National College of Ireland

Analyzing Obfuscation Techniques for Evasion: A Case Study on Machine Learning-based Malware Detection

MSc Research Project
Artificial Intelligence

Tugrul Un
Student ID: 22181695

School of Computing
National College of Ireland

Supervisor: Anh Duong Trinh

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Tugrul Un
Student ID:	22181695
Programme:	Artificial Intelligence
Year:	2024
Module:	MSc Research Project
Supervisor:	Anh Duong Trinh
Submission Due Date:	05/01/2024
Project Title:	Analyzing Obfuscation Techniques for Evasion: A Case Study on Machine Learning-based Malware Detection
Word Count:	3140
Page Count:	13

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	31st January 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Analyzing Obfuscation Techniques for Evasion: A Case Study on Machine Learning-based Malware Detection

Tugrul Un
22181695

Abstract

This research focuses on the intricate domain of malware obfuscation, a method utilized by malicious people to conceal the actual characteristics and operations of their code, thereby rendering its analysis and reverse engineering more difficult and time-consuming. Consequently, this enhances the malware's capacity to elude detection and preventive methods. An area that is crucial but not thoroughly examined is the effect of these obfuscation techniques on malware detectors that rely on Machine Learning (ML). The primary objective of the research is to carry out a thorough examination of several obfuscation techniques, such as encryption, code obfuscation, and polymorphism, employed by attackers to conceal their malware. Having this comprehension is crucial for evaluating the present and possible future scenario of cyber risks. Additionally, the project aims to assess the impact of different obfuscation approaches on Static ML-based malware detectors. This study rigorously evaluates the impact of obscuring malware on the precision and efficiency of machine learning-based detection algorithms. An assessment of this nature is essential for uncovering the current capabilities and constraints of machine learning detectors when faced with advanced obfuscation techniques. Finally, the study aims to improve the identification and prevention methods in cybersecurity by identifying the weaknesses of machine learning-based malware detectors when faced with obfuscation attacks. The objective is to make a significant contribution to the area by suggesting ways that can strengthen the resistance of machine learning-based detectors against sophisticated and disguised malware threats. This would enhance cybersecurity defenses against constantly emerging malware issues.

1 Introduction

The ever-evolving landscape of cyber threats has brought to the forefront the critical issue of malware obfuscation. This method, employed by malicious entities, successfully conceals dangerous software, rendering it progressively difficult to detect and analyze. The necessity of tackling this problem is underscored by the increasing complexity of these methods of obfuscation and their capacity to weaken existing cybersecurity safeguards. This study aims to address a crucial inquiry: What is the influence of malware obfuscation techniques on the efficacy of Machine Learning (ML)-based malware detectors, and how can these systems be enhanced to mitigate such risks?

The objective of our research is to conduct an extensive examination of various methods used to obscure malware, encompassing encryption and code polymorphism among

others. This investigation is crucial for comprehending the developing strategies employed by cyber assailants. Furthermore, our objective is to assess the durability of Static ML-based malware detectors when confronted with these intricate obfuscation tactics. Gaining a comprehensive understanding of the constraints and advantages of these detectors will provide us with valuable insights to propose improvements to existing cybersecurity defenses.

This study enhances the scientific community's knowledge by providing a detailed understanding of how malware obfuscation and machine learning-based detection systems interact with each other. This research contributes to the progress of cybersecurity defenses by identifying and proposing enhancements for current vulnerabilities. The study is organized as an introductory overview, which is then followed by a literature analysis that places our results within the current landscape of cybersecurity and malware obfuscation. The next sections provide a comprehensive explanation of our study approach, explain the discovered results, and analyze their significance within the wider context of cybersecurity. Ultimately, the report finishes by providing a concise overview of our significant contributions, acknowledging the constraints of the present study, and suggesting potential avenues for future research in this crucial field.

2 Related Work

The pervasive problem posed by malicious software in the field of information technologies requires advanced solutions. Over time, multiple methods have been developed to identify such malware. However, repeated breaches and subsequent enhancements to protection systems highlight the necessity for more flexible solutions. Although the technical feasibility of the iterative problem-solving cycle is evident, there is an urgent need for systems that can quickly adapt. This has led to the incorporation of artificial intelligence (AI). Artificial intelligence (AI) is becoming increasingly important and efficient in identifying dangerous software. This judgment is achieved by using datasets that include both harmful and benign software. Various AI models are trained on these datasets. The combination of these endeavors enables AI to effectively recognize complex patterns, thus improving its ability to identify potential dangers.

2.1 Machine Learning-based Malware Detectors and Datasets

2.1.1 Malware Detection by Eating a Whole EXE

"Malware Detection by Eating a Whole EXE" examines a new method for detecting malware by consuming complete executable (EXE) files as raw byte sequences. Within the domain of cybersecurity, the detection of malevolent software is of utmost importance, considering the increasing risks to computer systems. In contrast to traditional antivirus techniques that frequently fail to detect novel and advanced malware, the authors suggest utilizing neural networks for static analysis, with a specific focus on the raw binary data of EXE files. Detailed explanation is in Section 4[1]

This is a architecture diagram of MalConv model in Figure 3

[1]MalConv: `https://github.com/NeuromorphicComputationResearchProgram/MalConv2`

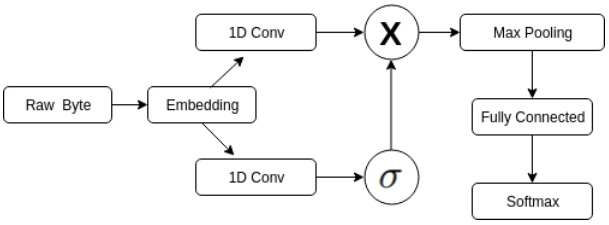

Figure 1: MalConv model Raff et al. (2018)

2.1.2 EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models

The EMBER dataset[2], introduced in this study, holds significant value in the field of machine learning for the purpose of detecting malware. The dataset comprises 1.1 million Windows portable devices that have been tagged. It aims to tackle the issues faced by the information security community and offers a substantial, accessible, and varied dataset for training models. The authors include examples of application cases, such as model comparison, concept analysis, interpretable machine learning, and trend analysis. LightGBM is utilized as a benchmark model to showcase the efficacy of MalConv, a non-spatial deep learning model. The primary objective of the EMBER dataset and code is to propel the progress of machine learning research in the field of malware detection, foster creativity, and provide a standard for future research endeavors.Anderson and Roth (2018)

2.1.3 SOREL-20M: A Large Scale Benchmark Dataset for Malicious PE Detection

The study introduces the SOREL-20M dataset[3], which serves as a valuable tool for enhancing malware detection through the application of machine learning techniques. Containing almost 20 million files, this dataset overcomes the constraints of earlier datasets by offering an abundant number of training examples and neutralized malware for investigation. The sample is equipped with high-quality features, metadata, and tags, together with consumer search information, to enable a comprehensive evaluation. The system comprises a foundational model that utilizes PyTorch and LightGBM, demonstrating exceptional performance while still allowing for further enhancements. The dataset and distribution, along with the code and trained models, have the purpose of promoting a more adaptable and consistent framework for research in malware detection and making a substantial contribution to the area.Harang and Rudd (2020)

2.1.4 BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware

The authors introduced the BODMAS dataset[4] as a solution to the shortcomings of current open PE malware datasets. This collection offers recently collected malware samples (from August 2019 to September 2020) that are time-stamped, together with

[2]EMBER: `https://github.com/elastic/ember`
[3]SOREL-20M: `https://github.com/sophos/SOREL-20M`
[4]BODMAS: `https://github.com/whyisyoung/BODMAS`

carefully organized information about 581 malware families. A preliminary research was conducted to assess the influence of concept drift on both binary malware classifiers and multiclass malware family classifiers. The findings revealed difficulties arising from the ever-changing characteristics of malware, such as the emergence of novel families that impact classifiers. Investigations have been conducted on mitigation techniques, such as progressive retraining and training with new knowledge. The work emphasizes the necessity for open-world classifiers to tackle previously unidentified families and outlines unresolved issues for future research in malware analysis.

Information for the datasets is given in Table 1

Dataset	Malware Time	Family	# Families	# Samples	# Benign	# Malware	Malware Binaries	Feature Vectors
Microsoft	N/A (Before 2015)	●	9	10,868	0	10,868	◐	○
https://www.overleaf.com/project/6584a3129[illegible]ff108aef37[illegible]	01/2010-12/2018	◐	N/A	2,050,000	750,000	800,000	○	●
Ember								
UCSB-Packed	01/2017*-03/2018	○	N/A	341,445	109,030	232,415	●	○
SOREL-20M	01/2017-04/2019	○	N/A	19,724,997	9.762,177	9.962,820	●	●
BODMAS	**08/2019-09/2020**	●	**581**	**134,435**	**77,142**	**57,293**	●	●

Table 1: Public PE malware datasets. ○="not available"; ◐="partially available", ●="available".Yang et al. (2021)

2.2 Obfuscation Techniques

Code obfuscation encompasses a wide range of approaches that aim to increase the complexity of a program's source code or binary code, making it more challenging to comprehend. This may encompass (alongside encryption and compression):

- Utilizing obscure or arbitrary names when renaming variables, functions, and classes.
- Introducing superfluous or redundant code (often referred to as "dead code") with the intention of perplexing analyzers.
- Rearranging code instructions to intentionally interrupt the sequential logic.
- String encryption: Malevolent strings, such as URLs, API calls, or command-and-control server addresses, can be encoded as code. These strings undergo decryption at runtime, rendering it more challenging to ascertain their intended function during static analysis.

Control Flow Obfuscation: Control flow obfuscation refers to the manipulation of a program's logical flow in order to disrupt the functioning of analysis tools. Possible inclusions may encompass:

- Including superfluous conditional branches.
- Constructing intricate iterations.
- Employing opaque predicates, which are conditions that consistently provide either true or false outcomes, although may seem intricate.

API Function Obfuscation: Malware use obfuscated or dynamically resolved API function calls to evade detection. This refers to the utilization of indirect calls or the resolution of function addresses at runtime.Srivastava et al. (2008)

Dynamic Loading: Malware has the capability to dynamically incorporate supplementary code or modules from external sources, such as remote servers, while it is running. This can impede the static examination of the entire code.Cho et al. (2017)

Encoding and decoding: These approaches might be employed to obscure data or code. Typical encodings consist of Base64 or personalized methods. The data is decrypted dynamically when it is required.Fukushima et al. (2008)

2.2.1 Packing

Malicious software creators frequently utilize packing as a method of obfuscation to conceal their harmful programs. This procedure entails utilizing an algorithm to condense the initial malware code, resulting in a reduction in size. Additionally, it may include the integration of rudimentary encryption, so introducing an additional level of intricacy. In order to activate the compressed malware, a packing stub, which is a small and specialized piece of code, is generated to unpack the compressed payload. The stub is merged with the compressed payload, creating a unified executable file. Upon activation of the malware, the packing stub is programmed to automatically decrypt and decompress the payload, thereby running the specified malicious code. Packing serves three main purposes: firstly, it masks the true nature of the malware; secondly, it makes it difficult for security analysts and antivirus programs to directly analyze the executable because it is encrypted and compressed; and thirdly, it modifies the binary signature of the file, thus avoiding detection based on known signatures. In order to address this issue, security experts need to analyze the packing technique by executing the program in a controlled and secure environment. This will enable the stub to expose the underlying code, which can then be scrutinized for any malicious intentions.

List of well-known packers: UPX, Bobsoft, PECompact, Themida, ASPack, Enigma Protector, Molebox, VMProtect, RAR Compression, MPress, Armadillo, BoxedApp Packer

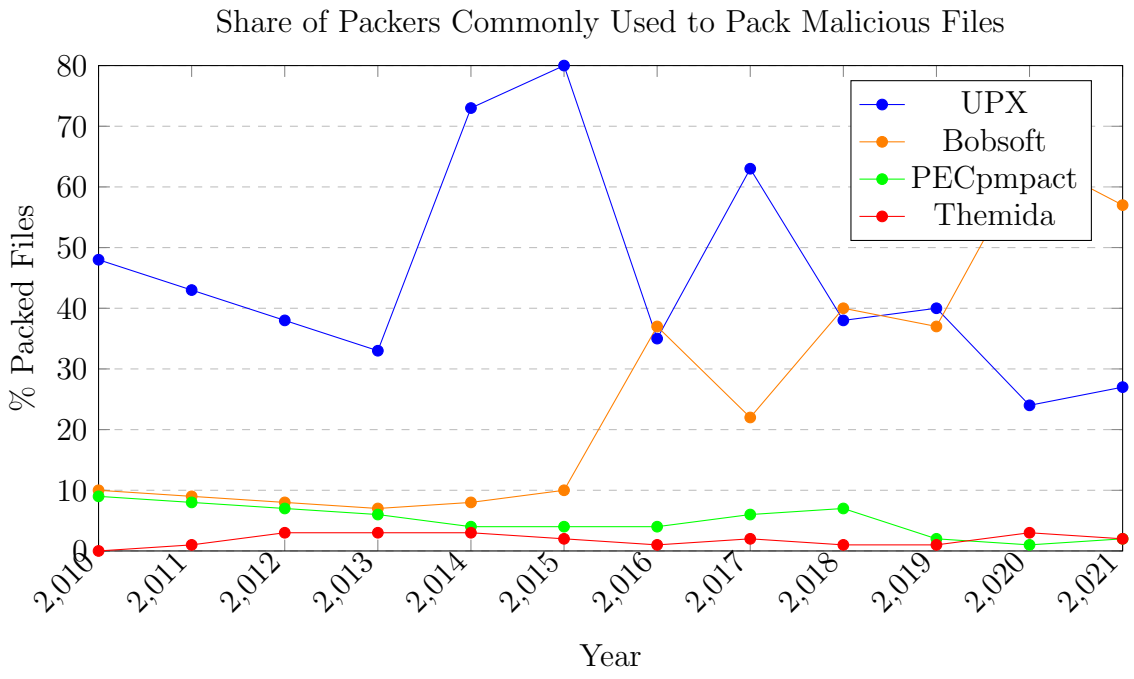

Figure 2: Share of packers commonly used to pack malicious files over the years.Muralidharan et al. (2022)

2.2.2 Encryption

Within the field of cybersecurity, individuals that create malware often employ encryption techniques to obscure their code, so effectively protecting the harmful payload from easy detection and analysis. The obfuscation process involves encrypting specific sections of

the code or the full payload using methods such as AES for symmetric encryption or RSA for asymmetric encryption. The essential elements necessary for deciphering the virus, namely the encryption key and the decryption procedure, are cleverly incorporated into the infection itself. When the virus is executed, it uses these pre-installed tools to decode and execute the hidden code. Encryption is strategically employed for several reasons: it conceals the code from security researchers and antivirus scans, evades detection systems that rely on recognizing known signatures of malware, and adds a level of intricacy that extends and complicates the analysis procedure. In order to understand this deliberate confusion, security researchers are assigned the responsibility of reversing the encryption. They do this by utilizing the malware's own decryption routine and key to expose the concealed code. This process allows them to uncover the actual behavior of the malware, which can then be studied in more detail.Omachi and Murakami (2020)

Well-known crypters: PELock, EXECryptor, SecureEngine, EXEShield, SecureCode, CodeLock

3 Methodology

This section describes the methodological framework designed for the investigation, outlining the systematic approach planned for assessing the impact of obfuscation techniques on ML-based malware detectors.

3.1 Data Collection and Preprocessing

The initial phase of the project involves the meticulous collection of diverse malware samples:

- The datasets, namely EMBER, SOREL-20M, and BODMAS, will be utilized to ensure a comprehensive representation of malware varieties.
- Standardization protocols will be applied to harmonize the data formats, addressing any discrepancies that may hinder the subsequent analysis.
- Preprocessing techniques such as normalization, encoding, and handling missing data will be implemented to optimize the datasets for efficient algorithmic processing.

3.2 Analysis of Malware Obfuscation Techniques

To dissect the obfuscation landscape, the following steps will be taken:

- Binary code and metadata from collected samples will be analyzed to identify prevalent obfuscation techniques.
- A framework will be established to classify these techniques based on complexity and the degree of analytical challenge they present.
- This exploratory analysis aims to discern patterns and methods commonly employed to evade detection.

3.3 Implementing Transfer Learning with MalConv

The project will adapt a pre-trained MalConv model to further our objectives:

- The MalConv model, proficient in capturing an extensive feature set from malware samples, will be fine-tuned to our collected datasets.
- Transfer learning techniques will be employed to adjust the model's parameters, aiming to enhance its sensitivity to obfuscation.
- Specific focus will be given to the adaptability of the model's final layers to our detection goals.

3.4 Evaluating ML-based Malware Detectors

Upon the completion of model adaptation, the following evaluation strategy will be enacted:

- The modified MalConv model's performance will be assessed against the processed datasets, with a focus on the detection of obfuscated malware.
- Accuracy and other relevant metrics will be calculated to measure the model's detection capabilities.
- The effectiveness of the model in identifying various obfuscation techniques will be thoroughly evaluated.

3.5 Anticipated Challenges and Proposed Solutions

In anticipation of potential challenges, the study will:

- Investigate and document the model's areas of vulnerability, particularly in detecting advanced obfuscation strategies.
- Develop a set of recommendations to overcome identified deficiencies, which may include algorithmic improvements and expanded feature integration.

The outlined methodology sets the stage for a rigorous inquiry into the resilience of ML-based malware detectors and their ability to cope with sophisticated evasion techniques employed by contemporary malware.

4 Design Specification

The Malware Obfuscation Detection project primarily use the MalConv model to analyze malware obfuscation strategies and their effects on Machine Learning (ML)-based detectors. MalConv is a specialized convolutional neural network used to classify raw executable files. It plays a crucial role in discovering and categorizing malware. This design specification provides a comprehensive overview of the structure, capabilities, and prerequisites of the MalConv model in relation to this project.

MalConv is built upon a deep learning architecture optimized for processing one-dimensional binary data. The model's key components include:

Embedding Layer: Transforms the unprocessed byte values of the executable into vectors of a specific size, making it easier to analyze byte-level patterns.

Convolutional Layers: Extract local patterns from the byte sequences that are embedded. These layers employ filters to systematically analyze the input data, identifying pertinent characteristics that are useful for categorization purposes.

Pooling Layer: Decreases the number of dimensions in the extracted features, improving the computational efficiency of the model.

Fully Connected Layers: Transform the combined characteristics into ultimate results for categorization.

Output Layer: Employs a sigmoid function to provide a probability score, which indicates the possibility of the input being malware.

5 Implementation

The MalConv model, which is a crucial element of the project "Analyzing Obfuscation Techniques for Evasion: A Case Study on Machine Learning-based Malware Detection," was implemented using the PyTorch module in the Python programming environment. The selection of this approach was based on its resilience and adaptability in managing intricate machine learning workflows.

The initial training phase of the model utilized the BODMAS dataset, which was selected for its extensive compilation of executable files, covering a wide variety of malware variants. The dataset played a crucial role in training the algorithm to identify different patterns linked to malware.

The study centered on examining the scores produced by the trained MalConv model using the sigmoid function3. The sigmoid function, which is crucial in binary classification tasks, produces a numerical value ranging from 0 to 1. This value represents the chance of an instance being assigned to a specific class.

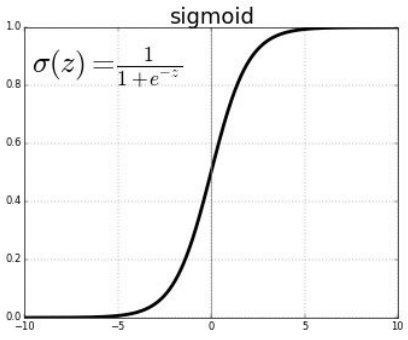

Figure 3: Sigmoid Function

A classification criterion of 0.5 was utilized: scores beyond this threshold indicate a greater probability of the file being harmful (malware), whilst scores below it indicate benign software (goodware). To conduct a detailed analysis, we employed a dual methodology to estimate the confidence ratios:

To get the confidence ratio for authentic goodware files, the sigmoid value was subtracted from 1. This offered a glimpse into the model's confidence in accurately categorizing

files as non-malicious. On the other hand, when it came to files categorized as malware, the sigmoid score was utilized as a direct indicator of the model's certainty in its classification.

In order to replicate real-life situations and improve the resilience of the model, the implementation incorporated several compression and packaging techniques, including RAR Compression, Themida, and PECompact. These technologies are frequently employed in the process of obscuring malware, hence creating a practical environment to assess the model's efficacy. In addition, the model underwent testing on files that were protected using various encryption technologies, such as PELock and EXECryptor. This stage was essential in assessing the model's capacity to differentiate between really encrypted files and those that employ encryption as a method of concealing information.

6 Evaluation

The main aim of this research was to assess the efficacy of various obfuscation tactics in making the detection process more complex. Our objective was to investigate the impact of compression and encryption algorithms, which are commonly used as obfuscation techniques, on the effectiveness of machine learning-based malware detectors.

The accuracy of the previously trained MalConv model may vary due to relative changes in the dataset and model parameters. The accuracy value of 0.94, as reported in our primary article, is also corroborated by the findings of this investigation. These results are presented in Table 2 below:

	MalConv		Byte n-grams	
Test Set	Accuracy	AUC	Accuracy	AUC
Group A	94.0	98.1	82.6	93.4
Group B	90.9	98.2	91.6	97.0

Table 2: Performance comparison of MalConv and Byte n-grams models.

Our research has uncovered a crucial discovery: obfuscation techniques present significant difficulties in identifying malware. However, there is a noticeable disparity in the effectiveness of compression and encryption algorithms. Our research suggests that encryption algorithms are more successful than compression algorithms in avoiding discovery.

The utilized obfuscation strategies are listed below and the outcomes are shown:

6.1 Packing or Compression

6.1.1 UPX

UPX achieved superior performance in compressing both benign and harmful data compared to other compression techniques. The accuracy ratio, obtained by comparing and processing the sigmoid score results from our project with the genuine dataset values (Malware: 1, Goodware: 2), is 0.90.

6.1.2 RAR Compression

RAR Compression has shown only moderate effectiveness in compressing both benign and harmful data. The algorithm achieved an accuracy rate of 0.92, indicating that it was less successful than UPX.

6.1.3 PECompact

While the compression algorithm created using PECompact has a slightly more detrimental impact on the success of hiding, specifically the accuracy rate, when compressing larger files compared to other files, it still does not achieve a superior rate compared to RAR in this algorithm. The accuracy value was determined to be 0.93 when applied to high-dimensional data. However, it was generally noticed that this value had minimal influence and was unable to decrease the accuracy.

6.1.4 Themida

While the application of the Themida method for compression had a slight detrimental impact on the sigmoid score, it did not alter the accuracy rate in the test data.

6.2 Encryption

Regarding encryption algorithms, there is a distinct variation in the effectiveness of obfuscation methods.

6.2.1 PELock

The impact of the PELock encryption technique on the accuracy of machine learning models used to detect malware is noteworthy, as both malicious and benign software encrypted using this algorithm have been found to have a substantial effect. Values that closely approached the threshold value utilized for classifying the value on the sigmoid score were noticed. Consequently, the algorithm successfully decreased the accuracy rate to 0.82.

The sigmoid score values produced from the PELock encryption method are presented in Figure 4 below.

6.2.2 EXECryptor

After encrypting the identical data using EXECryptor, it was noted that the accuracy rate dropped to 0.85. A comparable outcome was achieved with PELock.

6.2.3 EXEShield

The accuracy rate achieved by encryption using EXEShield was comparable to that of PELock and EXECryptor. The observed value was determined to be 0.86.

6.2.4 Final results with some graphs

Below are graphs illustrating certain results. Graph 4 illustrates the occurrence of malware scores in six distinct selected files. Graph 5 illustrates the disparities in accuracy between the primary dataset and the model's outcomes for UPX PELock and RAR.

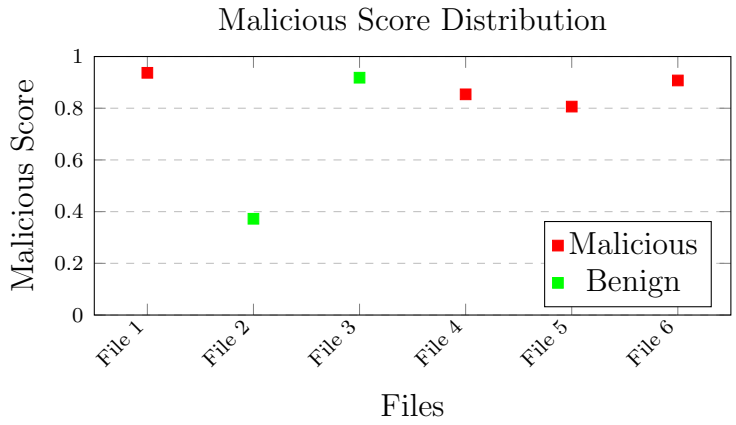

Figure 4: Malicious Score Distribution for various files. Red: real malware, Green: real goodware

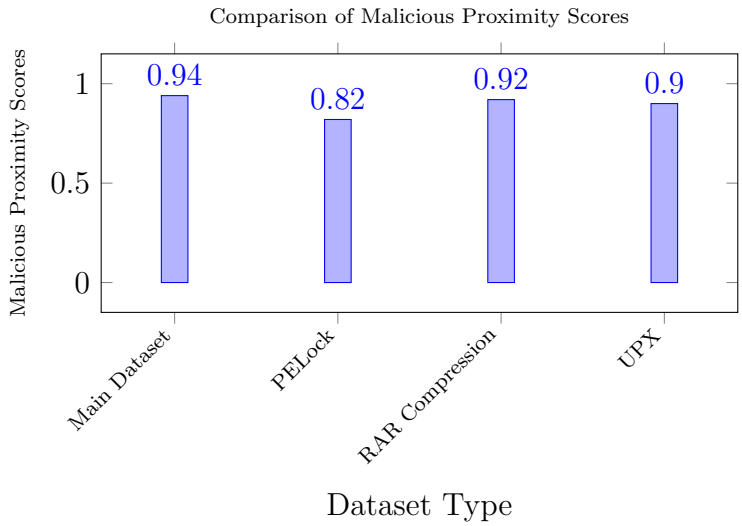

Figure 5: Comparison of Malicious Proximity Scores Across Datasets

6.3 Discussion

Due to the continuous development of genuine harmful software, security issues for personal computers frequently emerge as a consequence of these efforts. This presents other complications, such as the operating system eradicating this data. To ensure the advancement of the study and achieve reliable and unambiguous outcomes, it is crucial to utilize a larger dataset that closely aligns with the original data. It is necessary to devote certain resources for this purpose, and the malicious program can be installed on a machine that is both isolated and safe. The frequent deletion or alteration of data by the defensive system enhances the subjectivity of the research findings.

7 Conclusion and Future Work

The primary research objective of this study was to assess the impact of malware obfuscation strategies on the effectiveness of machine learning-based malware detectors. We want to analyze the intricacies caused by obfuscation techniques and assess the re-

silience of static machine learning detectors against these obstacles. After conducting thorough research, we found that obfuscation techniques indeed make it more difficult to detect malware. However, encryption algorithms are far more effective than compression strategies in avoiding detection.

Our research revealed that compression technologies like UPX and PECompact, while somewhat effective, were not as successful as encryption methods in concealing malware. The PELock encryption significantly reduced detection accuracy, highlighting its effectiveness as an obfuscation approach. These results emphasize the urgent requirement to enhance machine learning-based malware detection capabilities in order to match the complexity of obfuscation tactics.

The ramifications of our study have extensive reach in the field of cybersecurity. The effectiveness of encryption, as opposed to compression, in disguising signals indicates to both malware creators and cybersecurity experts the changing nature of malware detection and prevention. It also emphasizes the need for ongoing improvements in detection algorithms to overcome these tactics of concealment.

Although the study provides insights into important areas of malware obfuscation, it does have several drawbacks. The dynamic nature of malware growth is a constantly changing challenge, and the intervention of protective systems can occasionally distort outcomes. Hence, conclusions drawn from a limited dataset may not comprehensively represent real-life situations.

To enhance future research, an expansion of this study could incorporate a comprehensive approach that combines dynamic analysis with static machine learning-based identification. Exploring the possibility to construct detection technologies that mimic the behavioral characteristics of malware is worth considering. Additionally, partnering with experts in the sector to obtain a wide range of malware samples could enhance the accuracy of detection models and open up possibilities for commercialization. The findings obtained from this research provide a promising opportunity for commercialization, specifically in the development of sophisticated machine learning-based detection systems that may be included into commercial cybersecurity products. The market consistently requires real-time, adaptable solutions for detecting malware, and this study establishes a basis for the development and commercialization of such solutions.

A subsequent research endeavor could shift its focus towards:

- Investigating adaptive machine learning algorithms that proactively adapt to emerging obfuscation strategies.
- Creating hybrid detection systems that use static and dynamic analysis to achieve thorough malware identification.
- Exploring the application of artificial intelligence in predictive cybersecurity, which could result in the early identification and elimination of malware threats.
- Assessing the cost-effectiveness of different detection strategies to guide investment choices in cybersecurity infrastructure.

The research has clearly identified the existing difficulties posed by malware obfuscation and the potential methods for improving machine learning-based detection systems. This paper establishes a foundation for future research focused on enhancing the robustness of cybersecurity systems in an era characterized by growing digitalization.

References

Anderson, H. S. and Roth, P. (2018). Ember: an open dataset for training static pe malware machine learning models, *arXiv preprint arXiv:1804.04637* .

Cho, T., Kim, H. and Yi, J. H. (2017). Security assessment of code obfuscation based on dynamic monitoring in android things, *Ieee Access* **5**: 6361–6371.

Fukushima, K., Kiyomoto, S., Tanaka, T. and Sakurai, K. (2008). Analysis of program obfuscation schemes with variable encoding technique, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* **91**(1): 316–329.

Harang, R. and Rudd, E. M. (2020). Sorel-20m: A large scale benchmark dataset for malicious pe detection, *arXiv preprint arXiv:2012.07634* .

Muralidharan, T., Cohen, A., Gerson, N. and Nissim, N. (2022). File packing from the malware perspective: Techniques, analysis approaches, and directions for enhancements, *ACM Computing Surveys* **55**(5): 1–45.

Omachi, R. and Murakami, Y. (2020). Packer identification method for multi-layer executables with k-nearest neighbor of entropies, *2020 International Symposium on Information Theory and Its Applications (ISITA)*, IEEE, pp. 504–508.

Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B. and Nicholas, C. K. (2018). Malware detection by eating a whole exe, *Workshops at the thirty-second AAAI conference on artificial intelligence*.

Srivastava, A., Lanzi, A. and Giffin, J. (2008). System call api obfuscation, *Recent Advances in Intrusion Detection: 11th International Symposium, RAID 2008, Cambridge, MA, USA, September 15-17, 2008. Proceedings 11*, Springer, pp. 421–422.

Yang, L., Ciptadi, A., Laziuk, I., Ahmadzadeh, A. and Wang, G. (2021). Bodmas: An open dataset for learning based temporal analysis of pe malware, *2021 IEEE Security and Privacy Workshops (SPW)*, IEEE, pp. 78–84.