Intelligent Malware Defenses: A Survey

Azqa Nadeem[1](✉), Vera Rimmer[2], Wouter Joosen[2], and Sicco Verwer[1]

[1] Delft University of Technology, Delft, The Netherlands
{azqa.nadeem,s.e.verwer}@tudelft.nl
[2] imec-DistriNet, KU Leuven, Leuven, Belgium
{vera.rimmer,wouter.joosen}@cs.kuleuven.be

Abstract

With rapidly evolving threat landscape surrounding malware, intelligent defenses based on machine learning are paramount. In this chapter, we review the literature proposed in the past decade and identify the state-of-the-art in various related research directions — malware detection, malware analysis, adversarial malware, and malware author attribution. We discuss challenges that emerge when machine learning is applied to malware. We also identify the key issues that need to be addressed by the research community in order to further deepen and systematize research in the malware domain.

1 Introduction

Over the past two decades, malicious software (malware) has emerged as one of the biggest security threats. AV-test, a security research institute, has reported detecting more than 1000 Million malware samples in 2019[3]. According to Accenture, a malware attack on a company can cost $2.4M on average and can take 50 days to resolve[4]. Anti-Viruses (AVs) are considered to be the first line of defense. However, according to a survey by Ponemon Institute, 69% organizations do not believe that AVs can block the threats that they monitor. Given these staggering numbers, classical rule-based malware detectors can simply not be expected to detect the large influx of malware variants. The main problem with rule-based defenses is that they are reactive, where a rule is added only *after* experiencing an attack.

Machine Learning (ML) has become a promising ally for malware detection. The security community has been investigating ways to incorporate machine learning for intelligent malware detection, profiling, and analysis. Figure 1 shows a typical pipeline for malware defense and the opportunities to introduce machine learning in it. It is noteworthy that machine learning is also useful for attackers: Due to the intrinsic adversarial nature of the threat landscape, machine learning has not only been used to build intelligent defenses, but it has also been used to develop intelligent attacks that evade detection. In the past decade alone, this arms-race has resulted in more than 20,000 research articles.

[3] https://www.av-test.org/en/statistics/malware/
[4] https://www.accenture.com/us-en/insight-cost-of-cybercrime-2017

In this chapter, we conduct a systematic survey of the literature published in the past decade to establish a taxonomy of the main research themes. For an unbiased literature review, we select peer-reviewed papers containing a combination of fixed search queries that are highly cited in their domain. We summarize the state-of-the-art in various sub-fields of intelligent malware defenses, *i.e.,* malware detection, malware analysis, adversarial malware, and malware author attribution. The literature is greatly dominated by *malware detection* approaches with the aim of developing scalable behavioral signatures. Approaches from other domains have been applied to perform malware detection, such as natural language processing, image visualization, graph mining, and bioinformatics. We categorize the research in this domain according to the data source and feature representation used for their classifiers. *Malware analysis* is another research direction that develops tools that provide the necessary insights to improve malware detection. We discuss approaches that aim to increase interpretability, and provide smarter ways to collect behavioral traces. *Adversarial machine learning* has recently gained popularity, not only for machine learning-based offensive security, but also for hardening machine learning classifiers. Finally, *malware author attribution* aims to associate malware to its author(s), a field that is mainly driven by law enforcement agencies. Although not a very active area, it serves as a powerful use-case for interdisciplinary research. Figure 2 shows the literature overview in a chronological order, divided across the aforementioned research directions.

We discuss important considerations that emerge when machine learning is applied to malware, such as resilience against concept drift and evasion, handling imbalanced datasets, using appropriate evaluation metrics, and providing privacy and performance guarantees. We have observed that the absence of toy problems, representative datasets, explainable approaches, and the usage of noisy ground truth has limited the reproducibility of available research. Specifically, explainable approaches are necessary for debugging existing techniques and developing newer ones based on obtained insights. These issues need to be addressed by the research community in order to encourage systematized research in the intelligent malware defenses domain.

This chapter is organized as follows. Section 2 serves as a roadmap for the rest of the chapter: it identifies the feature sources and representations that have been used to characterize malware in the literature, including several feature engineering modes. Section 3 discusses the vast literature that explores effective and efficient malware detection methods. We expand the discussion on malware research in Section 4 by covering relevant areas, *i.e.,* malware analysis, adversarial malware, and author attribution. Section 5 enumerates the main challenges unique to machine learning-based malware defenses. Section 6 highlights the four key issues that should be addressed to enable reproducible research in the intelligent malware defenses domain. Finally, we conclude our discussion in Section 7.

2 Malware characterization

The success of machine learning classifiers lies in finding the data that appropriately characterizes malware. Determining these data and the input features required for machine learning is a difficult task since they will be used to detect new malware samples

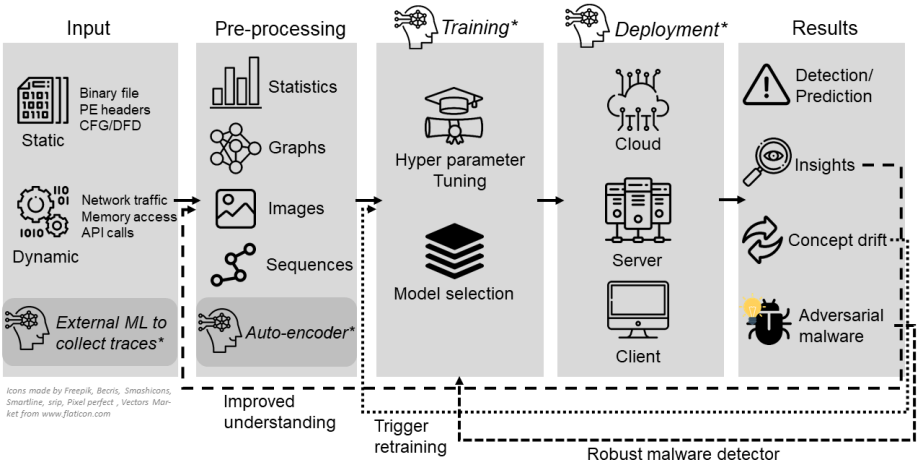

Fig. 1: Machine learning pipeline for malware defense: Raw input data gets transformed into features during pre-processing, which are given as input for model training. The raw data may be collected using ML-based sandboxing. Features may be automatically learned using auto-encoders. Depending on the problem, the model may be deployed at different locations, and the output may be used in several ways: Malware detection typically raises alerts; Insights obtained from malware analysis can lead to improved malware characterization; Detecting changes in data distribution can trigger model retraining; Incorporation of adversarial malware in the training process can lead to robust models.

that may behave in unexpected ways. Anderson *et al.* [1] suggest that effective feature engineering, including features obtained from domain experts, plays a key role in classifier performance. There exists a myriad of literature exploring the various features that can be used to characterize different types of malware [2,3]. In this chapter, we provide an overview of the past ten years of intelligent malware defenses from a technical machine learning perspective, grounded in the types of features used to characterize malware: statistical, graphs, images, and sequences. The type of input feature greatly influences which machine learning technology can be employed. Other important considerations are: (a) the target platform of the malware, (b) how to collect data from a malware, and (c) how to extract features from such data. We briefly introduce these other considerations below but will not go into detail.

2.1 Platform-specific malware and defenses

Malware often targets consumer devices, like desktop computers and handheld devices. The first malware was a PC-based virus, called Elk Cloner[5], discovered over 35 years ago. Since then, malware has targeted multiple operating system and browser vulnerabilities

[5] https://en.wikipedia.org/wiki/Elk_Cloner

to infect desktop computers. It is important to keep in mind that most research in intelligent malware defense also targets specific platforms, and this has a great effect on the used input features. Recent ransomware [4] and cryptojacking malware [5] are well-known for attacking desktops. A major strain in research therefore targets desktop malware, most frequently Windows-based malware. Recently, Ucci *et al.* [2] presented a survey of features characterizing Windows Portable Executable (PE) malware.

With the widespread use of Android smartphones, there is a growing amount of literature on Android malware detection [6,7,8,9,10,11,12,13,14,15], mainly for two reasons: Firstly, Android is an open-source operating system, so developers can investigate various vulnerabilities that malware has exploited over the years; Secondly, multiple large-scale and open-source datasets of malicious Android applications have supported research in this field. This does not mean that iOS-based malware does not exist [16], it is just not a frequent subject of research.

Nowadays consumer devices are increasingly becoming equipped with Internet connectivity, known as the *Internet of Things (IoT)*. This comes with new risks culminating in a novel strain of malware specifically targeting IoT devices [17]. IoT devices are commonly made available on the Internet with their default configurations, which makes them an easy target for the attackers. Since IoT devices have limited computational resources, their built-in security is significantly inferior to other internet-connected devices, making it an ideal use-case for intelligent malware defenses. Several recent works have proposed to enhance IoT security using ML-based malware detection [18,19]. In this chapter, we try to avoid this distinction between platforms, instead focusing on technological differences. However, often the platform and technologies are tightly linked, and solutions are typically not directly applicable to other platforms.

2.2 Feature sources

There are two major approaches for analyzing malware: (i) static analysis, and (ii) dynamic analysis. Machine learning has been applied successfully in both approaches. For static analysis, *static features* are extracted from a malware's code, *i.e.,* without executing it. For dynamic analysis, features are extracted by running malware and monitoring its behavior. These features can be obtained from two sources: standard *dynamic features* are generated on the host device, typically by interacting with the operating system, while *network features* are created from network traffic generated by hosts in a network.

Static features. The source code of a malware, often obtained by decompiling its binary, is the most reliable artefact to identify its objective. Early studies on malware detection have mainly characterized malware using features extracted from its code. These features are often obtained by doing a comparative analysis of goodware (benign software) and malware features, and selecting the ones that are observed more frequently in malware. PE headers are commonly used for Windows malware [20,21,22,23,24], while some works extract features directly from the malware binary itself [25,26,27]. Most of the recent literature from static analysis comes from the smartphone domain (*i.e.,* Android), where the features are extracted from either the APK's manifest file or

the disassembled Dalvik bytecode for signature generation. Shabtai *et al.* [28] is one of the earlier works on Android malware detection that uses features from the Dalvik bytecode in order to perform binary classification (*i.e.,* malware vs. goodware). One of their major contributions is to apply a myriad of classification algorithms and select the one that achieves the highest accuracy. Other features, such as Operational Codes (OpCodes) [10,19] and function API calls [8,11,29,30,31], are also commonly used for malware detection. Existing works also analyze the amount and types of permissions that applications request to measure their maliciousness [7,8,12,32]. For a higher-level semantic analysis, features from Control flow graphs [7,33] and Data flow graphs [34] have also been used.

Dynamic features. With the widespread use of code obfuscation tools to evade detection and to generate malware variants, syntactic analysis has become increasingly more difficult. Additionally, there has been a spike in fileless malware infections[6], where the malware code resides purely in the victim host's memory without leaving any code fingerprints. Hence, dynamic analysis is more popular for malware detection. In dynamic analysis, malware is executed in a controlled environment and its behavior is monitored [35]. Information such as, system API calls [36,37,38,39], memory access patterns [40,41], and running processes [42] are common sources for feature selection. Some works consider inter-file relationships between files present on a system for malware detection [43,44]. There also exist hybrid approaches, where static analysis guides dynamic analysis for thorough code coverage [45].

Network features. Network traffic analysis is popular because it can be performed remotely and presents lower overhead than its system-activity counterpart. However, machine learning has been slow to materialize in the network security domain because of noisy ground truth and non-stationary data distribution [1]. Existing sandboxes also have limited support for handling network requests due to the risk of lateral movement, *i.e.,* when the attacker who has gained access to the network spreads their reach to other hosts [46]. Nevertheless, the use of HTTP header fields [14,47,48] and traffic connections [6] is common for malicious traffic detection. Privacy concerns have also been addressed in network security. Boukhtouta *et al.* [49] evaluate the differences between Deep Packet Inspection (DPI) based methods and IP-header based methods for classifying malicious network traffic. They conclude that IP-header features make the machine learning model generalizable and can achieve higher accuracy due to the independence from packet payloads. They also suggest that using IP-header features can help fingerprint zero-day malware, *i.e.,* malware never seen before.

2.3 Feature engineering modes

Most of the existing works perform manual feature engineering, where the features are initially obtained from domain experts and are further cherry-picked based on a classifier's accuracy [47,29,8,31,24,20,21,25,19,10,30,11]. The downside to manually

[6] https://www.cybereason.com/blog/fileless-malware

engineering features is that it is a laborious and (potentially) subjective process, which may need to be repeated in case of concept drift, for example. The malware threats also need to be fully scoped-out before arriving at the optimal feature-set. Nevertheless, knowing the features beforehand helps with explaining and debugging a classifier's decisions.

In the recent years, deep neural networks have gained tremendous traction, making automated feature engineering through representation learning (*e.g.,* with auto-encoders) a popular choice. Pascanu *et al.* [50] propose a recurrent neural network approach, specifically Echo State Networks, for automatically extracting time-domain features. They use these features in a Logistic Regression (LR) model for malware classification, and achieve better performance than a trigram based manual feature engineering approach. David *et al.* [51] propose a system that uses Deep Belief Networks (DBN), which are a type of generative graphical deep neural network that can perform unsupervised learning, with a deep stack of de-noising auto-encoders to automatically generate behavioral signatures. Yuxin *et al.* [33] have also used DBNs as auto-encoders to automatically extract features from malware executables.

A common critique faced by features that are automatically learned from deep neural networks is that they are uninterpretable, and hence undesirable for building explainable solutions. Building interpretable deep learning models is an open area of research. Zhu *et al.* [9] have recently proposed an interesting approach that automatically engineers features by mimicking human analysts' feature engineering processes. Their system mines academic documents and synthesizes their knowledge into interpretable features that are later used for Android malware classification. They report comparable results to state-of-the-art manual feature engineering approaches.

2.4 Feature representation

After having selected the data source and features to use, the next step is to determine how to process these data. We identify four different kinds of feature representations in this chapter, *i.e.,* statistical, graphs, images, and sequences. Note that feature representation means the format of the input given to a ML classifier, not the intermediate representations from representation learning.

Represent as statistics. The most common representation for features in the literature is statistical [1,7,8,12,13,14,20,22,29,30,31,32,39,42,47,52,53,54,55,56,57]. Feature values are collapsed using aggregates or correlations. Some statistical features also capture the temporal aspect of the behavior without having to deal with sequential data, *e.g.,* the Power Spectral Density of the FFT of the Command & Control communication [58].

Represent as graphs. One branch of literature represents malware using graphs, *e.g.,* to represent the hosts a malware connects with, as an abstraction over the original feature-set to get a high-level view of malware's behavior. Graphs are used to either ease analysis [59] or to extract semantic features for malware detection [43,44,60,61]. Graphs are also used to perform malware causality analysis [62,63,64].

Represent as images. Another branch of literature explores various visualization methods in order to characterize malware. A malware binary can be converted into an image by encoding the raw values of the binary as the color intensity of pixels [27,65,66,67,68,69]. The intuition here is that a malware family may share similar code pieces, which will appear as similar motifs in the image. Visualizing malware, and hence exposing these similarities to the human eye, can potentially aid manual malware analysis. Furthermore, research has also applied standard image classification techniques to perform efficient and reasonably accurate malware detection. Despite these encouraging initial results, we note that image representation for malware classification should be considered with caution, as discussed in more detail in Section 3.3.

Represent as sequences. There is an increasing amount of literature that is investigating the use of sequential data in behavior characterization. Although sequential features capture the temporal aspect of behavior, machine learning algorithms with inherent support for sequences are rare. The main difficulty lies in appropriately measuring the distance between two sequences in the presence of noise, delays and misalignments. Methods from other fields, such as *sequence alignment* adopted from bioinformatics [70,71], and *n-grams* adopted from natural language processing [19,23,72,73,74] have been utilized to that end. Increasingly more approaches are using deep neural networks because they have good support for sequences, *e.g.,* Long Short Term Memory networks (LSTM) [37,38], Recurrent Neural Networks (RNN) [21,75,76] and Word2Vec [77,11,78].

3 Malware Detection

A central objective of malware research is to develop behavioral signatures that can automatically detect future malware variants. We make a distinction between two major strains of research in the malware domain: *Detection-based* (this section) and *Analysis-based* (Section 4.1).

Most of the existing literature is about malware detection and signature generation, with the end-goal of optimizing metrics, such as classifier accuracy and F1-scores [58,56,57,79,47,52,80]. To this end, a plethora of research has been conducted over the past ten years that explores various features, representations, and machine learning algorithms. LeDoux *et al.* [81] summarize the research on malware detection, particularly on code-reuse detection, using machine learning. They also enumerate malware analysis problems that machine learning *is* equipped to solve. The vast literature on feature extraction and data mining techniques for malware detection is comprehensively described by Ye *et al.* [3]. Souri *et al.* [82] evaluate various signature- and behavior-based intelligent malware detectors.

We survey malware detection methods by focusing on the feature representations used by the applied machine learning or data mining technology. Both supervised and unsupervised machine learning techniques have been used in the literature. We categorize them into three classes: (i) *Binary classification*: Determine whether an unlabeled software is goodware or malware, (ii) *Multi-class classification*: Given a set of unlabeled malware samples and a set of known malware family names, perform malware family attribution, and (iii) *Clustering*: Given a set of unlabelled software,

Fig. 2: Overview of ML-based malware defenses proposed over the past decade.

Table 1: Literature according to feature source and representation.

Source→ Representation↓	Static	Dynamic	Network
Statistical	[7,8,12,13,20] [29,30,31,32,52,53,54]	[36,39,42,52,55]	[1,14,47,52,56,57,58]
Graphs	[7,60,61]	[43,44,63,64]	[62]
Images	[26,27,65,66,67,68,69]	-	[83]
Sequences	[9,10,11,18,19,21,23,24,25] [33,50,51,54,70,73,77]	[37,38,71]	[79,80]

categorize them into distinct classes based on structural/behavioral differences. Table 1 summarizes the malware detectors reviewed in this chapter, categorized according to the feature source (*i.e.,* static, dynamic, network) and input representation (*i.e.,* statistical, graph, image, sequence) that they employ.

3.1 Statistical approaches

An aggregated feature-set is the most widespread feature representation used in literature. Statistical features are fast to compute and simple to incorporate in a machine learning classifier.

Experiments have shown that the prevalence of certain feature values is a decent indicator of malware. Naturally, binary classification proves to be more successful under this setting than multi-class classification. Hence, even if the experimental dataset is composed of malware from multiple families, earlier works considered them together as one 'malicious' class. Alazab *et al.* [30] perform binary classification for the task of 'zero-day' malware detection. They use the frequency of static API calls to characterize Windows binaries, and show that it is indeed possible to distinguish between malware and goodware using this characterization. Aafer *et al.* [31] propose a light-weight Android malware detector based on the frequency of static API calls. They use K-Nearest Neighbors (KNN) to detect and alert the user of malicious applications. Sahs *et al.* [7] characterize Android applications using a binary vector of used permissions. Because there exist significantly more benign applications than malicious ones, they utilize a one-class Support Vector Machine (SVM) which characterizes benign applications well, and helps detect Android malware. Yerima *et al.* [8] use Bayesian classification to detect malware. They characterize applications using API call- and permission- frequencies.

Santos *et al.* [53] utilize the frequency of OpCode occurrence in executables to detect malware. They show that Polynomial Kernel classifiers and Random Forests achieve the best performance, which is not surprising because these algorithms have a long history of performing well in text classification. Suarez *et al.* [13] extract statistical features from Control Flow Graph (CFG) code blocks, such as the number and redundancy of code chunks, and common and discriminant code chunks. These text-based features are used for multi-class classification. Earlier deep learning approaches have also characterized malware using aggregates. For example, Saxe *et al.* [20] propose a malware detection system that uses a feature-set of byte histograms and frequency of PE import calls.

Although statistical features have been applied successfully in a supervised manner, there are assumptions that may be difficult to realize in practice. For example, AV-assigned family labels are noisy and novel threats are common. In contrast, unsupervised learning can be used independently to identify novel threats. In the binary classification setting, an anomaly detection approach is often used to model the benign class and anomalies are labelled as malicious, giving the capability to detect novel threats. For example, Burguera *et al.* [36] built a malware detection system using K-Means clustering to identify anomalous system events by finding deviations from one 'normal' cluster. For the multi-class setting, clustering is used to identify different threat classes. Perdisci *et al.* [47] present one of the first unsupervised clustering approaches to detect HTTP-based malware. They propose multi-step clustering to enable large-scale malware behavioral signature generation. Unsupervised machine learning is also often used in combination with supervised approaches to improve detection capabilities. Rieck *et al.* [55] propose an incremental analysis approach for malware family identification: by first performing clustering to identify *novel malware classes*, and then classifying unknown malware samples by assigning them to these discovered classes. Burnap *et al.* [42] have recently developed an unsupervised learning method based on Self Organizing Feature Maps (SOMs) that cluster similar malware behavior. They use the clusters of similar behavior as features for later classification tasks. The key benefit of this approach is an added layer of abstraction for improved classification — instead of using raw features for classification, the system allows fuzzy boundaries that can map new samples onto the existing decision boundary. David *et al.* [51] use Deep Belief Networks, a type of unsupervised model, to automatically generate malware behavioral signatures. Finally, Li *et al.* [14] build a network traffic-based malware classifier that utilizes both supervised and unsupervised classifiers to improve classification accuracy.

Effective malware detectors provide stable and trustworthy results. Since statistical features provide an aggregated view, a malicious application may appear similar to a benign application from this point of view. Hence, the choice of feature representation plays a crucial role in a classifier's robustness. Recently, Milosevic *et al.* [54] compared two text-mining approaches for Android malware detection — using statistical features (*i.e.,* permissions) and sequential features (*i.e.,* bag-of-words of decompiled `Dex` code). Their experiments show that the bag-of-words approach performs better due to better malware characterization, indicating that statistical features may not be the optimal choice in all cases. Another way to improve the robustness of malware detectors is to use *Ensemble learning*, *i.e.,* a learning paradigm that combines the decisions of multiple classifiers to arrive at the final decision. Ensemble models, such as Random Forests have been shown to be robust to non-stationary data distribution, such as network traffic [1]. Recently, Zhu *et al.* [12] proposed an ensemble Rotation Forests model to classify Android malware. Rotation Forests [84] are an ensemble of Decision Trees where diversity through rotated principal components is given emphasis, resulting in more stable decisions.

3.2 Graph-mining approaches

Graph-mining approaches have been used to represent malware's relationships in a graphical format, in order to provide an added abstraction layer. Malware literature adopts scalable graph mining approaches to perform fast detection.

Security products often have to mark files as malicious or benign based on a partial view of the host's file system. Approaches using inter-file relationships have emerged as a solution. Chau *et al.* [43] perform malware detection using large-scale graph inference. They consider files as malware based on *guilt-by-association* — they exploit the inter-file relationships present on multiple systems to compute reputation scores for unlabelled files. They use Belief Propagation algorithm to mark files with low reputation as malware. They evaluate their approach on a 60 terabyte dataset composed of a Billion-node graph, and show significant improvement over existing approaches. Acar *et al.* [44] also use a similar approach to detect malware, with the additional use of Locality Sensitive Hashing (LSH) for efficiently binning similar files.

Hou *et al.* [60] build the first approach that uses a Structured Heterogeneous Information Network (HIN) to characterize API-relatedness. The HIN edges are used to measure the semantic similarity between API calls, that is used to measure maliciousness of an application using multi-kernel learning. Fan *et al.* [61] improve upon the previous approach using a *meta-graph* to determine inter-file relationships for malware detection. For cost-effectiveness, they use *MetaGraph2Vec* to learn low-dimensional representations for the HIN that preserves its structure and semantics. *MetaGraph2Vec* is a meta-graph approach that has shown competitive performance for heterogeneous graph-mining tasks such as node classification and clustering.

Using graphs to perform causality analysis is also an area of interest. Zhang *et al.* [62] detect malware by performing *causality analysis* on its network traffic. They build Triggering Relation Graphs (TRG) that show the inter-dependency of various network events. The TRGs show an absence of dependency between legitimate and malicious network events, hence making it easier to detect malicious activities. Liu *et al.* [63] build a backward- and forward-causality graph to detect abnormal system events, based on their rareness and location in the causality graph.

3.3 Image visualization approaches

Malware visualization has opened a new research direction that uses ML-based image classification to detect malware. These approaches rely on converting a malware binary into an image which is then provided to an image classifier, either as a raw image or as a set of extracted features. The key assumptions for these methods are: (a) *malware families have similar images because of code reuse*, and (b) *malware images are significantly different from goodware images*.

In 2011, Nataraj *et al.* [65] proposed a straightforward method to convert malware into an image: a malware executable represented as a binary vector is reshaped into a matrix of an arbitrary width and is viewed as a grey-scale image. The authors observed that malware binaries belonging to the same family were visually similar in both layout and texture. They extracted textural features from these images and applied K-Nearest-Neighbors to perform malware family identification. This approach proved to be very

efficient and achieved high accuracy, close to prior methods that used static features such as n-grams. Furthermore, they showed that malware belonging to one family packed with the same packer, or containing sections encrypted with polymorphic engines, are still categorized together as the same family, indicating some level of resilience to *naïve* obfuscation. Motivated with the initial positive results, several follow-up studies [27,66] expanded the research by investigating different types of image feature extractors and machine learning classifiers. Approaches to malware classification based on image similarity were confirmed to be effective on the commonly used Kaggle Microsoft Malware dataset [85].

Recent work has started applying deep neural networks for the classification task, inspired by their encouraging performance in the image classification domain. Kalash *et al.* [67] successfully perform malware family identification with a two-dimensional Convolutional Neural Network (CNN) architecture, and Singh *et al.* [69] convert malware binaries into colored-images to classify obfuscated malware with a `ResNet-50` architecture, *i.e.,* a CNN architecture with shortcut connections providing superior performance. Yakura *et al.* [68] applied a CNN with attention mechanism [86] that allows to explain which areas of an image contribute to particular classification decisions. Le *et al.*[26] have recently proposed a fully automated malware classification approach for non-domain experts. They represent raw binary files as grey-scale images using the approach by Nataraj *et al.* [65]. The images are given to a hybrid network, *i.e.,* CNN with Bi-directional LSTM model, which outperforms a traditional CNN model.

Despite promising performance on benchmark datasets, representing binaries as images for malware detection comes with several limitations. Apart from some empirical results, there appears to be little evidence to strongly support the aforementioned underlying assumptions of malware classification based on visual similarity of binaries. Some studies have encountered images from different malware families that exhibited such similar patterns that they were classified in one class [66]. Naturally, only previously seen malware can be identified, while zero-day malware which is structurally different will likely evade detection [27]. Even for known malware, a common case of false negatives is observed when malicious content of relatively small size is embedded within a goodware, such that the resulting image remains very similar to other benign examples. Additionally, when global image features are used, merely relocating sections in a binary or adding large sections with redundant data may be sufficient to alter the image texture and mislead a classifier. These issues illustrate that generalizability of this approach, beyond particular datasets, remains an open question.

While image representations allow large-scale malware classification in a computationally feasible way, deep neural networks do not necessarily require this intermediate transformation. One-dimensional CNN architectures and sequential deep neural networks are directly applicable to raw one-dimensional binaries. Meanwhile, reshaping an initially one-dimensional binary sequence to a two-dimensional image with an arbitrarily chosen width introduces artificial spatial relations that are not present in the original file. Additionally, this representation does not appear stable, as adding or removing binary data in one location may completely change the positional relations in the vertical direction of the converted image [68]. Such artefacts may obscure naturally occurring patterns in data and negatively impact classification. This line of reasoning may also

apply to a few studies that attempt to use CNN on dynamic features, such as malicious network traffic: Wang *et al.* [83] perform malware detection by representing network traffic as images. In any case, it is very challenging to make such two-dimensional CNNs capable of recognizing the temporal dependencies required for processing traffic data.

3.4 Sequence learning approaches

Sequential pattern mining has emerged as a promising approach for malware detection due to increasingly better-performing sequential ML classifiers.

Earlier sequence learning works utilize n-grams to characterize temporal behavior. Jain *et al.* [23] represent PE files as n-grams (with varying values for n) to perform binary classification. They select the prominent n-grams using Class-wise Document Frequency method. Their experiments show that trigrams with Random Forests give the best malware detection performance. Similarly, Canfora *et al.* [73] use n-grams generated from OpCode sequences to detect malware families. They show that bigrams with Random Forests give the best performance. We believe these differences exist due to different datasets and feature selection approaches. Fan *et al.* [24] uses function call sequences and an All-Nearest Neighbors (ANN) classifier for PE-malware detection. ANN is a modification of K-Nearest Neighbors algorithm that makes a decision based on *all* neighbors. ANN makes a compromise on run-time efficiency for the sake of robustness.

In recent works, deep learning-based approaches have dominated malware detection using sequence learning. Convolutional Neural Networks (CNN) are used due to their ability to detect complex and non-linear patterns in data. Raff *et al.* [25] build a CNN framework that takes an entire PE binary as input for automated feature engineering and malware detection. Their method consumes the entire executable as opposed to only the PE-header to avoid over-fitting on the header features. However, their results do not show significant improvement over their baseline — a byte n-gram model. Mclaughlin *et al.* [10] perform Android malware detection using a CNN framework that utilizes raw OpCode sequences. Azmoodeh *et al.* [18] also use OpCode sequences to perform binary classification on IoT malware that is specifically used for military purposes. They reduce the feature dimensions using Principle Component Analysis (PCA) by providing only the first two components to the classifier. Recently, Cakir *et al.* [77] have used Word2Vec feature embedding on OpCode sequences to characterize malware. They use Gradient Boosting, which is a type of ensemble learner, for binary classification of malware. Karbab *et al.* [11] use CNN and Word2Vec feature embedding on API call sequences for malware family identification. They evaluate their system on multiple datasets, such as the MalGenome dataset [87], the Drebin dataset [88], and benign applications from Google Play[7]. Kolosnjaji *et al.* [21] perform multi-class classification using hybrid deep learning, *i.e.,* Convolutional and Recurrent Neural Networks (CNN/RNN), for Windows malware family identification.

Recently, Haddadpajouh *et al.* [19] have explored various configurations of Long Short Term Memory networks (LSTMs) for IoT malware classification. They characterize binaries by their OpCode sequences, and then choose the features that maximize

[7] https://play.google.com/store/apps

the Information Gain (IG). However, it is unclear how generalizable their results are as they only use ~500 binaries for the classification task. Zhang *et al.* [37] use so-called behavior chains based on API call sequences to characterize malware's behavior. They use LSTMs to perform binary classification and report a false positive rate of less than 2% (in the best case). Mishra *et al.* [38] use deep learning and the sequence of dynamic system calls for malware classification in the cloud environment. Their system uses two layered approach – CNN for feature engineering and Bi-directional LSTMs for malware detection. They evaluate their system on a university's network traffic and show promising results.

Sequential features are a common occurrence in bioinformatics. As computer viruses attain their characterization from similarity to natural viruses, bioinformatics-inspired solutions have also been proposed to detect malware variants. A common approach to detect similar DNA sequences is through the use of sequence alignment algorithms, such as the Smith-Waterman algorithm. Sequence alignment methods work by assigning a score based on matches, mismatches and gaps. These values are embedded in a substitution matrix. Domain-specific substitution matrices exist for bioinformatics applications. Chen *et al.* [70] map malware's binary code to Amino acid characters, and use the so-called Residue substitution matrix, while Naidu *et al.* [89] report that the `PAM-350` substitution matrix performs the best for malware variant detection. Chen *et al.* [70] also develop a multiple sequence alignment method that uses neural networks to classify viruses and worms. They show that alignment-based methods allow classifiers to find similarities with more ease compared to other methods.

3.5 Performance Optimizations

Malware detectors need to be efficient to cope with the exponential increase in malware attacks. Hence, some works propose extensions to existing works for improving the performance of traditional malware detectors.

Feature reduction. A classifier's performance is directly dependent on the quality of features used for model learning. The key idea is to select the least number of features that maximally characterize a malware. Hence, a straight-forward optimization is to conduct a feature reduction step that eliminates redundant features. Li *et al.* [32] develop a fast Android malware detector using an SVM classifier. As a feature reduction step, they perform 'significant permissions' analysis, that selects only the permissions that distinguish between malicious and benign applications with high confidence. Their results achieve up to 32 times speed-up compared to two competing approaches, i.e Drebin [88] and Permission-induced Risk Malware Detection [90]. Similarly, Yerima *et al.*[8] select only the features with maximum Mutual Information (MI) to speed up their malware detector. Firdausi *et al.* [39] demonstrate elevated performance of their malware detector after conducting a best-first feature selection process.

Hardware-assisted detection. Hardware-assisted Malware Detection (HMD) has emerged as an alternative for improving malware detection using Hardware Performance Counters (HPC). HMDs are light-weight detectors that live on the microprocessor to provide a first-line of defense, and to reduce overhead on software-based detectors. Khasawneh *et al.* [40] show that hardware-detectors reduce performance overhead by up to 11 times compared to software-only detectors. Xu *et al.* [91] propose a

novel HMD that monitors system calls' memory access patterns, which are used to classify malware, *e.g.,* kernel rootkits.

Existing studies suggest that HMDs execute a malware sample multiple times in order to collect the required data, due to the limited number of HPCs available on microprocessors. Most of the methods propose to use an ensemble of light-weight classifiers to resolve this issue. Khasawneh *et al.* [40] propose an ensemble of *specialized* LR classifiers to improve the performance of HMDs, while only inducing minimal additional overhead on the cycle-time and power consumption. *Specialized classifiers* are malware-family specific classifiers, *i.e.,* one classifier is trained for one class of malware. They use LR due to its cheap and simple implementation on the microprocessor. Sayadi *et al.* [92] propose ensemble learning to collect the required data while using even less HPCs. Their results show that ensemble learning approach using only 4 HPCs can match the robustness and performance of standard classifiers that use 16 HPCs. In a recent work, Sayadi *et al.* [93] demonstrate that the performance of HMDs is directly related to the number of available HPCs. They propose a feature reduction step in order to select the most significant HPCs. They propose a two-step classification approach: a *course-grained classifier* that categorizes a software as either goodware or one of the malicious classes (*i.e.,* rootkit or trojan); followed by a *fine-grained specialized classifier* (*i.e.,* one for each type of malicious class). To further reduce the run-time, they utilize ensemble learning in the coarse-grained classifiers and show that using merely 4 HPCs outperforms state-of-the-art classifiers with 8 HPCs by a factor of 1.31.

3.6 Trend.

There is a growing interest towards alternative approaches for malware detection, such as causality-based, and ant-colony optimization-based approaches [94], and a more targeted focus on Android malware detection. The use of sequence learning is growing, especially due to superior performance of recurrent neural networks, such as LSTMs, but also due to sequences being better equipped to characterize behavior. For example, Amer *et al.* [78] have used a combination of Word2Vec and Markov chains to establish the relationship between malware API call sequences. Despite that, works on network-traffic based sequential models are meagre due to the difficulty of handling non-stationary and noisy sequences. There is also a growing concern for the brittle nature of neural networks, to be discussed in Section 4.2, which is driving research towards better interpretability of such models' output.

4 Additional research directions

Although malware detection is a central research objective, there are additional research directions that have been gaining traction lately. *Malware analysis*, as opposed to detection, aims to improve malware understandability rather than to optimize detection rates [13,95,96]. *Adversarial machine learning* techniques have gained particular popularity in recent years in relation to detecting evasive malware. Finally, *attributing malware* to its author(s) is also an area of interest, mainly driven by law enforcement agencies. In this section, we discuss the seminal works in these three popular research themes.

4.1 Malware analysis

Malware analysis methods aim to improve malware understandability, and provide essential insights that can improve malware detection methods. Although malware analysis and detection are often seen together in literature, below we present approaches that: (a) aim to improve malware understandability, *e.g.,* by providing insights into malware relationships, and (b) enable malware analysis, *e.g.,* by collecting traces and building analysis environments.

Ucci *et al.* [2] present a recent survey on ML-based malware analysis techniques. The survey provides a taxonomy of research objectives, features and ML algorithms used for Windows PE malware. They identify topical trends on malware triage. They also present the concept of *Malware Analysis Economics* that studies the trade-off between detection accuracy and the resources required for detection.

Increasing interpretability. Malware analysts have to frequently monitor large-scale network traffic, which is a laborious task. Zhang *et al.* [59] propose a framework to visualise the causal relationships between network requests to help detect abnormal events. Their user studies reveal that visualising network traffic in this way enhances analysts' malware detection capabilities. Mariconti *et al.* [64] perform causality analysis on user actions that trigger a malware infection. They characterize malware samples by the trigger-actions commonly performed by users. Their method can successfully infer relations between, *e.g.,* information-stealing malware and web pages asking for user credentials. Suarez *et al.* [13] build a dendrogram of malware families showing overlapping code snippets, which helps them to generate evolution-invariant signatures.

Smith *et al.* [97] have pointed towards the semantic gap between the machine learning and malware analysis communities. One of their proposals is to reposition the task from *identifying malware* to *identifying behavior*, making it possible to understand what a malware is doing. Along these lines, Nadeem *et al.* [98] have proposed the use of behavioral profiles to describe malware samples as opposed to using black-box family names. They develop MalPaCA, a clustering-based framework that discovers distinct behaviors present in network traffic and uses the cluster membership information to generate a profile for each malware sample.

Collecting traces. Collecting malware traces, especially for dynamic analysis, is a challenging problem due to the difficulty of finding live malware samples and setting up sandboxes. Burguera *et al.* [36] addresses the unavailability of malware datasets by setting up a crowd-sourcing system to collect system traces from unlimited number of real smartphone users. Secondly, effective features for malware are often not shared among the security community. Gu *et al.* [34] address this issue by introducing a consortium blockchain framework. The blockchain is used as a database of malware-characterizing features. Their classifier consumes the blockchain for malware family identification. Recently, Shibahara *et al.* [75] have proposed a machine learning-based data collection method for efficient dynamic analysis. Typically, malware traces are collected for a fixed amount of time before moving on to executing the next sample. The method proposed in [75] treats network traces as natural language, and uses the

communication pattern as a heuristic to suspend analysis. They use RNN to learn the underlying objective of communication and to detect when a change in purpose occurs. They suspend analysis when a malware has stopped its activities. With this approach, they report a reduction of 67.1% analysis time.

Sandboxing. Multiple works have proposed sandboxes that can forcefully trigger malware functionality in order to provide more holistic behavioral logs, but very few have used machine learning to do so. There also exist other approaches that use search-based algorithms to improve code coverage of malware samples, *e.g.,* Wang *et al.* [99] propose a fuzzing-based approach to forcefully trigger malware's hidden behaviors. Among the literature that uses machine learning is the work by Spreitzenbarth *et al.* [45]. They propose an end-to-end analysis environment for Android malware where applications are executed, traces are collected and a clustering algorithm categorizes them as malware or goodware. However, they only use machine learning to *post-process* behavioral traces. Additionally, their sandbox does not support latest Android versions. Yerima *et al.* [100] have recently proposed a machine learning based malware analysis framework. They learn a state machine of each Android application using code's static analysis. They use insights from the state machine to guide the so-called *stateful event generation*. They also compare with an existing approach based on random event-generation and show that the guided behavior-triggering approach results in better data collection.

An orthogonal research objective is *sandbox evasion*, where a malware uses machine learning to detect whether it is being executed in a sandbox or on a live system. When malware detects the presence of a sandbox, it either shuts down, or starts sending garbage data to mislead analysis. Yokoyama *et al.* [101] show that it is possible for attackers to use straightforward machine learning algorithms to differentiate between a sandbox and a live system based on leaking characteristics of Windows-based sandboxes. Miramirkhani *et al.* [41] propose sandbox evasion techniques based on the natural 'wear and tear' of a real system compared to that of a sandbox. They exploit the past usage of a system to determine its age and degree of use. They show that a simple decision tree classifier can differentiate between a sandbox and a real system with a very high accuracy.

Trend. The security community appears to be heavily biased towards detection-based solutions. Analysis is most often conducted as a precursor for detection methods, or as a part of Systematization of Knowledge studies. Recently, there has been a push towards using explainable machine learning for the malware domain, which specifically allows to reason about classifier decisions. Fan *et al.* [102] have evaluated various explanation techniques for malware analysis, and conclude that LIME [103] and SHAP [104] provide the most robust and stable explanations. Also, machine learning has not yet been applied to malware lineage — how a certain malware family evolves over time in terms of structure, behavior and its target. Most of the work in this domain is manual and requires a deep understanding of the evolving threat landscape: *e.g.,* Black *et al.* [96] perform an in-depth analysis of banking malware families, and Moubarak *et al.* [95] discuss the structural relationship between several potentially state-sponsored malware. Evidently, the success of this research is dependent on the quality and size of the used dataset.

4.2 Adversarial malware

The security of machine learning is an active area of research that has been gaining increasing popularity in the recent years. This theme addresses the arms-race between crafting evasive malware samples (offensive security) and developing robust methods to detect said samples (defensive security). Although the threat landscape is already adversarial in nature, many approaches in this area have been borrowed from the computer vision domain, where adversarial ML was pioneered. Biggio *et al.* [105] provide an overview of the developments in adversarial machine learning in the past ten years. An open problem in this area is using machine learning to craft adversarial malware samples where the perturbations are big enough to mimic goodware while preserving malicious functionality.

Offensive security. While evasive malware has existed for a long time, latest research applies machine learning to automatically craft these samples. These techniques work by performing small perturbations on a malware sample to create a variant that leads to a misclassification by the ML model. Most of the proposed attacks are gradient-based, as they target deep neural networks. White-box techniques require some knowledge of the target, such as the structure and weights of the target model, while black-box techniques do not assume any knowledge of the targeted classifier.

There are two main concerns in creating adversarial malware samples: (a) the perturbations are performed in the continuous domain, while malware binaries exist in the discrete domain; and (b) the frameworks often create perturbations that break functionality of the executable. Anderson *et al.* [106] have proposed a reinforcement learning-based method to guide the search for functionality-preserving perturbations. However, since their method is quite general, they report modest evasion rates. Grosse *et al.* [107] propose a method for crafting adversarial examples that operates in the discrete domain and preserves functionality. They craft adversarial Android malware by adding constraints to the perturbations — they only allow changes in the manifest file that adds a single line of code to the application. They use the adversarial examples on Drebin [88] and report a misclassification (evasion) rate of 69%. Hu *et al.* [76] target RNN models based on sequential API features. They learn a local substitute (surrogate) model of the victim RNN that propagates the gradients to a generative RNN that produces sequential adversarial examples. Their results show that more than 90% of the adversarial examples result in misclassifications. Kolosnjaji *et al.* [108] target a sequential model that learns from raw malware bytes. They craft adversarial examples using a gradient-based attack modifying the last 1% of the bytes that achieve misclassifications. They report a maximum evasion rate of 60%. Al-Dujaili *et al.* [22] adapt the saddle-point optimization problem from the continuous domain to generate adversarial examples in the discrete domain. They present a framework that discovers adversarial examples and incorporates them in the training process to harden the learnt classifier. They conclude that the randomized rounding technique helps discover four times as many adversarial examples. Chen *et al.* [109] craft adversarial Android malware by optimally perturbing the Dalvik byte code to target semantic features. They show the effectiveness of their adversarial examples by using them on two famous Android malware detectors, MaMaDroid [110] and Drebin [88], where they report an evasion rate of 100%. Recently, Verwer *et al.* [111]

used [22] to develop GRAMS, which is a greedy approach that randomly flips bits to obtain functionality-preserving high-quality adversarial examples in the discrete domain. GRAMS was successful in crafting evasive malware and defending against competitors' evasion attempts during the robust malware detection challenge[8].

Poisoning attacks are another important concern for machine learning classifiers. Poisoning attacks refer to an attacker's capability to inject adversarial examples during classifier *training phase*, such that it learns to classify malicious entities as benign. For example, Biggio *et al.* [112] poison behavioral malware clustering and Muñoz-González *et al.* [113] propose a poisoning algorithm for deep learning classifiers. Chen *et al.* [114] have specified attacker models for poisoning attacks in the malware domain: (a) a *weak attacker* who injects malicious code in the non-logical part of the application, such as manifest file; (b) a *strong attacker* who injects malicious code in resources, such as `jar` or `jpg`; and (c) a *sophisticated attacker* who uses Dynamic Code Loading via Reflection for injecting malicious code at run-time. Having concrete attacker models provides terminology to develop more streamlined defenses, and to compute resilience guarantees.

Defensive security. One approach for forensic malware analysis is to categorize malware based on similar evasion strategies. Kirat *et al.* [71] propose a bioinformatics-inspired solution to generate and analyze evasion signatures — they cluster similar evasive behavior among malware samples. They use a sequence alignment algorithm to measure similarity among different system call sequences. Then, they extract evasion signatures from the behavioral clusters. These signatures can be used to detect when a future malware sample attempts to evade detection in a similar way.

One of the key benefits of adversarial machine learning is that it hardens the security of an adversarially trained model [107]. When adversarial examples are part of the training process, they allow to discover samples in the so-called *blind spots* of the malicious domain, increasing its robustness to unseen evasive samples. Recently, there has been a lot of interest in developing ML-based adversary-aware approaches. The main difference from malware detectors previously discussed in Section 3 is that these approaches actively anticipate evasion attempts. Demontis *et al.* [115] propose a so-called *secure-learning* paradigm that suggests having the feature weights more evenly distributed in order to bound linear classifiers' sensitivity to feature changes. They also propose attacker models based on their capabilities, knowledge and skills. Zhang *et al.* [116] propose an adversary-aware feature selection method, since the choice of features may also be a factor in adversarial robustness. Their wrapper-based framework makes assumptions about the adversary and simulates evasion attacks at each step of the training phase. The framework chooses the features that maximize the classifier's generalizability in the absence of adversarial examples and minimize the classifier's impact against evasion attempts. Chen *et al.* [117] present a robust malware detection system based on two key components: (a) a *feature selection method* that picks features that maximize attacker's evasion costs, and (b) an *ensemble learning method* with diverse classifiers that incorporate a major part of the feature space. Li *et al.* [118] investigate the resilience of ensemble classifiers and the effectiveness of ensemble attacks. Their

[8] https://github.com/ALFA-group/malware_challenge

experiments show that while adversarial training for ensemble classifiers promotes robustness, they are unfortunately no match for adversarial examples learned through ensemble methods.

Another way to harden classifiers without explicit adversarial training is through the special handling of suspicious files. Chen *et al.* [114] have developed a self-adaptive learning scheme for detecting poisoning attacks. They introduce a so-called *camouflage detector* that finds suspicious false negatives by performing similarity analysis with the most-benign and most-malicious looking samples, and sends such *camouflaged samples* back to the training phase as malicious examples.

Trend. At the moment of writing, the security community seems to have a strong affinity towards offensive security research. Naturally, conducting defensive research is especially challenging due to the strict requirements that a defensive framework is expected to fulfill. In case of malware defenses, provable robustness to evasion is a major milestone that cannot be reached by the community without public datasets of evasive malware. We believe that adversarial learning for defensive model hardening is an unfolding but promising research field.

4.3 Malware author attribution

In general, authorship attribution can be considered from two perspectives: (i) *code authorship attribution* – attributing a software to its author(s); and (ii) *family attribution* – identifying code similarities between unlabelled software pieces. In malware research, the latter problem appears as *malware family identification*, which involves multi-class classification already covered in Section 3. Hence, here we discuss the code authorship attribution problem.

Code authorship attribution has a rich history in the Software Engineering literature. The aim of this research is to extract features that capture an author's programming style. Existing work can also be found in related fields, such as forgery and plagiarism detection where the goal is to extract distinguishing stylistic or fingerprinting attributes from a software that identifies where the code was copied from. Source code attribution is the simplest variant since the author's stylistic features can be relatively easily extracted. One such work is proposed by Alsulami *et al.* [119] who extract features from the Abstract Syntax Tree (AST) of source code collected from Google Code Jam (GCJ). In fact, GCJ and Github are popular sources of experimental data for authorship attribution, in general [120,121,122,123].

In many real-world settings however, source code is not directly available, rendering aforementioned techniques ineffective. It is also commonly believed that the compilation process removes most of the stylistic features. Rosenblum *et al.* [120] perform one of the first attempts to address this problem by using a ML approach that identifies the surviving stylistic features for binary authorship attribution.

It is noteworthy that code authorship attribution for malware is significantly more difficult because the authors have strong incentives to hide their identity. Using machine learning to solve the attribution problem is also tricky because code samples from known malware authors that are required to train a classifier are rarely available. Additionally,

the availability of malware-as-a-service indicates that samples are authored by multiple developers in a malware's lifetime. Hence, research in this area is scarce because of the difficulty of establishing a ground truth.

Saxe *et al.* [124] have written an introductory book on big data analysis for malware detection. They show the usage of static and dynamic analysis for performing *shared code analysis* with the aim of identifying similar adversary groups. Alrabaee *et al.* [121] propose a multi-layered approach to improve malware binary authorship attribution by conducting both syntax- and semantic- analysis. They attempt to reconstruct the source code from malware binaries, which they compare with code-based signatures of other families. They also extract semantic features, such as the way registers are manipulated, to establish strong evidence for attribution. Rosenberg *et al.* [125] use deep neural networks for the attribution of nation-state Advanced Persistent Threats (APTs). They observe that nation-state APTs have different styles and objectives, which makes their classification feasible. To that end, they take raw dynamic logs as input to the neural network that learns a high-level abstraction of the APTs. Their system is evaluated on two major nation-state malware families and show it to be effective for the purpose.

Natural Language Processing has also been proposed for attribution purposes after its success in attributing cyber-stalkers [72]. Kalgutkar *et al.* [74] build an Android malware detection system based on 'malware author' signatures. Their system leverages strings extracted from the malware binaries to generate profiles of malware authors, with the expectation that future malware samples authored by the same person will match the profiles. They use n-grams for feature representation and Support Vector Machine for APK classification.

Research related to adversarial attacks also exists in code authorship attribution. Simko *et al.* [123] propose adversarial stylometry attacks to defeat source code attribution classifiers. They demonstrate that current code attribution classifiers are not robust to adversarial attacks, even when they are executed by non-experts. The authors claim that although not fool-proof, augmenting machine learning classifiers with human analysts proves to be more resilient against adversarial attacks, especially when they are warned about potential forgeries in the code. They analyze C/C++ programs and conclude that semantic features, such as those extracted from ASTs are more resilient to forgery attacks.

Alrabaee *et al.* [122] present a literature survey of existing techniques for malware binary attribution. The survey also lists features that can be used for author attribution because they survive compilation, *e.g.,* compiler information, system calls, and the usage of particular strings may characterize coding styles. Additionally, certain type of bugs in the code may also point to semantic hints that can be used for author attribution. They also note that a key research challenge is feature selection that captures author's style rather than functionality of the program. Following this, Murenin *et al.* [126] have used LIME to understand the role of selected features in source code attribution for Android malware.

Iqbal *et al.* [127] have recently published a book on authorship attribution and cyber forensics using machine learning in which they comprehensively describe research into authorship identification and attribution using few training samples, authorship characterization and verification. Kalgutkar *et al.* [128] show how the field has evolved

from basic software matching techniques to sophisticated methods based on API calls and dependency graphs. They conclude that although there is no one-size-fits-all solution yet for malware attribution, the existing work on varying levels of abstraction has brought us one step closer to the solution. Nevertheless, this field still has many open problems that are yet to be explored.

Trend. The popularity of the malware attribution field is impacted by the adversarial and ad-hoc nature of the threat landscape. Unavailability of open datasets further complicates realistic evaluation. Further, having a narrow target audience (*e.g.,* law enforcement, intelligence agencies) means that the works do not get highly cited and thus remain undiscovered. We believe that this field can get a new life with explainable approaches, open benchmark datasets and access to ground truth.

5 Challenges in ML-applied malware defenses

The malware domain presents unique challenges for machine learning application. After years of research, the security community has made a significant headway in highlighting the proper usage of machine learning for malware defenses. As a result, additional problems have emerged that require further investigation. Souri *et al.* [82] and Ye *et al.* [3] identify several unsolved problems in the data-mining based malware detection domain. One must remember that machine learning is not a silver bullet that can solve all malware-related problems [128]. In this section, we describe common pitfalls and challenges that emerge when ML is used for malware detection, which should be accounted for when designing and evaluating such methods.

Robustness against time-decay. Some of the existing work is filled with unrealistic simplifying assumptions about the malware landscape. One of the most prevalent assumptions is the *closed-world assumption*, which assumes that training data is fully representative of all categories of samples that may appear at test-time. However, as malware is an ever-evolving threat, static training data will inevitably become outdated. Consequently, researchers have shown that ML-classifiers' performance degrades over time [20,129]. Recent works have incorporated *concept drift detection* in their ML classifiers for handling non-stationary data population. These classifiers continually re-learn the changing concepts in order to maintain an acceptable detection accuracy. Jordaney *et al.* [130] and Wang *et al.* [131] use P-values that can proactively detect concept drift before the classifier's performance starts to degrade. There is also a growing interest in semantic features that are less affected by malware evolution and hence slow down the aging of malware detectors [132].

Robustness against evasion. Evasion resilience is an important characteristic for deployable classifiers. A misleading expectation from ML classifiers is that they should be *fully and provably evasion resilient*. To this end, defensive adversarial machine learning has emerged as a promising solution for evasion resilient classifiers, which has been previously discussed in Section 4.2. The purpose of defensive adversarial ML is to explore additional search space in order to harden models against evasion attempts. However, this search is still bounded by the Independent and Identically Distributed data

(i.i.d) assumption. As a consequence, out-of-distribution adversarial examples prevalent in the open world are unlikely to be detected by an adversarially trained ML classifier.

Imbalanced training-set. Benign examples occur significantly more frequently than malicious ones. Failure to incorporate this trait in the training dataset creates a so-called *spatial-bias* [129] in the classifier. Existing works have often used unrealistic class distribution, *e.g.,* the use of inverted class distribution [6] and equal class distribution [133,134]. Chen *et al.* [48] propose a solution for imbalanced network traffic classification to perform accurate Android malware detection. They experiment with various combinations of imbalanced classification algorithms, such as Synthetic Minority Oversampling Technique (SMOTE) with SVM, SVM cost-sensitive and `C4.5` cost-sensitive. They also develop Simplex Imbalanced Data Gravitation-based Classification (S-IDGC) that works faster while maintaining the stability of IDGC. In the deep learning domain, Le *et al.* [26] use the class re-balance sampling procedure in bi-directional LSTMs to address the class imbalance problem.

Evaluation metrics. The usage of appropriate evaluation metrics is an underrated challenge. For example, using accuracy to measure a classifier's performance when it is trained with a highly imbalanced dataset results in misleading conclusions. Similarly, precision and recall values can also be altered based on the choice of empirical thresholds and dataset-specific parameters. Meanwhile, evaluating approaches using such varied metrics limits objective analysis, and the obtained results become incomparable. Jordaney *et al.* [135] demonstrate that traditional evaluation metrics show misleading information about classifiers' performance. They propose two metrics based on nonconformity measures for evaluating a classifiers' performance. *Credibility* measures the homogeneity of a given label compared to others of the same class, and *Confidence* measures the separation between a given label and other classes. Pendlebury *et al.* [129] have recently identified experimental biases in existing Android malware classifiers, namely (a) *spatial bias* due to unrealistic class distribution in training and testing data, and (b) *temporal bias* due to incorrect time splits causing impossible configurations. They propose a new metric, namely Area Under Time (AUT), to characterize classifier robustness when time decay is present.

Privacy concerns. Machine learning classifiers typically perform better with fine-grained contextual features. In an attempt to perform large-scale classification, classifiers have access to both benign and malicious data. Privacy concerns arise as the feature-set becomes more and more fine-grained. For example, DPI-based approaches analyze the payload of each packet, which may contain privacy sensitive information. With data protection laws being widely enforced, such methods are tricky to deploy at large-scale. There are a couple of solutions for being privacy-aware: (i) selecting abstract features that do not violate the privacy of user actions, while still being able to characterize malicious behavior; (ii) deploying a distributed classifier, as in the case of federated learning, that trains on local data provided by multiple clients [136]. In the latter solution, secure multi-party computation (SMC) and differential privacy (DP) are required to provide privacy guarantees.

Performance optimizations. Malware infections are a widespread security threat faced by all network-connected devices. Classical machine learning solutions are often not ideal for fending off millions of malware infections each day. Effective intelligent

defenses should be fast, proactive and evolve with the changing threat landscape. Hence, a dedicated research direction exists that designs online, optimized classifiers capable of detecting malware in real-time. Federated learning, discussed earlier as a possible solution to privacy issues, provides a distributed infrastructure that enables efficient large-scale detection. Other works are discussed in Section 3.5.

6 Open problems in ML-based malware defenses

In this section, we discuss what we believe are the key problems that the research community should address. At the heart lies the problem of reproducible research: the absence of *toy problems* and *representative datasets* makes the results from different papers incomparable. Furthermore, the results often cannot be taken at their face value because malware *ground truth* is inherently inconsistent and unreliable. Crucially, many solutions eagerly emphasize metric optimization but overlook *explainability*, providing little new insight into the problem of malware detection. We believe that these four issues hinder fair assessment of new contributions in the intelligent malware defenses domain. It is very difficult to objectively compare new methods against state-of-the-art solutions for the same problem, using the same data, and the same ground truth. An alarming side-effect is the lack of meaningful contributions to the field even though many new papers are published each year.

Toy problems. Toy problems are important in the early development of a research field. These are simplified challenges that can help develop and test methodologies that solve a more challenging problem. Computer science in general and artificial intelligence in particular have established traditional toy problems that are still used to develop newer methodologies. However, malware research has always aimed to solve real-world threats. We observe that limited access to data and resources that are necessary for the evaluation of proposed methodologies has constrained systematic and open academic research. Moreover, building fool-proof methodologies in malware detection is an especially challenging problem, because the adversaries keep evolving rapidly. Since malware is constantly evolving, the research is driven by the availability of newer threats, and is reactive in nature. In light of these inevitable issues, we urge the community to introduce standardized toy problems, which could act as a starting point for developing new methods in a more synchronized and proactive manner. Toy problems would allow the assessment of proposed algorithms in isolation from the general practical limitations of malware detection. Ultimately, as the algorithms become more mature, they should be enhanced for deployment — practical feasibility should not be fully discarded at the envisioning stage.

Representative datasets. The biggest hurdle in ML-based malware analysis research is the absence of representative datasets. These datasets are crucial for the development of usable and generalizable defensive solutions. However, with the rapid evolution of malware, any available dataset becomes obsolete in a matter of years, *e.g.,* the well-known VX Heavens dataset [137] from 2010, the Drebin dataset [88] from 2010-2012, and the MalGenome dataset [87] from 2012 are arguably no longer representative. Since most of the available datasets are not representative, the trained models only describe part of the real threat landscape. This is not to say that open-source datasets are not available.

In fact, the Stratosphere IPS project[9] has published large-scale network traffic, *e.g.,* the CTU-13 dataset [138] captures traffic for 13 botnet scenarios, and the recently published IoT-23 dataset [139] captures 20 IoT malware scenarios and 3 benign ones. The Kaggle Microsoft Malware dataset [85] was also widely used in multiple works. Other works have also released their private datasets for reproducibility [29,140]. Nevertheless, one promising way for the academic community to gain access to reliable and representative data is to establish a long-term collaboration with industry partners who directly monitor the threat landscape and can provide updated threat intelligence for the development of robust machine learning solutions. This is an excellent way to keep up with the rapidly changing threat landscape. The downside is that the data often contains highly sensitive information that cannot be released to the public, thus exacerbating problems of reproducibility.

Noisy ground truth. Existing literature has repeatedly shown that AV-provided malware family labels are inconsistent. These labels are used as ground truth by researchers to evaluate newer malware detection methods, making the results unreliable. Popular tools, such as VirusTotal[10], run multiple AV scanners and return an array of labels predicted by each scanner, without any indication as to which is correct. There is also an absence of a common vocabulary that all security companies can follow to label malware samples. Research has shown that the consensus reached by AV scanners regarding the labeling of a single malware sample is no better than a coin-toss (around 50%) [141]. Machine learning has also been slow to materialize in network security domain due to non-stationary data and noisy ground truth [1]. Unsupervised ML can already provide a foundation to address this issue. However, in practice, existing unsupervised ML approaches often use some form of ground truth for evaluation. For example, Perdisci *et al.* [47] evaluate their malware clustering by introducing a notion of AV graphs that depict the agreement between AV vendors as a measure of cluster cohesion and separation. Yuping *et al.* [142] use majority-voted family labels from 25 AV vendors as their ground truth to evaluate malware clustering. Li *et al.* [143] have advised caution when deciphering highly accurate clustering results as they can be impacted by spatial bias: performing majority voting on AV-provided labels is hazardous, since if most of the AV vendors are in agreement, it typically indicates that the families are already easy to detect. Hence, we either need better ground truth [144] or purely data-driven unsupervised evaluation approaches.

Explainable solutions. In recent works, deep learning based malware detectors have surpassed the performance of traditional ML classifiers. They have also automated the detection pipeline for the most part. However, deep neural networks are inherent black-boxes that provide limited interpretability. It is also alarming how brittle deep learning is to adversarial attacks. Alternatively, non-deep learning approaches are not much more interpretable — they are frequently packed with complicated filtering steps to maximise performance [52], which also turns them into black-boxes. This concern has motivated research on explainable machine learning. Explainable models enable identification of bias in raw data, debug errors in trained models, enable model optimization, and allow analysts to extrapolate advanced results, *i.e.,* to get detailed insights from data instead

[9] https://www.stratosphereips.org/datasets-overview
[10] https://www.virustotal.com/

of simply reading off detection rates. Without explainability, such extrapolation will be difficult and error-prone. Mathews *et al.* [145] provide a summary of explainable ML techniques for malware classification, including both intrinsic and post-hoc methods. The research trend shows that moving forward, special emphasis will be given to explainable and human-in-the-loop solutions.

7 Summary

Machine learning has emerged as a promising ally for developing intelligent malware defenses. However, the research in this area is scattered across different venues and domains. In this chapter, we identify the key research themes and assemble the state-of-the-art literature that has been proposed in the past decade. In doing so, we highlight trends in these research themes.

The literature is greatly dominated by *malware detection* approaches with the aim of developing scalable behavioral signatures. We categorize the research in this domain according to the data source and feature representation used for their classifiers. The trends in the literature suggest that sequence learning and explainable machine learning are considered promising areas of research. *Malware analysis* is another research direction that develops tools that provide the necessary insights to improve malware detection. *Adversarial machine learning* has recently gained popularity to harden machine learning classifiers. Also, *malware author attribution* proves to be a challenging field with limited progress due to the unavailability of datasets, and an absence of concrete problem statements that data-driven methods can realistically address.

We have discussed important considerations that emerge when machine learning is applied in the malware domain, such as resilience against concept drift and evasion, handling imbalanced datasets and using appropriate evaluation metrics. We have also identified key issues that need to be addressed in our opinion by the research community in order to encourage systematized research in the malware domain: toy problems, representative datasets, noisy ground truth, and explainable solutions. Without overcoming these issues, limited progress can be made due to the inability to compare research results.

It is evident that intelligent malware defenses will continue to grow. However, understanding the unique challenges that the malware domain brings to the table is absolutely essential for developing effective machine learning enabled solutions that can withstand the test of time.

References

1. B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1723–1732, 2017.
2. D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, 2019.
3. Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, pp. 1–40, 2017.

4. B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions," *Computers & Security*, vol. 74, pp. 144–166, 2018.
5. S. Eskandari, A. Leoutsarakos, T. Mursch, and J. Clark, "A first look at browser-based cryptojacking," in *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 58–66, IEEE, 2018.
6. F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Computing*, vol. 20, no. 1, pp. 343–357, 2016.
7. J. Sahs and L. Khan, "A machine learning approach to android malware detection," in *2012 European Intelligence and Security Informatics Conference*, pp. 141–147, IEEE, 2012.
8. S. Y. Yerima, S. Sezer, G. McWilliams, and I. Muttik, "A new android malware detection approach using bayesian classification," in *2013 IEEE 27th international conference on advanced information networking and applications (AINA)*, pp. 121–128, IEEE, 2013.
9. Z. Zhu and T. Dumitraş, "Featuresmith: Automatically engineering features for malware detection by mining the security literature," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 767–778, 2016.
10. N. McLaughlin, J. Martinez del Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupé, *et al.*, "Deep android malware detection," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, pp. 301–308, 2017.
11. E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "Maldozer: Automatic framework for android malware detection using deep learning," *Digital Investigation*, vol. 24, pp. S48–S59, 2018.
12. H.-J. Zhu, Z.-H. You, Z.-X. Zhu, W.-L. Shi, X. Chen, and L. Cheng, "Droiddet: effective and robust detection of android malware using static analysis along with rotation forest model," *Neurocomputing*, vol. 272, pp. 638–646, 2018.
13. G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, and J. Blasco, "Dendroid: A text mining approach to analyzing and classifying code structures in android malware families," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1104–1117, 2014.
14. Z. Li, L. Sun, Q. Yan, W. Srisa-An, and Z. Chen, "Droidclassifier: Efficient adaptive mining of application-layer header for classifying android malware," *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Security and Privacy in Communication Networks*, p. 597–616, 2017.
15. G. Suarez-Tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, and L. Cavallaro, "Droidsieve: Fast and accurate classification of obfuscated android malware," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, pp. 309–320, 2017.
16. H. H. Pajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "Intelligent os x malware threat detection with code inspection," *Journal of Computer Virology and Hacking Techniques*, vol. 14, no. 3, pp. 213–223, 2018.
17. Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in internet-of-things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.
18. A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, "Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 88–95, 2018.
19. H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep recurrent neural network based approach for internet of things malware threat hunting," *Future Generation Computer Systems*, vol. 85, pp. 88–96, 2018.
20. J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 11–20, IEEE, 2015.

21. B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Australasian Joint Conference on Artificial Intelligence*, pp. 137–149, Springer, 2016.
22. A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 76–82, IEEE, 2018.
23. S. Jain and Y. K. Meena, "Byte level n–gram analysis for malware detection," in *International Conference on Information Processing*, pp. 51–59, Springer, 2011.
24. Y. Fan, Y. Ye, and L. Chen, "Malicious sequential pattern mining for automatic malware detection," *Expert Systems with Applications*, vol. 52, pp. 16–25, 2016.
25. E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. K. Nicholas, "Malware detection by eating a whole exe," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
26. Q. Le, O. Boydell, B. Mac Namee, and M. Scanlon, "Deep learning at the shallow end: Malware classification for non-domain experts," *Digital Investigation*, vol. 26, pp. S118–S126, 2018.
27. D. Kirat, L. Nataraj, G. Vigna, and B. Manjunath, "Sigmal: A static signal processing based malware triage," in *Proceedings of the 29th Annual Computer Security Applications Conference*, pp. 89–98, 2013.
28. A. Shabtai, Y. Fledel, and Y. Elovici, "Automated static code analysis for classifying android applications using machine learning," in *2010 International Conference on Computational Intelligence and Security*, pp. 329–333, IEEE, 2010.
29. A. Sami, B. Yadegari, H. Rahimi, N. Peiravian, S. Hashemi, and A. Hamze, "Malware detection based on mining api calls," in *Proceedings of the 2010 ACM symposium on applied computing*, pp. 1020–1025, 2010.
30. M. Alazab, S. Venkatraman, P. Watters, M. Alazab, *et al.*, "Zero-day malware detection based on supervised learning algorithms of api call signatures," *2011 Australasian Data Mining Conference (AusDM 11)*, 2010.
31. Y. Aafer, W. Du, and H. Yin, "Droidapiminer: Mining api-level features for robust malware detection in android," in *International conference on security and privacy in communication systems*, pp. 86–103, Springer, 2013.
32. J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant permission identification for machine-learning-based android malware detection," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216–3225, 2018.
33. D. Yuxin and Z. Siyi, "Malware detection based on deep learning algorithm," *Neural Computing and Applications*, vol. 31, no. 2, pp. 461–472, 2019.
34. J. Gu, B. Sun, X. Du, J. Wang, Y. Zhuang, and Z. Wang, "Consortium blockchain-based malware detection in mobile devices," *IEEE Access*, vol. 6, pp. 12118–12128, 2018.
35. R. S. Pirscoveanu, S. S. Hansen, T. M. Larsen, M. Stevanovic, J. M. Pedersen, and A. Czech, "Analysis of malware behavior: Type classification using machine learning," in *2015 International conference on cyber situational awareness, data analytics and assessment (CyberSA)*, pp. 1–7, IEEE, 2015.
36. I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pp. 15–26, 2011.
37. H. Zhang, W. Zhang, Z. Lv, A. K. Sangaiah, T. Huang, and N. Chilamkurti, "Maldc: a depth detection method for malware based on behavior chains," *World Wide Web*, pp. 1–20, 2019.
38. P. Mishra, K. Khurana, S. Gupta, and M. K. Sharma, "Vmanalyzer: Malware semantic analysis using integrated cnn and bi-directional lstm for detecting vm-level attacks in cloud," in *2019 Twelfth International Conference on Contemporary Computing (IC3)*, pp. 1–6, IEEE, 2019.

39. I. Firdausi, A. Erwin, A. S. Nugroho, *et al.*, "Analysis of machine learning techniques used in behavior-based malware detection," in *2010 Second international conference on advances in computing, control, and telecommunication technologies*, pp. 201–203, IEEE, 2010.
40. K. N. Khasawneh, M. Ozsoy, C. Donovick, N. Abu-Ghazaleh, and D. Ponomarev, "Ensemble learning for low-level hardware-supported malware detection," in *International Symposium on Recent Advances in Intrusion Detection*, pp. 3–25, Springer, 2015.
41. N. Miramirkhani, M. P. Appini, N. Nikiforakis, and M. Polychronakis, "Spotless sandboxes: Evading malware analysis systems using wear-and-tear artifacts," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 1009–1024, IEEE, 2017.
42. P. Burnap, R. French, F. Turner, and K. Jones, "Malware classification using self organising feature maps and machine activity data," *computers & security*, vol. 73, pp. 399–410, 2018.
43. D. H. "Polo" Chau, A. Wright, C. Nachenberg, C. Faloutsos, and J. Wilhelm, "Polonium: Tera-scale graph mining and inference for malware detection," in *Society for Industrial and Applied Mathematics.Proceedings of the SIAM International Conference on Data Mining*, pp. 131–142, 2011.
44. A. Tamersoy, K. Roundy, and D. H. Chau, "Guilt by association: Large scale malware detection by mining file-relation graphs," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (New York, NY, USA), p. 1524–1533, Association for Computing Machinery, 2014.
45. M. Spreitzenbarth, T. Schreck, F. Echtler, D. Arp, and J. Hoffmann, "Mobile-sandbox: combining static and dynamic analysis with machine-learning techniques," *International Journal of Information Security*, vol. 14, no. 2, pp. 141–153, 2015.
46. M. Ijaz, M. H. Durad, and M. Ismail, "Static and dynamic malware analysis using machine learning," in *2019 16th International bhurban conference on applied sciences and technology (IBCAST)*, pp. 687–691, IEEE, 2019.
47. R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of http-based malware and signature generation using malicious network traces.," in *NSDI*, vol. 10, 2010.
48. Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, and B. Yang, "Machine learning based mobile malware detection using highly imbalanced network traffic," *Information Sciences*, vol. 433, pp. 346–364, 2018.
49. A. Boukhtouta, S. A. Mokhov, N.-E. Lakhdari, M. Debbabi, and J. Paquet, "Network malware classification comparison using dpi and flow packet headers," *Journal of Computer Virology and Hacking Techniques*, vol. 12, no. 2, pp. 69–100, 2016.
50. R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1916–1920, IEEE, 2015.
51. O. E. David and N. S. Netanyahu, "Deepsign: Deep learning for automatic malware signature generation and classification," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2015.
52. A. Mohaisen, O. Alrawi, and M. Mohaisen, "Amal: High-fidelity, behavior-based automated malware analysis and classification," *computers & security*, vol. 52, 2015.
53. I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Information Sciences*, vol. 231, pp. 64–82, 2013.
54. N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, "Machine learning aided android malware classification," *Computers & Electrical Engineering*, vol. 61, pp. 266–274, 2017.
55. K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.
56. M. Z. Rafique and J. Caballero, "Firma: Malware clustering and network signature generation with mixed network behaviors," in *International Workshop on Recent Advances in Intrusion Detection*, pp. 144–163, Springer, 2013.

57. L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *ACSAC*, pp. 129–138, ACM, 2012.
58. F. Tegeler, X. Fu, G. Vigna, and C. Kruegel, "Botfinder: Finding bots in network traffic without deep packet inspection," in *CoNEXT*, pp. 349–360, ACM, 2012.
59. H. Zhang, M. Sun, D. Yao, and C. North, "Visualizing traffic causality for analyzing network anomalies," in *Proceedings of the 2015 ACM International Workshop on International Workshop on Security and Privacy Analytics*, pp. 37–42, 2015.
60. S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "Hindroid: An intelligent android malware detection system based on structured heterogeneous information network," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1507–1515, 2017.
61. Y. Fan, S. Hou, Y. Zhang, Y. Ye, and M. Abdulhayoglu, "Gotcha-sly malware! scorpion a metagraph2vec based malware detection system," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 253–262, 2018.
62. H. Zhang, D. D. Yao, N. Ramakrishnan, and Z. Zhang, "Causality reasoning about network events for detecting stealthy malware activities," *computers & security*, vol. 58, pp. 180–198, 2016.
63. Y. Liu, M. Zhang, D. Li, K. Jee, Z. Li, Z. Wu, J. Rhee, and P. Mittal, "Towards a timely causality analysis for enterprise security.," in *NDSS*, 2018.
64. E. Mariconti, J. Onaolapo, G. Ross, and G. Stringhini, "The cause of all evils: Assessing causality between user actions and malware activity," in *10th {USENIX} Workshop on Cyber Security Experimentation and Test ({CSET} 17)*, 2017.
65. L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th international symposium on visualization for cyber security*, pp. 1–7, 2011.
66. M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," in *Proceedings of the sixth ACM conference on data and application security and privacy*, pp. 183–194, 2016.
67. M. Kalash, M. Rochan, N. Mohammed, N. D. Bruce, Y. Wang, and F. Iqbal, "Malware classification with deep convolutional neural networks," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5, IEEE, 2018.
68. H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama, and J. Sakuma, "Malware analysis of imaged binary samples by convolutional neural network with attention mechanism," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, pp. 127–134, 2018.
69. A. Singh, A. Handa, N. Kumar, and S. K. Shukla, "Malware classification using image representation," in *International Symposium on Cyber Security Cryptography and Machine Learning*, pp. 75–92, Springer, 2019.
70. Y. Chen, A. Narayanan, S. Pang, and B. Tao, "Malicioius software detection using multiple sequence alignment and data mining," in *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, pp. 8–14, IEEE, 2012.
71. D. Kirat and G. Vigna, "Malgene: Automatic extraction of malware analysis evasion signature," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 769–780, 2015.
72. I. Frommholz, H. M. Al-Khateeb, M. Potthast, Z. Ghasem, M. Shukla, and E. Short, "On textual analysis and machine learning for cyberstalking detection," *Datenbank-Spektrum*, vol. 16, no. 2, pp. 127–135, 2016.
73. G. Canfora, A. De Lorenzo, E. Medvet, F. Mercaldo, and C. A. Visaggio, "Effectiveness of opcode ngrams for detection of multi family android malware," in *2015 10th International Conference on Availability, Reliability and Security*, pp. 333–340, IEEE, 2015.

74. V. Kalgutkar, N. Stakhanova, P. Cook, and A. Matyukhina, "Android authorship attribution through string analysis," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pp. 1–10, 2018.
75. T. Shibahara, T. Yagi, M. Akiyama, D. Chiba, and T. Yada, "Efficient dynamic malware analysis based on network behavior using deep learning," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2016.
76. W. Hu and Y. Tan, "Black-box attacks against rnn based malware detection algorithms," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
77. B. Cakir and E. Dogdu, "Malware classification using deep learning methods," in *Proceedings of the ACMSE 2018 Conference*, pp. 1–5, 2018.
78. E. Amer and I. Zelinka, "A dynamic windows malware detection and prediction method based on contextual understanding of api call sequence," *Computers & Security*, vol. 92, p. 101760, 2020.
79. S. Garcia, "Modelling the network behaviour of malware to block malicious patterns. the stratosphere project: a behavioural ips," *Virus Bulletin*, 2015.
80. G. Pellegrino, Q. Lin, C. Hammerschmidt, and S. Verwer, "Learning behavioral fingerprints from netflows using timed automata," in *IFIP*, pp. 308–316, IEEE, 2017.
81. C. LeDoux and A. Lakhotia, *Malware and Machine Learning*, pp. 1–42. Springer, 2015.
82. A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Computing and Information Sciences*, vol. 8, no. 1, p. 3, 2018.
83. W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*, pp. 712–717, IEEE, 2017.
84. J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 10, pp. 1619–1630, 2006.
85. R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," *arXiv preprint arXiv:1802.10135*, 2018.
86. K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, pp. 2048–2057, 2015.
87. Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *2012 IEEE symposium on security and privacy*, pp. 95–109, IEEE, 2012.
88. D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket.," in *Ndss*, vol. 14, pp. 23–26, 2014.
89. V. Naidu and A. Narayanan, "Using different substitution matrices in a string-matching technique for identifying viral polymorphic malware variants," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2903–2910, IEEE, 2016.
90. W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X. Zhang, "Exploring permission-induced risk in android applications for malicious application detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 11, pp. 1869–1882, 2014.
91. Z. Xu, S. Ray, P. Subramanyan, and S. Malik, "Malware detection using machine learning based analysis of virtual memory access patterns," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 169–174, IEEE, 2017.
92. H. Sayadi, N. Patel, S. M. PD, A. Sasan, S. Rafatirad, and H. Homayoun, "Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2018.

93. H. Sayadi, H. M. Makrani, S. M. P. Dinakarrao, T. Mohsenin, A. Sasan, S. Rafatirad, and H. Homayoun, "2smart: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 728–733, IEEE, 2019.
94. H. Huang, H. Deng, Y. Sheng, and X. Ye, "Accelerating convolutional neural network-based malware traffic detection through ant-colony clustering," *Journal of Intelligent & Fuzzy Systems*, no. Preprint, pp. 1–15, 2019.
95. J. Moubarak, M. Chamoun, and E. Filiol, "Comparative study of recent mea malware phylogeny," in *Computer and Communication Systems (ICCCS), 2017 2nd International Conference on*, pp. 16–20, IEEE, 2017.
96. P. Black, I. Gondal, and R. Layton, "A survey of similarities in banking malware behaviours," *Computers & Security*, 2017.
97. M. R. Smith, N. T. Johnson, J. B. Ingram, A. J. Carbajal, B. I. Haus, E. Domschot, R. Ramyaa, C. C. Lamb, S. J. Verzi, and W. P. Kegelmeyer, "Mind the gap: On bridging the semantic gap between machine learning and malware analysis," in *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, pp. 49–60, 2020.
98. A. Nadeem, C. Hammerschmidt, C. H. Gañán, and S. Verwer, "Beyond labeling: Using clustering to build network behavioral profiles of malware families," *Malware Analysis Using Artificial Intelligence and Deep Learning*, p. 381, 2021.
99. X. Wang, Y. Yang, and S. Zhu, "Automated hybrid analysis of android malware through augmenting fuzzing with forced execution," *IEEE Transactions on Mobile Computing*, vol. 18, no. 12, pp. 2768–2782, 2018.
100. S. Y. Yerima, M. K. Alzaylaee, and S. Sezer, "Machine learning-based dynamic analysis of android apps with improved code coverage," *EURASIP Journal on Information Security*, vol. 2019, no. 1, p. 4, 2019.
101. A. Yokoyama, K. Ishii, R. Tanabe, Y. Papa, K. Yoshioka, T. Matsumoto, T. Kasama, D. Inoue, M. Brengel, M. Backes, *et al.*, "Sandprint: fingerprinting malware sandboxes to provide intelligence for sandbox evasion," in *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 165–187, Springer, 2016.
102. M. Fan, W. Wei, X. Xie, Y. Liu, X. Guan, and T. Liu, "Can we trust your explanations? sanity checks for interpreters in android malware analysis," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 838–853, 2020.
103. M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.
104. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in Neural Information Processing Systems*, vol. 30, pp. 4765–4774, 2017.
105. B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.
106. H. S. Anderson, A. Kharkar, B. Filar, and P. Roth, "Evading machine learning malware detection," *black Hat*, 2017.
107. K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *European Symposium on Research in Computer Security*, pp. 62–79, Springer, 2017.
108. B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 533–537, IEEE, 2018.
109. X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, and K. Ren, "Android hiv: A study of repackaging malware for evading machine-learning detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 987–1001, 2019.

110. L. Onwuzurike, E. Mariconti, P. Andriotis, E. D. Cristofaro, G. Ross, and G. Stringhini, "Mamadroid: Detecting android malware by building markov chains of behavioral models (extended version)," *ACM Transactions on Privacy and Security (TOPS)*, vol. 22, no. 2, pp. 1–34, 2019.
111. S. Verwer, A. Nadeem, C. Hammerschmidt, L. Bliek, A. Al-Dujaili, and U.-M. O'Reilly, "The robust malware detection challenge and greedy random accelerated multi-bit search," in *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, pp. 61–70, 2020.
112. B. Biggio, K. Rieck, D. Ariu, C. Wressnegger, I. Corona, G. Giacinto, and F. Roli, "Poisoning behavioral malware clustering," in *Proceedings of the 2014 workshop on artificial intelligent and security workshop*, pp. 27–36, 2014.
113. L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 27–38, 2017.
114. S. Chen, M. Xue, L. Fan, S. Hao, L. Xu, H. Zhu, and B. Li, "Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach," *computers & security*, vol. 73, pp. 326–344, 2018.
115. A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli, "Yes, machine learning can be more secure! a case study on android malware detection," *IEEE Transactions on Dependable and Secure Computing*, 2017.
116. F. Zhang, P. P. Chan, B. Biggio, D. S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," *IEEE transactions on cybernetics*, vol. 46, no. 3, pp. 766–777, 2015.
117. L. Chen, S. Hou, and Y. Ye, "Securedroid: Enhancing security of machine learning-based detection against adversarial android malware attacks," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, pp. 362–372, 2017.
118. D. Li and Q. Li, "Adversarial deep ensemble: Evasion attacks and defenses for malware detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3886–3900, 2020.
119. B. Alsulami, E. Dauber, R. Harang, S. Mancoridis, and R. Greenstadt, "Source code authorship attribution using long short-term memory based networks," in *European Symposium on Research in Computer Security*, pp. 65–82, Springer, 2017.
120. N. Rosenblum, X. Zhu, and B. P. Miller, "Who wrote this code? identifying the authors of program binaries," in *European Symposium on Research in Computer Security*, pp. 172–189, Springer, 2011.
121. S. Alrabaee, N. Saleem, S. Preda, L. Wang, and M. Debbabi, "Oba2: An onion approach to binary code authorship attribution," *Digital Investigation*, vol. 11, pp. S94–S103, 2014.
122. S. Alrabaee, P. Shirani, M. Debbabi, and L. Wang, "On the feasibility of malware authorship attribution," in *International Symposium on Foundations and Practice of Security*, pp. 256–272, Springer, 2016.
123. L. Simko, L. Zettlemoyer, and T. Kohno, "Recognizing and imitating programmer style: Adversaries in program authorship attribution," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 1, pp. 127–144, 2018.
124. J. Saxe and H. Sanders, *Malware Data Science: Attack Detection and Attribution*. No Starch Press, 2018.
125. I. Rosenberg, G. Sicard, and E. O. David, "Deepapt: nation-state apt attribution using end-to-end deep neural networks," in *International Conference on Artificial Neural Networks*, pp. 91–99, Springer, 2017.

126. I. Murenin, E. Novikova, R. Ushakov, and I. Kholod, "Explaining android application authorship attribution based on source code analysis," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, pp. 43–56, Springer, 2020.
127. F. Iqbal, M. Debbabi, and B. C. Fung, *Machine Learning for Authorship Attribution and Cyber Forensics*. Springer, 2020.
128. V. Kalgutkar, R. Kaur, H. Gonzalez, N. Stakhanova, and A. Matyukhina, "Code authorship attribution: Methods and challenges," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–36, 2019.
129. F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "Tesseract: Eliminating experimental bias in malware classification across space and time," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 729–746, 2019.
130. R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in *26th USENIX Security Symposium (USENIX Security 17)*, pp. 625–642, 2017.
131. Z. Wang, M. Tian, and C. Jia, "An active and dynamic botnet detection approach to track hidden concept drift," in *International Conference on Information and Communications Security*, pp. 646–660, Springer, 2017.
132. X. Zhang, Y. Zhang, M. Zhong, D. Ding, Y. Cao, Y. Zhang, M. Zhang, and M. Yang, "Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 757–770, 2020.
133. Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 114–123, 2016.
134. N. Peiravian and X. Zhu, "Machine learning for android malware detection using permission and api calls," in *2013 IEEE 25th international conference on tools with artificial intelligence*, pp. 300–305, IEEE, 2013.
135. R. Jordaney, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Misleading metrics: On evaluating machine learning for malware with confidence," *Tech. Rep.*, 2016.
136. T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 756–767, IEEE, 2019.
137. V. Heaven, "Vx heaven virus collection 2010-05-18," 2010-05-18.
138. S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.
139. A. Parmisano, S. Garcia, and M. J. Erquiaga, "Stratosphere laboratory. a labeled dataset with malicious and benign iot network traffic," 2020.
140. M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. Van Der Veen, and C. Platzer, "Andrubis–1,000,000 apps later: A view on current android malware behaviors," in *2014 third international workshop on building analysis datasets and gathering experience returns for security (BADGERS)*, pp. 3–17, IEEE, 2014.
141. A. Mohaisen, O. Alrawi, M. Larson, and D. McPherson, "Towards a methodical evaluation of antivirus scans and labels," in *ISA workshop*, pp. 231–241, Springer, 2013.
142. Y. Li, J. Jang, X. Hu, and X. Ou, "Android malware clustering through malicious payload mining," in *RAID*, pp. 192–214, Springer, 2017.
143. P. Li, L. Liu, D. Gao, and M. K. Reiter, "On challenges in evaluating malware clustering," in *RAID*, pp. 238–255, Springer, 2010.
144. A. Kantchelian, M. C. Tschantz, S. Afroz, B. Miller, V. Shankar, R. Bachwani, A. D. Joseph, and J. D. Tygar, "Better malware ground truth: Techniques for weighting anti-virus vendor labels," in *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, pp. 45–56, 2015.

145. S. M. Mathews, "Explainable artificial intelligence applications in nlp, biomedical, and malware classification: A literature review," in *Intelligent Computing-Proceedings of the Computing Conference*, pp. 1269–1292, Springer, 2019.