement (IMW '02)*, 273–284.
- Morse, J. M., Barrett, M., Mayan, M., Olson, K., & Spiers, J. (2008, December 19). Verification Strategies for Establishing Reliability and Validity in Qualitative Research. *International Journal of Qualitative Methods*.
- Muhaya, F. Bin, Khan, M. K., & Xiang, Y. (2011). Polymorphic Malware Detection Using Hierarchical Hidden Markov Model. *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, (9), 151–155.
- Murphy, K. P., Weiss, Y., & Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. *Proceedings of Uncertainty in AI*, 9(4), 467–475.
- Nazario, J., Ptacek, T., & Song, D. (2004). Wormability: A description for vulnerabilities. *Arbor Networks*.
- Neuman, S. B., & McCormick, S. (1995). *What is single-subject experimental research? Single-Subject Experimental Research: Applications for Literacy* (128th ed.). Newark, Delaware: International Reading Association.
- Nicho, M., & Khan, S. (2014). Identifying Vulnerabilities of Advanced Persistent Threats: *International Journal of Information Security and Privacy*, 8(1), 1–18.
- Noreen, S., Murtaza, S., Shafiq, M. Z., & Farooq, M. (2009). Evolvable malware. *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation GECCO 09*, 1569–1576.
- Panda Security. (2014). PANDA Labs Annual Report 2014.
- Paulevé, L., Jégou, H., & Amsaleg, L. (2010). Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11), 1348–1358.
- Pék, G., Lanzi, A., & Srivastava, A. (2014). On the feasibility of software attacks on commodity virtual machine monitors via direct device assignment. *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, 305–316.
- Pfeffer, A., Call, C., & Chamberlain, J. (2012). Malware Analysis and attribution using Genetic Information. *2012 7th International Conference on Malicious and Unwanted Software*, 39–45.
- Ponomarev, S., Durand, J., Wallace, N., & Atkison, T. (2013). Evaluation of random projection for malware classification. *Proceedings - 7th International Conference on Software Security and Reliability Companion, SERE-C 2013*, 68–73.

- Pradesh, A. (2014). Poster : Machine-learning Approaches for P2P Botnet Detection using Signal-processing Techniques. *Debs '14*, 338–341.
- Pramono, Y. W. T., & Suhardi. (2015). Design of anomaly-based intrusion detection and prevention system for smart city web application using rule-growth sequential pattern mining. *Proceedings - 2014 International Conference on ICT for Smart Society: "Smart System Platform Development for City and Society, GoeSmart 2014"*, *ICISS 2014*, 56–60.
- Qu, Y., & Hughes, K. (2013). Detecting metamorphic malware by using behavior-based aggregated signature. *Internet Security (WorldCIS), 2013 World Conference Proceedings*, 13–18.
- Rafique, M., & Chen, P. (2014). Evolutionary algorithms for classification of malware families through different network behaviors. *In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 1167–1174.
- Rekdal, J., & Bloemerus, P. (2013). Advanced Persistent Threat (APT) Beyond the hype, 1–50.
- Reynolds, J. K. (1989). The helminthiasis of the Internet. *Computer Networks and ISDN Systems*, 22(5), 347–361.
- Richardson, R. (2011). 15th Annual 2010/2011 Computer Crime and Security Survey. *Computer Security Institute*, 1–90.
- Rieck, K., Trinius, P., Willems, C., & Holz, T. (2009). Automatic Analysis of Malware Behavior using Machine Learning. *Journal of Computer Security*, (18–2009), 1–30.
- Rieck, K., Trinius, P., Willems, C., & Holz, T. (2011). Automatic Analysis of Malware Behavior using Machine Learning, 1–30.
- Ritchey, R. W., & Ammann, P. (2000). Using Model Checking to Analyze Network Vulnerabilities. *IEEE Security and Privacy*, 156–165.
- Rizvi, S. L., & Nock, M. K. (2008). Single-case experimental designs for the evaluation of treatments for self-injurious and suicidal behaviors. *Suicide & Life-Threatening Behavior*, 38(5), 498–510.
- Rodrigues, M. (2011). Utilizing Rootkits To Address the Vulnerabilities Exploited By, (December).
- Rodríguez-Gómez, R. a., Maciá-Fernández, G., & García-Teodoro, P. (2013). Survey and taxonomy of botnet research through life-cycle. *ACM Computing Surveys*, 45(4), 1–33.
- Rossow, C. (2013). Using Malware Analysis to Evaluate Botnet Resilience. *Vrije Universiteit*, (April), 145.
- Rossow, C., Dietrich, C. J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., ... Van Steen, M. (2012). Prudent practices for designing malware experiments: Status quo and outlook. *Proceedings - IEEE Symposium on Security and Privacy*, 65–79.
- Schneidewind, N. (2010). Metrics for mitigating cybersecurity threats to networks. *IEEE Internet Computing*, 14(1), 64–71.
- Seeley, D. (1989). A Tour of the Worm. *Proceedings of the USENIX Winter Technical*

- Conference, 287–304.
- Seigneur, J., & Kölnsdorfer, P. (2013). A survey of trust and risk metrics for a BYOD mobile working world. *Third International Conference on Social Eco-Informatics (SOTICS 2013)*, 11–22.
- Shafiq, M. Z., Khayam, S. A., & Farooq, M. (2008). *Embedded malware detection using markov n-grams*. *Dimva* (Vol. 7591).
- Sharma, A., & Sahay, S. (2014). Evolution and Detection of Polymorphic and Metamorphic Malwares: A Survey. *International Journal of Computer Applications*, 90(2), 7–12. Retrieved from <http://adsabs.harvard.edu/abs/2014IJCA...90b...7S>
- Sharma, N., Bajpai, A., & Litoriya, R. (2012). Comparison the various clustering algorithms of weka tools. *International Journal of Emerging Technology and Advanced Engineering*, 2(5), 73–80.
- Siddiqui, M. A. (2008). *Data Mining Methods for Malware Detection*. ProQuest.
- Singh, A., Walenstein, A., & Lakhotia, A. (2012). Tracking concept drift in malware families. *Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence - AISec '12*, 81–92.
- Singh, P. K., & Lakhotia, A. (2002). Analysis and detection of computer viruses and worms. *ACM SIGPLAN Notices*, 37(2), 29.
- Smith, M., & Glass, G. (1987). *Research and evaluation in education and the social sciences*. Prentice Hall.
- Spafford, E. H. (1989). The internet worm program: an analysis. *ACM SIGCOMM Computer Communication Review*, 19(1), 17–57.
- Spafford, E. H. (1991). Computer viruses and ethics. *Purdue University ePubs*, (Technical Report 91-061), 1–22.
- Spafford, E. H. (1994). Computer Viruses as Artificial Life. *Artificial Life*, 1(3), 249–265.
- Subramanya, S. R., & Lakshminarasimhan, N. (2001). Computer viruses. *IEEE Potentials*, 20(4), 18–21.
- Sulaiman, A., Ramamoorthy, K., Mukkamala, S., & Sung, A. (2005). Malware examiner using disassembled code (medic). *Proceedings of the 2005 IEEE Workshop on Information Assurance and Security*, 428–429.
- Symantec. (2014). Internet Security Threat Report, 19(April).
- Symantec Corporation. (2014). Symantec Intelligence Report, (June).
- Symantec Corporation. (2016). 2016 Internet Security Threat Report. *Industry Report*, 1–81.
- Szor, P. (2005). *The Art of Computer Virus Research and Defense*. Pearson Education.
- Tamersoy, A., Roundy, K., & Chau, D. H. (2014). Guilt by Association: Large Scale Malware Detection by Mining File-relation Graphs. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '14*, 1524–1533.

- Tankard, C. (2011). Advanced persistent threats and how to monitor and deter them. *Network Security*, (8), 16–19.
- Thompson, K. (1984). Reflections of Trust. *Communications of the ACM*, 27(8), 761–763.
- Virvilis, N., Gritzalis, D., & Apostolopoulos, T. (2013). Trusted Computing vs. Advanced Persistent Threats: Can a Defender Win This Game? *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, 396–403.
- Vogt, R., Aycock, J., & Jacobson, M. (2007). Army of botnets. *Network and Distributed System Security Symposium*, 111–123.
- von Nuemann, J., & Burks, A. W. (1966). Theory of self-reproducing automata. *IEEE Transactions on Neural Networks*, 5(10), 3–14.
- Wagner, D., & Dean, R. (2001). Intrusion detection via static analysis. *Security and Privacy, 2001. Proceedings 2001 IEEE Symposium on Security and Privacy, 2001.*, 156–168.
- Wall, D. S. (2012). Enemies within: Redefining the insider threat in organizational security policy. *Security Journal*, 26(2), 107–124.
- Weaver, N., Paxson, V., Staniford, S., & Cunningham, R. (2003). A taxonomy of computer worms. *Proceedings of the 2003 ACM Workshop on Rapid Malcode*, 11–18.
- Wheeler, D. a. (2005). Countering trusting trust through diverse double-compiling. *Proceedings - Annual Computer Security Applications Conference, ACSAC, 2005*, 33–45.
- White, S. R. (1998). Open Problems in Computer Virus Research. *Virus Bulletin Conference, October*, 1–11.
- Williamson, M. M., Parry, A., Byde, A., Bristol, H. P. L., Road, F., & Gifford, S. (2004). Virus Throttling for Instant Messaging. *Virus Bulletin Conference*, (September), 38–48.
- Willson, V., & Putnam, R. (1982). A meta-analysis of pretest sensitization effects in experimental design. *Educational Research Journal*, 19(2), 249–258.
- Winter, G. (2000). A comparative discussion of the notion of 'validity' in qualitative and quantitative research. *The Qualitative Report*, 4(3), 1–14.
- Wüchner, T., Ochoa, M., & Pretschner, A. (2014). Malware Detection with Quantitative Data Flow Graphs. *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security - ASIA CCS '14*, 271–282.
- Xiao, X., Yuxin, D., Yibin, Z., Ke, T., & Wei, D. A. I. (2013). Malware Detection Based on Object-oriented Association Mining. *Proceedings of the 2013 International Conference on Machine Learning and Cybernetics*, 14–17.
- Ye, Y., Wang, D., Li, T., Ye, D., & Jiang, Q. (2008). An intelligent PE-malware detection system based on association mining. *Journal in Computer Virology*, 4(4), 323–334.

- Yedidia, J., Freeman, W., and Weiss, Y. (2003). Understanding belief propagation and its Generalizations. *Exploring Artificial Intelligence in the New Millennium*, 8, 236–239.
- Yerima, S. Y., Sezer, S., McWilliams, G., & Muttik, I. (2013). A New Android Malware Detection Approach Using Bayesian Classification. *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, 121–128.
- Yin, H., Song, D., Egele, M., Kruegel, C., & Kirda, E. (2007). Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis. *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, 116–127.
- Yin, R. (2015). *Qualitative research from start to finish*. Guilford Publications.
- Young, A., & Yung, M. (2016). Cryptography as an Attack Technology: Proving the RSA/Factoring Kleptographic Attack. *The New Codebreakers*, 243–255.
- Zhang, J., Luo, X., Perdisci, R., Gu, G., Lee, W., & Feamster, N. (2011). Boosting the scalability of botnet detection using adaptive traffic sampling. *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security - ASIACCS '11*, 124.
- Zhang, Y., & Paxson, V. (2000). Detecting Backdoors, (August), 1–11.
- Zhu, X., Vondrick, C., Ramanan, D., & Fowlkes, C. (2012). Do We Need More Training Data or Better Models for Object Detection? *Procedings of the British Machine Vision Conference 2012 (BMVC12)*, 80.1-80.11.
- Zhuang, W., Ye, Y., Chen, Y., & Li, T. (2012). Ensemble clustering for internet security applications. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 42(6), 1784–1796.
- Zolotukhin, M., & Hamalainen, T. (2013). Support vector machine integrated with game-theoretic approach and genetic algorithm for the detection and classification of malware. *2013 IEEE Globecom Workshops (GC Wkshps)*, 211–216.

Appendix A – Advanced Detection Studies

Lead Author	Year	Study Type	Dataset	Malware %	Labeled Samples	Static Analysis	Dynamic Analysis
Park, Y. (2010)	2010	Experiment	200	55%	Yes	Yes	Yes
Shamili, A. S. (2010, August).	2010	Quantitative Experiments	897,922	33%	No	None	None
Sun, X. (2010)	2010	Experiment	90	67%	Yes	None	Yes
Wang, Z. H. (2010, May)	2010	Iterative Experiment	5,223	68%	Yes	Yes	None
Ye, Y. (2010).	2010	Experiment	50,000	70%	Yes	None	None
Caballero, J. (2011)	2011	Experiment	313,791	Not Specified	Yes	Yes	None
Jacob, G. (2011)	2011	Experiment	37,572	Not Specified	Yes	None	Yes
Jang, J. (2011)	2011	Experiment	20,000	100%	Yes	Yes	Yes
Kolbitsch, C. (2011)	2011	Experiment	Not Specified	Not Specified	Yes	Yes	Yes
Rossow, C. (2011)	2011	Experiment	100,000	85%	Yes	Yes	None
Zhang, J. (2011)	2011	Experiment	Not Specified	Not Specified	Yes	None	None
Chen, Y. (2012)	2012	Experiment	10,000	60%	Yes	Yes	None
Eskandari, M. (2012).	2012	Experiment	956	52%	Yes	Yes	Yes
Ghiasi, M. (2012)	2012	Experiment	1,211	68%	Yes	None	Yes
Zhuang, W. (2012)	2012	Experiment	75,000	78%	Yes	None	None
Borojerdi, H. R. (2013)	2013	Experiment	360	67%	Yes	None	Yes
Jha, S. (2013)	2013	Experiment	961	95%	Yes	None	Yes
Naval, S. (2013)	2013	Experiment	1,296	37%	Yes	Yes	None
Ponomarev, S. (2013)	2013	Experiment	1,544	54%	Yes	Yes	None
Tsuruta, H. (2013)	2013	Experiment	23,234,538	27%	Yes	Yes	None

Lead Author	Year	Study Type	Dataset	Malware %	Labeled Samples	Static Analysis	Dynamic Analysis
Wang, H. T., Wei, T. E., & Lee, H. M. (2013)	2013	Experiment	3,000	96%	Yes	Yes	Yes
Xiao, X. (2013)	2013	Experiment	3,401	53%	Yes	None	Yes
Zolotukhin, M. (2013)	2013	Experiment	1,089	55%	Yes	Yes	None
Adebayo, O.S. (2014)	2014	Empirical Study	1,500	67%	Yes	Yes	None
Dornhackl, H. (2014)	2014	Observation	Not Specified	Not Specified	No	None	None
El Attar, A. (2014)	2014	Experiment	Not Specified	Not Specified	No	None	None
Elaziz, P. E. A. (2014)	2014	Experiment	80,077	Not Specified	No	None	None
Li, J. (2014)	2014	Experiment	Not Specified	Not Specified	No	None	None
Pramono, Y. W.T. (2014, August)	2014	Experiment	1,059,419	Not Specified	No	None	None
Pramono, Y. W.T. (2014, September)	2014	Experiment	Not Specified	Not Specified	No	None	None
Li, Y.H. (2015)	2015	Experiment	Not Specified	Not Specified	No	None	Yes
Hansen, S.S. (2016)	2016	Experiment	270,837	100%	Yes	None	Yes

Appendix B – Cuckoo Installation and Configuration

Eugene Kolo's blog - Provided useful information for installing and configuring Cuckoo (Kolo, 2016).

<https://eugenekolo.com/blog/installing-and-setting-up-cuckoo-sandbox/>

Installation dependencies for Cuckoo

```
sudo apt-get install python
sudo apt-get install mongodb
sudo apt-get install g++
sudo apt-get install python-dev python-dpkt python-jinja2 python-magic python-
pymongo
python-gridfs python-libvirt python-bottle python-pefile python-chardet python-pip
sudo apt-get install libxml2-dev libxslt1-dev
sudo pip2 install sqlalchemy yara
sudo pip2 install cybox==2.0.1.4
sudo pip2 install maec==4.0.1.0
sudo pip2 install python-dateutil
sudo apt-get install python-dev libfuzzy-dev
sudo pip2 install pydeep
sudo apt-get install tcpdump # If not installed
# Allow tcpdump to read raw TCP data without root:
sudo setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump
wget http://downloads.volatilityfoundation.org/releases/2.4/volatility-2.4.zip && unzip
volatility-2.4.zip && cd volatility-2.4
sudo python setup.py install
# Install the libraries that volatility wants:
sudo pip2 install distorm3
```

Install Cuckoo and Virtual Box

```
git clone git://github.com/cuckoosandbox/cuckoo.git
```

```
wget http://download.virtualbox.org/virtualbox/5.0.14/virtualbox-5.0_5.0.14-105127~Ubuntu~trusty_i386.deb
```

```
sudo dpkg -i virtualbox-5.0_5.0.14-105127~Ubuntu~trusty_i386.deb
```

```
sudo apt-get install -f
```

Networking

```
sudo iptables -A FORWARD -o eth0 -i vboxnet0 -s 192.168.56.0/24 -m conntrack --ctstate NEW -j ACCEPT;
```

```
sudo iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT;
```

```
sudo iptables -A POSTROUTING -t nat -j MASQUERADE;
```

```
sudo sysctl -w net.ipv4.ip_forward=1;
```