SMU Data Science Review

Volume 7 | Number 2 Article 4

2023

Static Malware Family Clustering via Structural and Functional Characteristics

David George
Southern Methodist University, davidg@mail.smu.edu

Andre Mauldin
Southern Methodist University, amauldin@mail.smu.edu

Josh Mitchell
Southern Methodist University, joshmitchell@mail.smu.edu

Sufiyan Mohammed
Southern Methodist University, sufim@mail.smu.edu

Robert Slater
Southern Methodist University, rdslater@mail.smu.edu

Static Malware Family Clustering via Structural and Functional Characteristics

Josh Mitchell, Sufiyan Mohammed, Andre Mauldin[1], David George, Robert Slater[2]
[1]Master of Science in Data Science, Southern Methodist University, Dallas, TX 75275 USA
[2]Faculty, Data Science Department
Southern Methodist University,
Dallas, TX 75275 USA

Abstract. Static and dynamic analyses are the two primary approaches to analyzing malicious applications. The primary distinction between the two is that the application is analyzed without execution in static analysis, whereas the dynamic approach executes the malware and records the behavior exhibited during execution. Although each approach has advantages and disadvantages, dynamic analysis has been more widely accepted and utilized by the research community whereas static analysis has not seen the same attention. This study aims to apply advancements in static analysis techniques to demonstrate the identification of fine-grained functionality, and show, through clustering, how malicious applications may be grouped into associated family types. The scope of this research is focused on malicious software utilizing the Portable Executable ("PE") file format for Microsoft Windows operating systems.

1 Introduction

Malicious software, herein referred to as malware, is one of the largest risks to safe and secure internet usage. In 2022, more than 30 million new malware samples were found, and malware authors created more than 316 thousand malware samples daily. (Database and Network Journal, 2022). As a result, certain aspects of the malware problem have been studied at length. Many of these studies involve different combinations of detection, classification, and clustering either through dynamic analysis or hybrid approaches (those involving a combination of static analysis and dynamic analysis) (ALGorain & Clark, 2022) which involves executing the malware for classification and clustering. The dynamic approach also requires more resources than the static approach (classifying and/or clustering without executing the malware).

There are several issues when attempting to cluster malware into familial groups solely based on static analysis. First, the presence of an industry-accepted ground-truth dataset is unavailable. Datasets have been generated through honeypots, public databases, the collection and processing of cyber threat bulletins, or other means (Smith, et al, 2017). However, these datasets are insufficient for the familial classification task due to a lack of family labels and insufficient information as to the

internal state of the application. Secondly, data transformations in the form of packers, obfuscation, and encryption are commonly applied to malware to prevent or hinder analysis and detection. These data transformations can be of arbitrary complexity and depth and have, up to this point, discouraged meaningful attempts to statically categorize malware. Finally, malware is generated at an extraordinary rate. Malware authors and cyber criminals are in a perpetual survival-of-the-fittest struggle against one another and security vendors. These issues have a compounding effect when attempting to classify and cluster malware into families and are leading reasons why research has been primarily devoted to the dynamic analysis arena.

Unfortunately, dynamic analysis is not a panacea. Dynamic analysis systems are complex and involve a significantly higher resource utilization footprint. The dynamic approach only records what happened when the malware was executed (what it did). This does not show other potential harm the malware could cause (what it can do). Furthermore, malware regularly detects dynamic analysis systems such as virtual machines and sandboxes. This can cause the malware to alter its behavior and change the observable characteristics recorded by the dynamic analysis system. These changes in behavior can be imperceptible to the system.

Static and dynamic analysis are two sides of the same coin. Each has advantages and disadvantages, and neither are complete. This research aims to identify key observational metrics with respect to malware family clustering through static analysis and enabling interested parties to cluster malware into family groups without executing the malware.

This research will enable interested parties to cluster malware into the known families without executing the malware. This will save time and the cost of acquiring a sandbox to execute the malware, and the cost of the computing power required to extract data from the sandbox to perform data analysis. Additionally, this research will reduce the cost of analyzing multiple malware samples of the same family and reduce the possibility of alerting threat actors to analysis by not using public data sources or executing the malware.

The goal of this research is to use the static approach to cluster malware into the known malware families. This system's results are expected to augment the dynamic approach and/or provide interested parties with a much less expensive alternative to using the dynamic approach.

2 Literature Review

In order to identify and mitigate the consequences of malware, analysis techniques are crucial. The three primary methods for analyzing malware are static, dynamic, and hybrid. Static analysis is a quick and resource-effective technique with limitations in detecting new viruses. It entails looking at the source code or executable file without running it. Dynamic analysis, on the other hand, involves executing the malware in a controlled environment to observe its behavior, and although it provides more comprehensive analysis and can detect novel malware, it requires more resources

and is time-consuming. Hybrid analysis combines static and dynamic methodologies, combining their advantages to produce a more potent method of malware analysis.

The EMBER dataset is a valuable resource for researchers looking to better understand use cases for malware classification. In their study, Anderson and Roth (2018) provide a detailed exploration of this dataset, which contains a comprehensive collection of Windows executable files, both malicious and benign. The authors describe the dataset in detail and highlight the value of its inclusion of benign executables, which eliminates the need for researchers to generate their own benign data for testing purposes. The focus of the study is on training machine learning models to detect Windows malware executables, making it a useful resource for researchers working in the field of malware analysis.

2.1 Static Analysis

Static analysis is primarily a method of analyzing software without actual execution- it involves examination the source code or executable binary. The main advantage of it being that it can be performed quickly and with less resources as compared to dynamic analysis. (Jusoh, Rosmalissa, et al., 2021) reviewed various techniques and approaches that were applied to the analysis of Android malware. With respect to static analysis, signature and source code review methods were utilized. These techniques involved examining the code to detect known patterns of malicious behavior or code structures known to be malware families. Accurate feature selection was identified as a critical component of the system and obfuscation was identified as a hurdle, and although the methods were useful, they have been known to be limited in scope and have difficulty detecting novel malware.

To aid precise malware identification, Keong Ng, Jiang, et al. (2019) examined the deep embedded clustering method using the Virus dataset from the UCI repository. This method used autoencoders, PCA, and KNN to identify patterns and similarities in the binary files and group them into different malware libraries. Use of 3 layers of encoders/decoders was found to have the best results and outperformed K-means methods. Little work had been done on deep learning representations of malware clustering prior to the authoring of this study, and the advantage of this over source code review is its ability to detect novel malware that does not necessarily have a known code structure.

Another promising approach to static detection is through Kim, Sangwon, et al (2022) which introduces a fully automated labeling system based on AVClass2. The tool extracts family information from antivirus and measures the frequency of label occurrence per sample family. The system then requires more than a million labels to build a graph using the same accuracy metric at AVClass2. The biggest benefit is that no prior knowledge is required to run this approach. As the author aims to efficiently extract existing malware data from antivirus engines and use those same samples to train ML algorithms, the accuracy is only based on the clustering and not the appropriateness of the labels.

On a similar note, Sebastian and Caballero's (2020) malware tag extraction tool uses AV labels (Detection labels by anti-virus engines) that contains information about the malware. The malware tags can be indexed to extract existing data from anti-virus engines to train ML algorithms. This method does however rely on the accuracy of the

AV labels themselves to train the model and may not capture the full behavior of the malware.

Finally, Siwach, Costa, and De Nicola (2021) introduced a rule-based feature specification language, Symbolic Feature Specification Language (SFSL). The rules are based on the behavior of the malware sample- if the behavior matches the defined rules, the result will "generate feature vectors for a machine learning classifier." The process includes "symbolizing" execution of the malware sample without actual execution- maintaining the process' safety.

2.2 Dynamic Analysis

Execution in a controlled environment is the base premise of dynamic analysis. The main advantage of this method is being able to record the program's exact behavior, which can aid in the discovery of novel or unknown malware. Although it is typically more comprehensive than static analysis, dynamic analysis does require more resources and can be more time-consuming. This analysis has been executed in a variety of techniques and Smith et al. (2017) talk about the difference between malware detection and behavioral analysis, the latter of which excludes watching malware behavior. To increase the effectiveness of malware detection and classification, the authors emphasize the necessity of observing malicious activity. This study also discusses the use of sandboxing as one of the main techniques of dynamic analysis. This involves using a controlled environment in which malware can be safely executed without fear of harming the host system. The behavior of the malware is observed, and results are then analyzed to identify malicious behavior.

Nikolopoulos, Stavros D, and Polenakis (2017) defined a directed acyclic graph to represent the functionality of a malware sample by capturing API calls made by the sample during execution. As the API calls interface between the program and operating system, performing various essential functions, capturing them makes it possible to understand the behavior and understand malware activity. As this method captures behavior, it is possible to detect novel malware and is especially useful in detecting malware techniques that change the code of the malware with each execution.

2.3 Hybrid Analysis

Hybrid analysis combines elements of both static and dynamic analysis to overcome their individual limitations and provide a more thorough understanding of malware. This process entails combining both code examination techniques with execution of the code in a controlled environment. By utilizing the advantages of both methodologies, hybrid analysis can detect malicious patterns that might not be apparent with a single method alone. Aside from utilizing a sandbox, other techniques have been developed. Rizvi, et al. (2022) performed unsupervised clustering with neural networks to group malware samples based on features extracted from the code. The dataset was collected from high and low interaction honeypots and enterprise networks and labels were added based on VirusTotal search results. After analyzing files with PEStudio to determine if they were packed or obfuscated, they are then executed in a sandbox and the behavior is captured.

Another hybrid system proposed was through Ghouti and Imam (2020) in which binary values of a malware sample were converted into greyscale images and then classified using image processing techniques. The images were trained on a SVM model on the PCA data of the images. The authors used the EMBER dataset along with others to complete the analysis. A major advantage of this approach was its ability to provide a new dimension to the analysis through visual similarities, giving it an edge when dealing with obfuscated samples. However, the authors acknowledged the limitations of this approach, including lack of feature interpretability, high computational requirements, and impact of image resolution on classification accuracy. Despite the single dimensionality and novelty of the technology, the results of the study showed that there is potential in this method.

It has been established that each of the three analysis techniques- static, dynamic, and hybrid have their own advantages and disadvantages. Static is ideal for quick and resource efficient analysis but limited in its ability to detect novel malware. Dynamic analysis allows for more comprehensive analysis and potential detection of novel malware through execution in a controlled environment but suffers from being far more time and resource intensive. A hybrid approach combines both static and dynamic analyses and can provide a more efficient solution. Additionally, using high-level representations of malware can help in identifying malware clusters through static analysis, potentially making it even more efficient than the hybrid approach.

Hypothesis

Malware can be clustered into family groups using features which fall into three categories. Features describing the malware's shape, those describing the internal structure, and features describing its capabilities. The shape of the malware is ascertained by utilizing features found in the EMBER dataset. Internal structure is extracted via measurements of graph node centrality. Capabilities are a higher-level representation of the functionality of an application and include: the ability to download a file, ability to upload a file, ability to encrypt a hard drive etc. Capability features were extracted using the Mandiant CAPA tool. This is a tool that has rules designed to identify specific malware capabilities.

Analysis proves that this methodology can be used to identify malware clusters using a static approach. This is an effective augmentation to the dynamic approach. While the dynamic approach exposes what happens during malware execution, it does not expose what *could* happen. The benefits of this approach lie in its cost saving, in terms of dollar cost and required processing power, and the identification of latent functionality.

3 Methods

This study follows a systematic process for malware sample analysis, which involves the collection of malware samples, feature extraction, dataset formation, and subsequent clustering. A ground truth dataset is established by collecting malware

samples from VX-Underground, MalwareBazaar, and VirusTotal. Within these databases, crowd-sourced and automatic labeling is applied. Ground truth is established through the validation of these family labels by processing all malware samples with configuration extraction utilities designed for each specific family. Features from the ground truth samples are then extracted using Radare2, CAPA, and the PageRank algorithm. Once the dataset is established, clustering analysis is leveraged to verify the established methodology by way of the labels provided by the ground truth collection. The methodology applied in this research is delineated in the following subsections:

3.1 Sample Collection

Malware samples were obtained from community databases, namely VX-Underground, MalwareBazaar, and VirusTotal. These malware samples are collected through a variety of sources, and the databases provide a mechanism whereby samples can be submitted and tagged by contributors. In general, the tags enable identification and family association of the samples within these databases. However, malware families contain multiple variants and components which obscure the relationship between a sample, the sample's purpose, and the family. Thus, the samples must undergo additional processing to identify those which truly represent the family.

After collecting the samples, the extraction scripts try to extract configuration data from the sample by running YARA rules and configuration extraction utilities. These tools are specifically designed to identify and extract configuration data from the malware family. As the primary focus of this study is on samples that are an actual representation of the family in which they were tagged, the research team has brought together multiple publicly available configuration extraction tools and tested them against each sample. These identification and extraction utilities attempt to extract the malware settings, and, as such, are “keys” for a specific malware family. If the key does not work, it is not a clean representation of the malware family.

The research team used the PEiD packer detection library to remove all samples identified as having a packer or obfuscation applied to them. Packed samples are deemed not useful as the packer is a separate application than the malware sample. As such, all features extracted would not be representative of the malware family, but rather, would be describing the characteristics consistent with the packer application. Conceptually, software packers are akin to Matryoshka dolls.

3.2 Feature Collection

Similar to the EMBER, UCI Repository (Anderson and Roth, 2018, ALGorain and Clark, 2022), the research team identified features related to the Portable Executable (PE) file format. These features describe external aspects of PE applications and have been used in malware classification and categorization research. However, most features collected by the research team describe internal aspects of PE applications.

These features can be separated into two groups: those that describe the application's structure and those that describe its functionality.

To characterize the structure of an application, the research team focused on collecting features from the top 100 functions which were identified by building an adjacency matrix of the interprocedural control-flow-graph (the graph representing the relationships between functions, CFG). This matrix was used to generate scores from the PageRank algorithm, a Google innovation that ranks the importance of web pages returned in search engine results (Murrugarra, David, et al., 2016).

Describing the functionality of an application requires a higher-level understanding of the various components within, and the relationship between, them. To that end, the research team utilized CAPA, Mandiant's open-source tool for describing the tactics, techniques, procedures (TTP), and capabilities present within an application. CAPA provides a mechanism to list known capabilities present within the malware sample and map them to the MITRE adversarial tactics, techniques, and common knowledge (ATT&CK) framework and the malware behavior catalogue (MBC).

3.3 Data Description

The dataset for this study contains 854 features with 318 depicting malware capabilities, 503 describing the internal structure of the application, and 35 features which represent the external shape of the malware. For example, " capa_check_for_sandbox_and_av_modules" is a feature that describes a capability, while "function_0_func_size" describes the size of a function in the malware code. The size is a property of the malware structure.

This study tests whether the features used for clustering will work. Figure 1 is used to describe the rationale behind the methods.

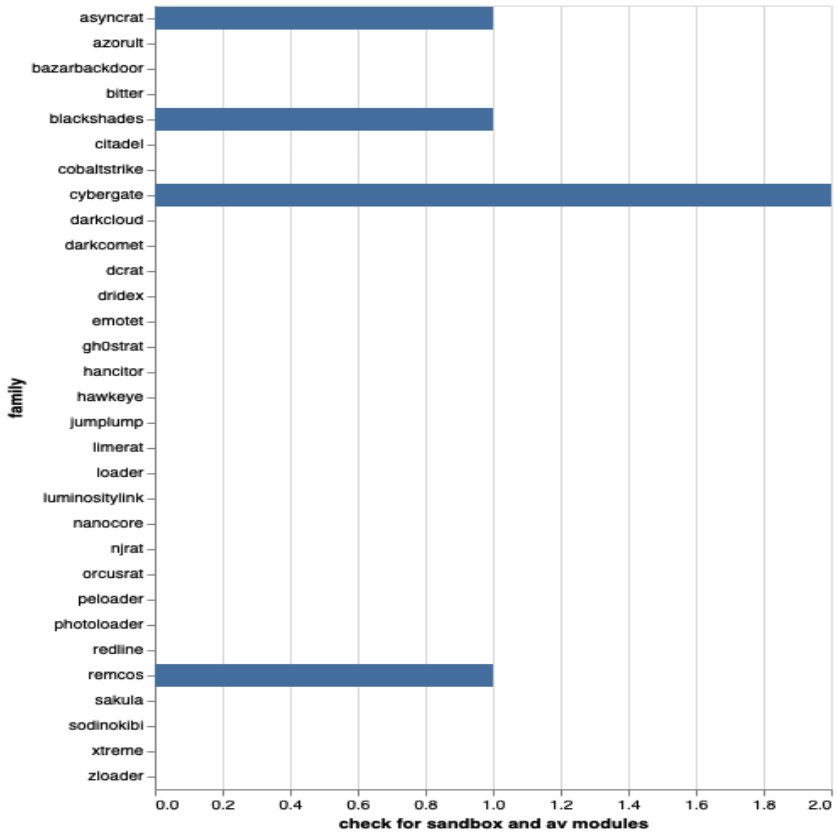

Figure 1. Y-axis: The number of times "capa_check_for_sandbox_and_av_modules" was executed. X-axis: The malware family.

Observe in Figure 1 that the cybergate family executed "capa_check_for_sandbox_and_av_modules" twice. There are 45 records in this dataset where this feature is executed twice. The research team believes that if a human can make such a discovery using one feature, a machine can cluster malware samples into families with great accuracy, using all the features in the dataset.

3.4 Clustering

Before clustering malware samples, dimensionality reduction was used to find the most important features. Traditionally, linear dimension reduction techniques like Principal Component Analysis are used but a nonlinear and non-deterministic method was more appropriate and produced better clustering results later. So instead of using Principal Component Analysis, Uniform Manifold Approximation and Projection

(UMAP) was used. UMAP is an alternative dimensionality reduction technique that uses Riemann geometry instead of constructing orthogonal vectors like PCA. Stratified shuffle split was used to validate the model and tune the hyperparameters for UMAP, where 80% of the data was used for training the model and 20% was set aside for testing. After dimensionality reduction, the clustering algorithm used was HDBSCAN. The difference between HDBSCAN and DBSCAN is that HDBSCAN can immediately determine the optimal number of clusters needed and can work with clusters of different densities. Originally, K-Means clustering was used but after tuning the number of clusters with the elbow method, the accuracy, precision, and recall were poor. HDBSCAN is a clustering method that finds clusters of varied sizes without having to specify a distance threshold first and can be used to predict future cluster membership, unlike DBSCAN. For this use case, optimal hyperparameters for HDBSCAN were identified as min samples of 2, min cluster size of 3, excess of mass cluster selection method, the Manhattan distance metric, and a cluster selection epsilon of 0.01.

4 Results

4.1 Metrics

Family Name	Quantity
sodinokibi	464
darkcomet	224
njrat	156
remcos	71
nanocore	66
limerat	51
dcrat	38
systembc	35
gh0strat	24
quasarrat	22
asyncrat	16
orcusrat	15
redline	13
blackshades	10

Table 1: Family Distribution

Test Mean Accuracy	Test Weighted Mean Precision	Test Weighted Mean Recall	Mean Percent Classified
97%	97%	97%	96%

Table 2: Overall Metrics

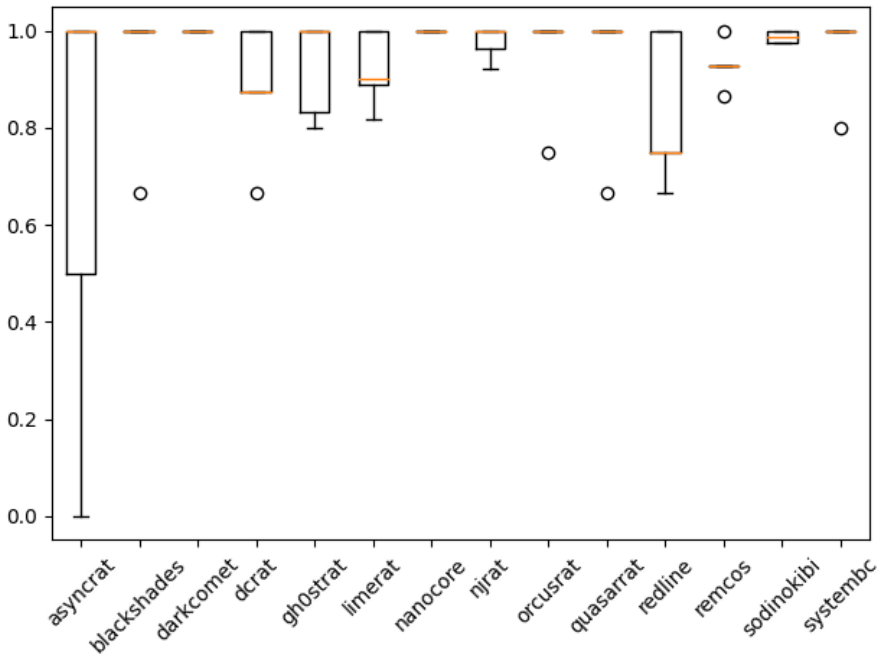

Figure 2: Multiclass Test Precision

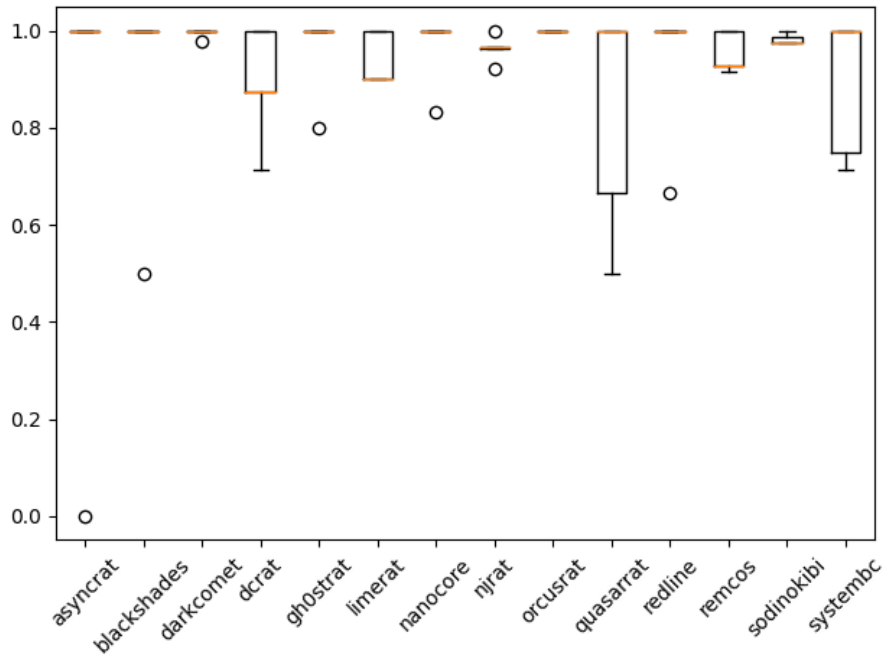

Figure 3: Multiclass Test Recall

For each cluster it is desirable to have high accuracy. Specifically, high cluster family homogeneity. If, for example, a sample is not in the asyncrat family but is clustered into a grouping mainly comprised of that family then an error would have occurred. Using 5-fold stratified 80/20 shuffle split, the mean accuracy on the test set was 97% with, on average, 96% of the samples successfully clustered. The mean cluster size obtained by this methodology is 7.57, reducing the 1205 total samples down to 152 clusters.

4.2 Cluster Visualization

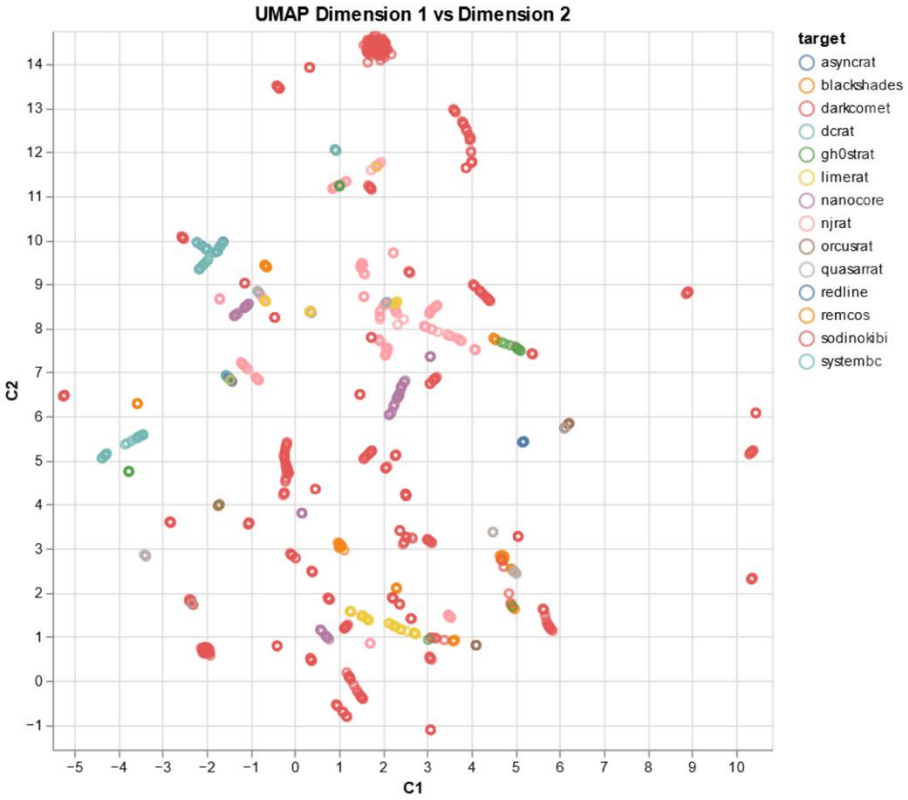

Figure 4: Malware Samples Colored by Family

After performing dimensionality reduction, the first and second UMAP components are plotted as x and y respectively. Notice the many different clusters in Figure 4. This is a 2-dimensional representation of multi-dimensional data. It is not possible to visualize more than three dimensions. However, the usefulness of this plot is that samples that are alike are near each other. The location and color of each sample within a cluster are important. Observing the clusters reveals that most of the points are near other points of the same color.

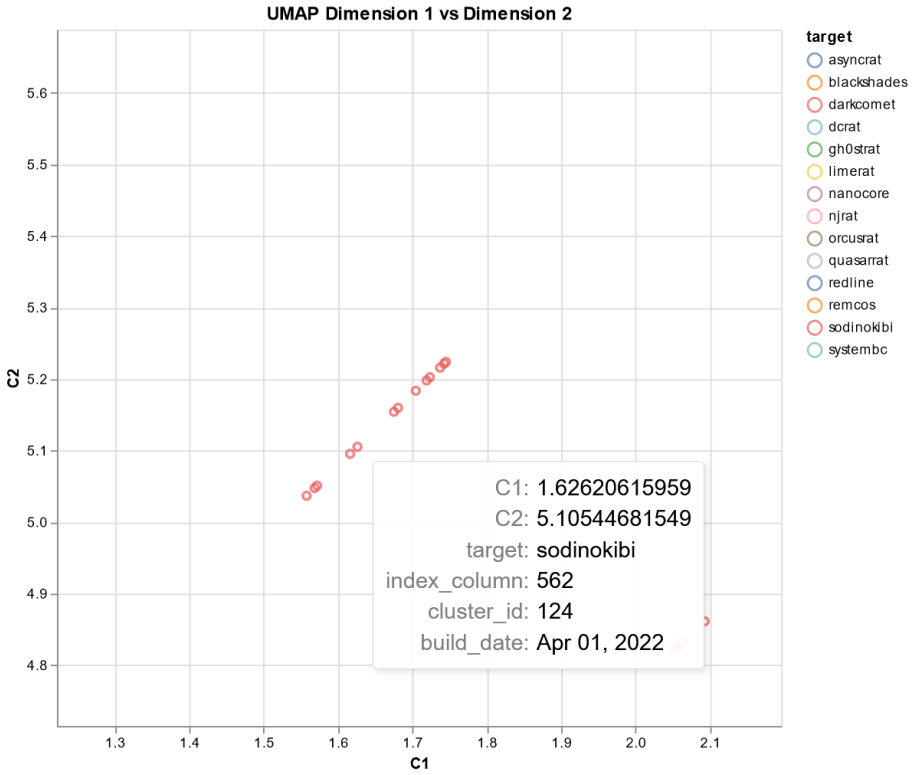

Figure 5. Malware Sample 'sodinokibi' Family

Zooming in on a specific cluster, Figure 5, reveals that the malware samples form lines. This is an interesting result that occurs across multiple families. More investigation is required to understand why these lines are formed; anecdotal evidence indicates that this visual representation of a change in the malware family with respect to time. However, the main point of interest for this study is that the samples are clustered accurately and there is strong visual evidence.

4.3 Capabilities

Malware Sample	Execute anti-debugging instructions	Check for Unmoving mouse cursor	Check HTTP status code	Encode data using Base64	Encode data using XOR
648	0	1	1	1	1
649	1	0	0	1	13

Table 3 Malware Samples from the 'sodinokibi' Family

Table 3 shows two malware samples from the dataset with five of the many CAPA rules. These rules are actions that malware samples can execute. The numbers in the CAPA rules columns represent the number of times the rule was observed in the sample. Although these samples did not trigger the same rules, they are clustered into the same family. This is evidence that the robustness of the methodology enables clustering even though exact matching is not present.

5 Discussion

The research conducted and the literature reviewed so far presents a comprehensive understanding of how malware files can be analyzed and classified based on their similarities and unique characteristics. The advent of advanced static analysis tools such as CAPA is radically changing the approach data science teams take in analyzing and classifying malware. An intrinsic advantage of these advancements is the ability to identify clusters of malware samples which have similarities in external shape, internal structure, and capabilities. This represents a significant stride towards understanding both the scope and intent of various malware types that pervade across the digital landscape.

These sample clusters provide us with groups of malwares that display high similarity within their nature and functionalities. Through this clustering approach, we can decipher what these grouped samples are designed to accomplish, whether it involves downloading or uploading a file, encryption of a hard drive, or other potential functionalities. Clustering also provides insights into their unique objectives such as executing unauthorized system commands, encrypting data, or infringing upon user privacy. These capabilities directly correlate to the potential damage these malware files can inflict upon execution.

Through understanding these clusters, we gain a critical insight into the severity of the potential damage these malware files can cause. For instance, if a cluster of malware

files can encrypt a hard drive or have system control capabilities, it signifies a serious threat to data security and system integrity. With this knowledge, we can prioritize the development and deployment of countermeasures targeted at these specific threats. Examples of this can include improving system defenses, implementing more robust data backup, and bolstering intrusion detection systems. Moreover, increased awareness of such threats would allow for targeted user education and understanding of digital hygiene practices.

Another pivotal advantage is the efficiency that clustering introduces to malware analysis. Considering a scenario where 50 malware files are grouped into a cluster based on similar functionality, an in-depth analysis of a single representative file could offer insights into the behavior and objectives of the entire group. This saves considerable time and computational resources while avoiding unnecessary duplication of work, leading to an accelerated response time in combating threats.

Despite its effectiveness, we must acknowledge that this clustering approach has its limitations. In the constantly evolving landscape of cyber threats, malware creators continue to innovate obfuscation and encryption techniques to bypass detection systems. Consequently, our static analysis tools and methodologies must continually adapt and, to maintain their reliability in detecting and classifying these threats, should be integrated into pre- and post-processing components of a hybrid analysis pipeline. This would decrease overall analysis costs and facilitate more flexible and adaptive strategies that can keep up with the advancements in malware technology.

6 Conclusion

The aforementioned methodology shows promise. Although no industry standard ground truth dataset with familial labels exists, this research hopes to have provided an outline that the research community at large can use. Furthermore, the goal of extracting features from malware samples, generating a ground truth dataset, and performing cluster analysis on that dataset with a high level of accuracy was achieved. Additionally, the data science team was able to perform this analysis without executing any of the malware samples, raising the bar for what can be accomplished through static analysis.

Clustering malware samples using this methodology, with the accuracy levels mentioned under Results, the research shows that for a given cluster, everything within the cluster can be generalized to the same malware family with 97% accuracy. Generally speaking, this methodology may be applied to cluster any binary code of sufficient size. Future work may involve adding more CAPA rules, structural features, and extending this approach to binary code formats for different operating systems.

References

1. ALGorain, Fahad T., and John A. Clark (2022). Bayesian Hyper-Parameter Optimisation for Malware Detection. Electronics (Basel), vol. 11, no. 10, 2022, p. 1640–, https://doi.org/10.3390/electronics11101640
2. Anderson & Roth (2018). EMBER: An Open Dataset for Training Static Malware Machine Learning Models. ArXiv:1804.04637v2 [cs.CR] 16 Apr 2018
3. Chee Keong Ng, Frank Jiang, Leo Yu Zhang, Wanlei Zhou. (2019). Static malware clustering using enhanced deep embedding method. https://doi.org/10.1002/cpe.5234
4. Chinmay Siwach, Gabriele Costa, Rocco De Nicola (2021). Enhancing Malware Classification with Symbolic Features. ITASEC'21: Italian Conference on
5. Cybersecurity, April 07–09, 2021, all-digital conference. chinmay.siwach@imtlucca.it (C. Siwach); gabriele.costa@imtlucca.it (G. Costa); rocco.denicola@imtlucca.it (R. D. Nicola)
6. Database and Network Journal (2022). Over 30 Million New Malware Samples Found in 2022 as Cyber Threats Evolve. Database and Network Journal, vol. 52, no. 2, Apr. 2022, p. 25. Gale Academic OneFile, https://link.gale.com/apps/doc/A703171095/AONE?u=txshracd2548&sid=bookmark-AONE&xid=60203499
7. Ghouti, Lahouari, and Muhammad Imam (2020). "Malware Classification Using Compact Image Features and Multiclass Support Vector Machines." IET Information Security, vol. 14, no. 4, 2020, pp. 419–29, https://doi.org/10.1049/iet-ifs.2019.0189.
8. Jusoh, Rosmalissa, et al (2021). "Malware Detection Using Static Analysis in Android: a Review of FeCO (features, Classification, and Obfuscation)." PeerJ. Computer Science, vol. 7, 2021, p. e522–, https://doi.org/10.7717/peerj-cs.522.
9. Kim, Sangwon, et al (2022). "Sumav: Fully Automated Malware Labeling." ICT Express, vol. 8, no. 4, 2022, pp. 530–38, https://doi.org/10.1016/j.icte.2022.02.007.
10. Michael R. Smith, Armida J. Carbajal, Ramayaa, Nicholas T Johnson, Bridget I. Haus, Christopher C. Lamb, W. Philip Kegelmeyer, Joe B. Ingram, Eva Domschot, Stephen J. Verzi (2017). Mind the Gap: On Bridging the Semantic Gap between Machine Learning and Malware Analysis Murrugarra, David, et al (2016). "Estimating Propensity Parameters Using Google PageRank and Genetic Algorithms." Frontiers in Neuroscience, vol. 10, 2016, pp. 513–513, https://doi.org/10.3389/fnins.2016.00513.
12. PEiD (2023). https://www.aldeid.com/wiki/PEiD (Last accessed, February 2023)
13. Rizvi, Syed Khurram Jah, et al (Jan. 2022). "PROUD-MAL: Static Analysis-Based Progressive Framework for Deep Unsupervised Malware Classification of Windows Portable Executable." Complex & Intelligent Systems, vol. 8, no. 1, 2022, pp. 673–85, https://doi.org/10.1007/s40747-021-00560-1.
14. Silvia Sebastian, Juan Caballero (Oct. 2020). AVclass2: Massive Malware Tag Extraction from AV Labels. arXiv:2006.10615V2 [cs.CR] 28 oct 2020