Tracing the Ransomware Bloodline: Investigation and Detection of Drifting Virlock Variants

Salwa Razaulla*, Claude Fachkha*, Amjad Gawanmeh*, Christine Markarian*, Benjamin C. M. Fung†, Chadi Assi‡
*College of Engineering and IT, University of Dubai, Dubai, UAE
corresponding author: cfachkha@ud.ac.ae
†School of Information Studies, McGill University, Montreal, Canada
‡Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada

***Abstract*—Malware, especially ransomware, has dramatically increased in volume and sophistication in recent years. The growing complexity and destructive potential of ransomware demand effective countermeasures. Despite tremendous efforts by the security community to document these threats, reliance on manual analysis makes it challenging to discern unique malware variants from polymorphic variants. Moreover, the easy accessibility of source code of prominent ransomware families in public domains has led to the rise of numerous variants, complicating manual detection and hindering the identification of phylogenetic relationships. This paper introduces a novel approach that narrows the focus to analyze one such prominent ransomware family, Virlock. Using binary code similarity, we systematically reconstruct the lineage of Virlock, tracing its relationships, evolution, and variants. Employing this technique on a dataset of over 1000 Virlock samples submitted to VirusTotal and VirusShare, our analysis unveils intricate relationships within the Virlock ransomware family, offering valuable insights into the tangled relationships of this ransomware.**

***Index Terms*—Ransomware, Malware attribution, Virlock, Contrastive learning, Concept drift**

I. Introduction

The escalating threat of advanced ransomware poses serious challenges to cybersecurity society [1] which is dependable on cyberspace and its infrastructure, from the Internet of Things (IoT) to critical OT elements [2], [3]. Ransomware's evolving nature, marked by increasing disruption and profitability, underscores the urgency for innovative strategies in threat detection and mitigation. A notable characteristic of many malware instances lies in the substantial code similarities exhibited with prior variants, both within the same family and across different malware families [4]. This phenomenon, while not novel, becomes increasingly prevalent as malicious actors leverage code-recycling strategies, often borrowing extensively from publicly leaked sources [5]. The development of unique variations has been made easier with the advent of ransomware-as-a-service [6]. Over time, the efficacy of machine learning (ML) classifiers is greatly impacted by this trend of code repetition, as new malware profiles show more sophisticated and distinct traits, and old malware profiles become obsolete [7]. Using machine learning models with such dynamic input presents a significant problem called idea drift. When testing data gradually deviates from the initial training data, a phenomenon known as this occurs, which causes significant failures in deployed models [8]. The testing data distribution should roughly mirror the training data distribution for machine learning models to function. Nevertheless, the ever-changing malware landscape [9] presents significant problems to the models since it consistently deviates from the initial training data. In response to concept drift, the majority of learning-based models necessitate periodic re-training or fine-tuning [10]. However, this process is often resource-intensive, requiring the labeling of a substantial number of new samples, thus incurring significant expenses. Moreover, determining the optimal timing for model retraining poses a considerable challenge [11]. The vulnerability of the model to new attacks rises when retraining is deferred, as the outdated model may struggle to effectively adapt to evolving data distributions.

To address the effects of concept drift, this work introduces a novel approach focusing on effectively detecting drifting Virlock samples. The proposed method employs a self-supervised contrastive learning model to learn meaningful representations of binary assembly codes. This facilitates obtaining semantically rich descriptions of the input samples. By mapping training samples to a low-dimensional latent space through contrastive learning, the model increases the distances between samples of different classes while decreasing distances between samples within the same class. Such a process detects drifting samples effectively. To preserve contextual information from assembly instructions, the study employs a balanced instruction normalization method, providing effective comparison and distinction between different ransomware samples. The main contributions of this study are summarized as follows:

- We propose a novel method to address intra-family attribution challenges of Virlock using contrastive representation learning. This innovative approach enables accurate identification and classification of ransomware families based on their unique behaviors.
- We employ a robust instruction normalization method to retain semantic and contextual information from assembly instructions.
- We facilitate future research in the field by making our dataset of ransomware MD5 hash values and pertinent

graphs available to the research community. [1]

The remainder of this paper is organized as follows. In section II relevant related works are discussed. Section III outlines the problem, while Section IV presents the proposed framework. In Section V, we discuss our findings and results. In Section VI, we provide some discussions on some of the aspects of concept drift and highlight their relationship to our study. Finally, in Section VII we conclude and discuss future research directions.

II. Related Work

The issue of concept drift has recently garnered considerable attention, where research has been directed toward how machine learning performs in the presence of concept drift IoT malware, Cozzi et al. [12] investigated the lineage, relationship, and evolution of IoT malware families using binary code similarity. Addressing the challenge of concept drift, Yang et al. [13] introduced a system called CADE that detects drifting samples. Their proposed approach involves using contrastive learning to calculate the distance and dissimilarity between input samples. Dib et al. [14] presented EVOLIoT, which employs contrastive method to learn meaningful representations of IoT malware binaries. The authors have presented the tangled relationships between IoT malware and inter-family IoT malware classification. The authors in [15] introduced a method for tracing the lineage of 10 prominent malware families. Their approach involves unpacking and disassembling binaries to identify shared code within input samples, followed by a detailed analysis to effectively trace the lineage. Park et al. [16] utilized classification algorithms to categorize extracted behavioral feature sets into distinct malware families. Each cluster exhibits similar family features, and a cluster head is selected within each cluster, serving as a key element in the subsequent lineage inference process.

While previous studies have predominantly focused on various malware types, including those impacting IoT and Android, none have specifically delved into the issue of ransomware. To the best of our knowledge, our work is the first to investigate concept drift within ransomware families. By utilizing assembly instructions, strings, and API calls, we enhance the overall result of our research.

III. Problem Statement

The continuous emergence of new attacks and data daily presents a challenge for models to maintain fully updated and labeled datasets. Additionally, the laborious and costly process of labeling incoming samples demands specialized expertise, creating a bottleneck for security analysts who need to address various challenges such as accurate labeling, lineage tracing, and family attribution. A convenient approach to tackle these challenges is to model the binary code of an executable, providing an accurate representation of its behavior and evolutionary essence, particularly regarding reusing codes and adding functions. Evolving ransomware strains can be framed as a binary similarity comparison problem, where syntactic or semantic (dis)similar feature vectors of two samples are compared.

In this paper, we introduce a novel method that employs contrastive learning with an autoencoder model, such as BERT [17]. This method utilizes a contrastive learning-based autoencoder to train the model without explicit reliance on training data. To implement this, we leverage the evolution of ransomware samples as an augmentation strategy, encoding the same binary sample assembly instruction with different dropout rates to generate two distinct word embeddings, forming positive pairs [18]. Simultaneously, we maximize the distance between the word embeddings of other binary assembly codes, serving as negative pairs. This approach allows us to focus on individual samples, discerning those that significantly deviate from the original input data, facilitating the detection of drifting samples and their appropriate labeling. By identifying these "drifting" samples, we can label them appropriately and utilize them to re-train the machine learning model, improving its performance. The subsequent section delves into a detailed discussion of our proposed design framework.

IV. Design Methodology

We introduce a comprehensive framework, shown in Figure 1 designed for intra-class classification of Virlock ransomware, addressing the challenge of concept drift. Our approach starts with the extraction and normalization of assembly instructions. Subsequently, we employ a contrastive learning approach to learn a meaningful representation of the underlying meaning of these instructions. As shown in Figure 1, the detection module identifies the drifting samples and a lineage inference module derives the lineage graph of Virlock variants based on these drift and non-drifting samples. The details of our framework are described below.

For studying ransomware evolution, we leverage assembly code sequences as features. Assembly instructions are extracted from the sample dataset using the Ghidra disassembler tool [19]. Following assembly code extraction, we perform instruction normalization to convert instructions into tokens. We use a balanced instruction normalization method by incorporating rules so that the contextual and semantic information of these instructions are preserved., We then employ a multi-layer transformer encoder called Bidirectional Encoder Representations from Transformers (BERT), inspired by the work in [20], that learns vector representations for input words and sentences. Contrastive learning seeks to learn effective data representations by bringing semantically related samples closer (positive pairs) and pushing non-semantically related samples far apart (negative pairs). The contrastive learning objective function is defined as follows:

$$\mathcal{L} = -\log\left(\frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1\ [k\neq i]}^{2N}\exp(\text{sim}(z_i, z_k)/\tau)}\right) \quad (1)$$

[1] https://github.com/malXhunter/Virlock-lineage-inference

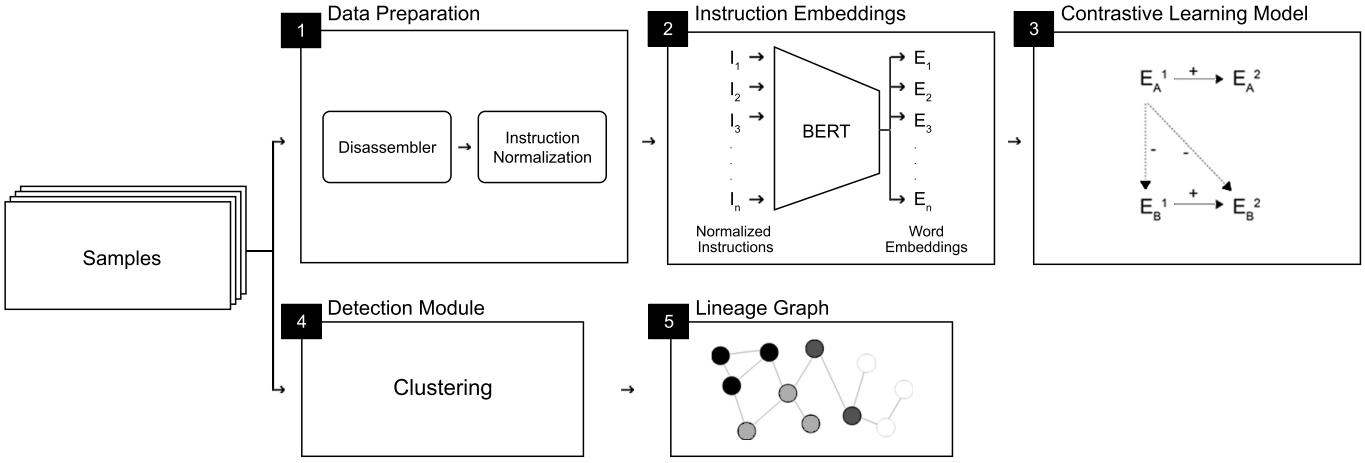

Fig. 1. Proposed workflow of our approach

where τ is a temperature parameter; $\text{sim}(z_i, z_j)$ and $\text{sim}(z_i, z_k)$ represents the similarity between the representations of two samples x_i, x_j and x_i, x_k respectively.

Our contrastive learning model transforms high-dimensional feature vectors into a lower-dimensional space, discriminating between positive and negative pairs based on their distances. The *t-SNE* plots in Figure 2 compare standard and contrastive representations of Virlock variants in our dataset, illustrating the effectiveness of our approach in capturing semantic similarity.

A. Detection Module

Our methodology involves training the model on specified training sets and then clustering the samples, prioritizing intra-cluster similarity over inter-cluster similarity. The detection module then carefully examines the testing dataset, marking samples that noticeably differ from what the model learned during training as "drifting samples." These peculiarities, showcasing patterns different from the training set, alert us to potential new or unusual instances that require closer inspection. After training the BERT model with a contrastive objective, we use its abilities to spot drifting samples. By representing samples in a lower dimensional space, we calculate cluster centroids and their standard deviations. Each sample of the testing dataset is placed in this space, and we measure the distances of each test point to the centroids of all clusters. For each test point, we calculate the minimum distance d_{min}, which denotes how close that sample is to the nearest centroid. Distinguishing between drifting and non-drifting data points is done by checking whether the distance d_{min} between a test point and its nearest centroid is less than the standard deviation of the closest cluster If it is, the point is assigned to the cluster; if not, it's identified as a drifting sample. This process helps us find samples that significantly deviate from what the model learned, aiding in the discovery of new or unusual instances during testing.

V. Evaluation Results

In this section, we examine how contrastive representation learning influences binary clustering and assess the performance of our detection module in pinpointing drifting ransomware samples.

A. Dataset

Our dataset comprises Virlock samples collected from malware repositories, including VirusTotal and VirusShare. The total Virlock samples that were collected for this study were 1061 In addition, we also collected around 1000 samples of each Gnadcrab and Lamer ransomware samples from these malware repositories. In dataset preparation, we conducted pre-processing steps to eliminate corrupt files and unpacked samples using the UPX packer tool [21]. Despite successfully unpacking a significant number of samples, some instances were unsuitable for this approach. For further analysis, we used the Ghidra disassembler tool to disassemble the remaining ransomware binaries. However, Ghidra encountered difficulties disassembling some samples, leading to their exclusion. Consequently, our final dataset comprises around 1000 Virlock samples for subsequent analyses.

B. Detection of drifting samples

In this section, we discuss the functionality of the proposed detection module. To evaluate the detection module, we consider two other ransomware families namely, Gandcrab and Lamer along with Virlock. In this way, we have three ransomware families to form a balanced dataset. One of these families is designated as the hidden family in each iteration, and the other families are split into training and testing sets. Consequently, during testing, the goal is to correctly identify samples that belong to the hidden family as drifting samples when the unseen family is unavailable for training. This iterative approach allows us to thoroughly evaluate the performance of our model across all ransomware families. Our data is divided into an 80:20 ratio of the training and testing datasets of any two families. Subsequently, we assigned the

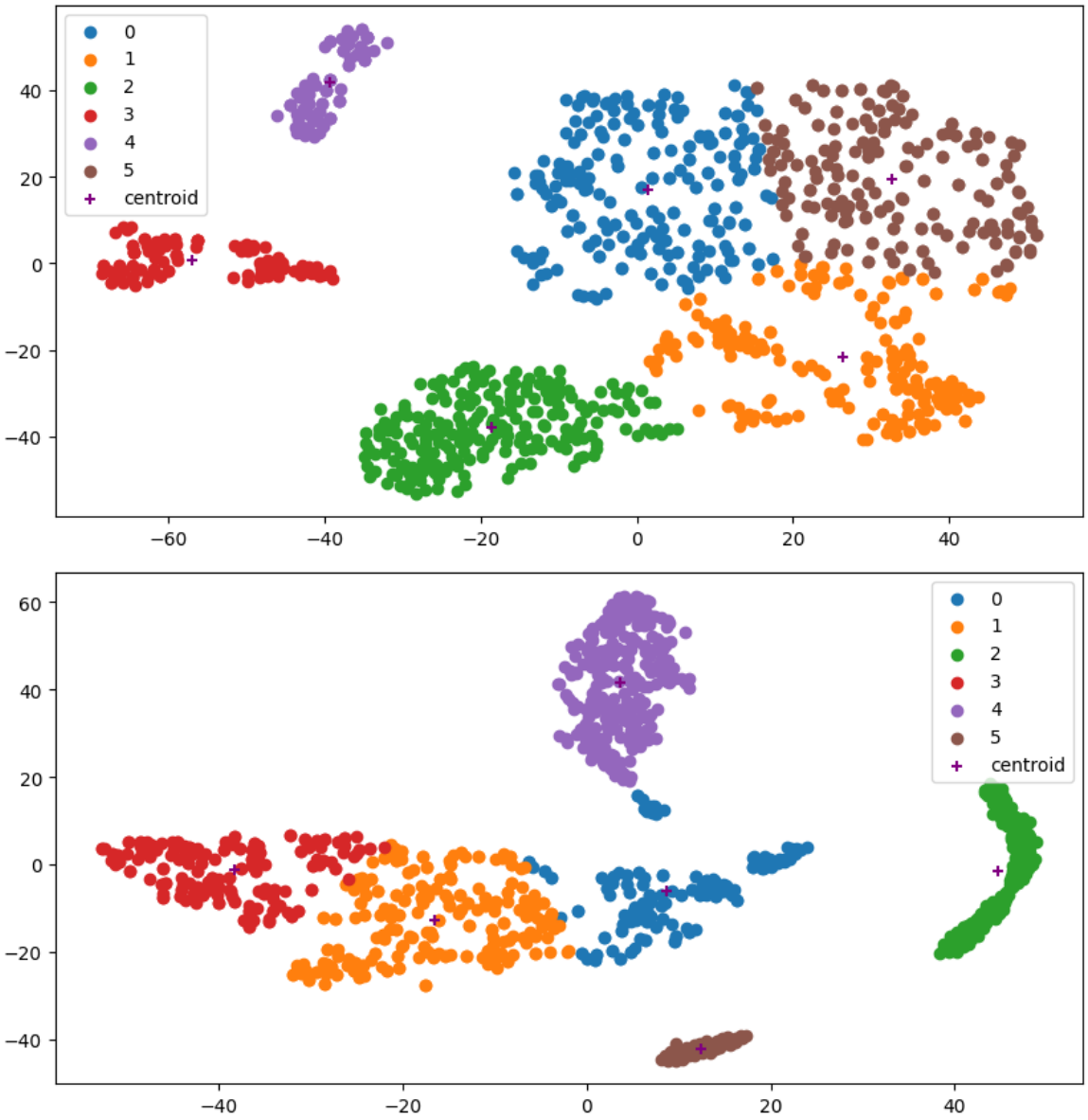

Fig. 2. t-SNE plots comparing Standard (top) and Contrastive (bottom) Representations of Virlock family (intra-class)

remaining family as the unseen category and incorporated it into the testing set to form the ultimate testing dataset. In our evaluation process, we treat the unseen family as negative samples, while the remaining testing samples serve as positive samples. Following the execution of our detection module on this dataset, we employed evaluation metrics to assess the performance of our approach for identifying drifting samples. Three evaluation metrics were used namely, accuracy, precision, and recall. Accuracy is a measure of the number of samples correctly identified as either drifting or non-drifting within the entire testing dataset. Precision, on the other hand, quantifies the ratio of correctly identified drifting samples to the total number of samples classified as drifting samples in our dataset. Finally, recall measures the ability of the model to correctly detect drifting samples out of all the samples that are genuinely drifting in the dataset.

Figure 3 shows the evaluation results of the proposed detection model. Overall, our model performed well in clustering Virlock samples with accuracy and precision exceeding 88%. While the recall value reached 100%. This shows that our model performed well in correctly detecting drifting samples out of all the samples that are genuinely drifting in the dataset. Our model was able to distinguish Virlock samples clearly, as shown in Figure 4, the clusters for Virlock were notably isolated from those of the other families. In contrast, the remaining families exhibited greater similarity and shared code patterns.

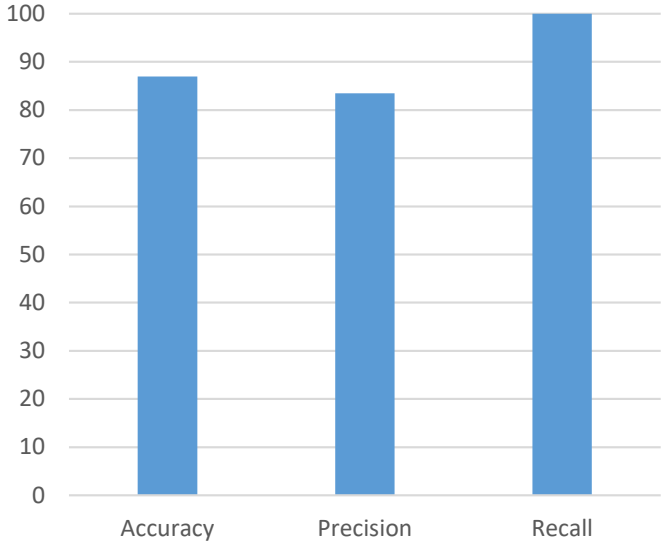

Fig. 3. Evaluation results - Accuracy vs Precision vs Recall

Fig. 4. UMAP graph showing the connection between different variants of the Virlock family

C. Lineage Inference

In Figure 4, the UMAP graph illustrates the interconnectedness of Virlock samples within our dataset. Darker lines signify high similarity, whereas faint lines indicate lower similarity. Notably, specific Virlock samples appear more connected, resulting in closer placement on the graph. In this section, we delve into understanding these connections among samples within each cluster that were derived by the detection module.

In our analysis of various Virlock samples, distinct patterns emerged in the sequences of their API calls. Common and essential API calls were observed across multiple Virlock variants, including *NtOpenFile*, *NTQueryInformationFile*, *NtProtectVirtualMemory*, and *NtTerminateProcess*, facilitating functions such as file manipulation, file information retrieval, memory protection modification, and process termination, respectively. Less frequently employed API calls were also identified, such as *FindResourceA*, *StartService*, *RegCreateKeyExW*, *CreateToolhelp32Snapshot*, and *IsDebuggerPresent*. Notably, samples in cluster #4, dating back to 2013 and submitted to VirusTotal in 2021, strategically utilized the Process Environment Block (PEB) to ascertain debugger presence, employing *IsDebuggerPresent* for debugger detection and *ExitProcess* to evade analysis [22]. Additionally, cluster #2 employed a technique involving *CreateToolhelp32Snapshot* to recognize known debugger names within the parent process [23]. These API calls were utilized to scrutinize system artifacts and determine the involvement of a debugger. Notably, cluster #2 variants succeeded cluster #4 variants and were observed in 2015.

VI. Discussions

A shift in the relationships between a model's input features and variables over time is known as the concept drift phenomenon. The latter has been well investigated in the literature. Several factors may cause concept drift, such as modifications in the environment, processes being modeled, and user behavior. We list below some of the aspects of concept drift and highlight their relationship to our study on ransomware malware.

Some researchers have developed models and tools to detect and monitor concept drift offline and live (in real-time). This includes determining whether the model's performance is declining as a result of shifting data distributions by using statistical tests, keeping an eye on performance measures, or using ensemble approaches. Because our model is based on contrastive representation learning, and our dataset is of a particular kind, our approach focuses on off-life inference. Even though we semi-automated the detection process, we still need to make some improvements to our model to enable the system to function in a real-time setting. When drift is detected, the detection model's parameters should be dynamically adjusted or updated on a regular basis. Adaptive learning [24] and incremental learning [25] algorithms are briefly discussed in this domain. The latter is more useful in real-world settings where ransomware variations are rapidly created.

Other researchers have studied concept drift thoroughly for evaluation and benchmarking and to understand precise drift dynamics. In order to evaluate the effectiveness of idea drift detection and adaptation algorithms, scientists proposed benchmark datasets and evaluation procedures. This involves creating realistic simulation frameworks or comparing the efficacy of various strategies using real-world datasets with established drift patterns. In addition, to understand the fundamental reasons and dynamics of concept drift in various

fields, scientists have also looked into drift dynamics. To find the causes of drift and forecast upcoming data changes, authors have also been required to evaluate past data, subject expertise, and contextual data. Last but not least, authors have examined imbalanced data and its effect on drift detection using a variety of datasets (such as ransomware API calls, network logs, source code, etc.). Authors have, for example, looked into ensemble-based [26] and resampling [27] techniques to mitigate the negative impact of drift on model effectiveness.

VII. Conclusion and Future Work

In this study, we addressed concept drift by employing a contrastive learning autoencoder model, effectively detecting Virlock samples that deviated from the original training set. Using an assembly instructions similarity-based approach, we attributed more than 1000 Virlock ransomware samples, leveraging extracted strings and API calls for lineage inference. Utilizing *t-SNE* plots, we illustrated that contrastive representation learning yields superior word representations compared to standard models. Our analysis revealed evidence of code reuse among these Virlock variants.

Our research has some limitations. We used a dataset with only 1061 samples, which might be considered small, affecting the strength of our findings. Additionally, a significant number of collected samples were packed, making it challenging to analyze them fully. This could compromise the accuracy and applicability of our results due to the limited dataset size. In future work, we plan to overcome these limitations by actively seeking more ransomware samples and including a wider variety of ransomware families. Additionally, we plan to extend our analysis to include inter-family ransomware attribution by considering other ransomware families. Furthermore, we will design a more robust instruction normalization method that can maintain semantic information of malware binary code while reducing the number of tokens used. This expansion is aimed at making our research more credible and robust.

References

[1] S. Razaulla, C. Fachkha, C. Markarian, A. Gawanmeh, W. Mansoor, B. C. M. Fung, and C. Assi, "The age of ransomware: A survey on the evolution, taxonomy, and research directions," *IEEE Access*, vol. 11, pp. 40 698–40 723, 2023.

[2] S. Amiroon and C. Fachkha, "Digital forensics and investigations of the internet of things: A short survey," in *2020 3rd International Conference on Signal Processing and Information Security (ICSPIS)*, 2020, pp. 1–4.

[3] K. Biron, W. Bazzaza, K. Yaqoob, A. Gawanmeh, and C. Fachkha, "A big data fusion to profile cps security threats against operational technology," in *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2020, pp. 397–402.

[4] J. Upchurch and X. Zhou, "First byte: Force-based clustering of filtered block n-grams to detect code reuse in malicious software," in *2013 8th International Conference on Malicious and Unwanted Software: "The Americas" (MALWARE)*, 2013, pp. 68–76.

[5] A. Calleja, J. Tapiador, and J. Caballero, "The malsource dataset: Quantifying complexity and code reuse in malware development," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 12, pp. 3175–3190, 2018.

[6] P. H. Meland, Y. F. F. Bayoumy, and G. Sindre, "The ransomware-as-a-service economy within the darknet," *Computers & Security*, vol. 92, p. 101762, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820300468

[7] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, mar 2014. [Online]. Available: https://doi.org/10.1145/2523813

[8] J. a. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, mar 2014. [Online]. Available: https://doi.org/10.1145/2523813

[9] E. Bou-Harb, C. Fachkha, M. Debbabi, and C. Assi, "Inferring internet-scale infections by correlating malware and probing activities," in *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 640–646.

[10] A. Liu, G. Zhang, K. Wang, and J. Lu, "Fast switch naïve bayes to avoid redundant update for concept drift learning," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–7.

[11] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2019.

[12] E. Cozzi, P.-A. Vervier, M. Dell'Amico, Y. Shen, L. Bilge, and D. Balzarotti, "The tangled genealogy of iot malware," in *Annual Computer Security Applications Conference*, 2020, pp. 1–16.

[13] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "{CADE}: Detecting and explaining concept drift samples for security applications," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2327–2344.

[14] M. Dib, S. Torabi, E. Bou-Harb, N. Bouguila, and C. Assi, "Evoliot: A self-supervised contrastive learning framework for detecting and characterizing evolving iot malware variants," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 452–466.

[15] I. Ul Haq, S. Chica, J. Caballero, and S. Jha, "Malware lineage in the wild," *Computers & Security*, vol. 78, pp. 347–363, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404818308605

[16] L. H. Park, J. Yu, H.-K. Kang, T. Lee, and T. Kwon, "Birds of a feature: Intrafamily clustering for version identification of packed malware," *IEEE Systems Journal*, vol. 14, no. 3, pp. 4545–4556, 2020.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[18] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[19] "Ghidra," https://github.com/NationalSecurityAgency/ghidra, March 2019, national Security Agency of the United States.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[21] L. M. . J. R. Markus Oberhumer, "UPX, The Ultimate Packer for eXecutables," https://https://github.com/upx/upx, 1996.

[22] K. Yoshizaki and T. Yamauchi, "Malware detection method focusing on anti-debugging functions," in *2014 Second International Symposium on Computing and Networking*, 2014.

[23] A. Afianian, S. Niksefat, B. Sadeghiyan, and D. Baptiste, "Malware dynamic analysis evasion techniques: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–28, 2019.

[24] S. Yang, S. Li, W. Chen, and Y. Liu, "A real-time and adaptive-learning malware detection method based on api-pair graph," *IEEE Access*, vol. 8, pp. 208 120–208 135, 2020.

[25] A. A. Darem, F. A. Ghaleb, A. A. Al-Hashmi, J. H. Abawajy, S. M. Alanazi, and A. Y. Al-Rezami, "An adaptive behavioral-based incremental batch learning malware variants detection model using concept drift detection and sequential deep learning," *IEEE Access*, vol. 9, pp. 97 180–97 196, 2021.

[26] M. Jain and G. Kaur, "Distributed anomaly detection using concept drift detection based hybrid ensemble techniques in streamed network data," *Cluster Computing*, vol. 24, pp. 2099–2114, 2021.

[27] A. Liu, J. Lu, and G. Zhang, "Concept drift detection via equal intensity k-means space partitioning," *IEEE transactions on cybernetics*, vol. 51, no. 6, pp. 3198–3211, 2020.