Adversarial Attacks and Defenses in Malware Classification: A Survey

Ilja Moisejevs *iljamoisejevs@gmail.com*
Calypso AI
2955 Campus Drive #110 San Mateo, CA 94403

Abstract

As malware continues to grow more sophisticated and more plentiful – traditional signature and heuristics-based defenses no longer cut it. Instead, the industry has recently turned to using machine learning for malicious file detection. The challenge with this approach is that machine learning itself comes with vulnerabilities – and if left unattended presents a new attack surface for attackers to exploit.

In this paper we present a survey of research in the area of machine learning-based malware classifiers, the attacks they encounter, and the defensive measures available. We start by reviewing recent advances in malware classification, including the most important works using deep learning. We then discuss in detail the field of adversarial machine learning and conduct an exhaustive review of adversarial attacks and defenses in the field of malware classification.

Keywords: Machine Learning, Malware Classification, Adversarial Attacks, Evasion Attacks.

1. INTRODUCTION

In computer security, the detection of malware continues to be of paramount importance. Traditional approaches have relied heavily on signatures and heuristics to spot malicious files, however, there is increasing recognition in the community that these defenses are too easy to evade and don't scale very well. Instead there seems to be a tectonic shift happening towards using machine learning [1], [2], [3].

First attempts at applying machine learning to malware made use of traditional feature engineering [4], [5], [6], however, more recent works attemptedthe more advanced, end-to-end deep learning techniques [1], [7], [8], [9]. For example, [7] developed MalConv, a pioneering neural network solution able to ingest historically unprecedented byte sequences (over 2 million steps), [8] developed an artificial neural network focused on the header of portable executable (PE) files and [9] examined the role of spatial and temporal biases in deep learning based approaches.

Despite all the successes in applying machine learning to malware, however, it is a well-known fact that adaptive adversaries can evade detection through adversarial attacks. Attacks come in different shapes, with some leveraging model's gradients [12], [10], others using the model as an oracle [13], [14], [15], and others still relying on nothing more than random perturbations [1], [10].

In response to such attacks, a number of defense tactics have emerged. These include information hiding [12], [16], [17], regularization to prevent overfitting [19], feature selection [19], [20], [21], [22], [23], [24], [25], [26], [27], adversarial training [11], [14], [19], [20], [28], [29], the use of non-linear kernel functions (for algorithms such as Support Vector Machines) [30], distillation [20], [31], [32], and more [20], [32], [33].

The motivation for this paper is to review the various adversarial attack and defense methods as applied to malware classification. To the best of our knowledge, this is the first work to undertake such a survey.

This paper is organized as follows:in Section 2 we review machine learning as applied to malware classification, defining static and dynamic analysis and exploring the pros and cons of each;in Section 3 we review thetaxonomy of adversarial machine learning;and in Sections 4 and 5 we discuss evasion attacks and defenses in great detail.

2. MACHINE LEARNING IN MALWARE CLASSIFICATION

2.1 Machine Learning

Machine learning is a process by which computers learn complex representations from sample data through a process known as “training”. Typically, a training dataset is put together and fed to an algorithm to “learn” the patterns inside of it, with the hope that that same algorithm can later be applied to unknown data, or in other words it can “generalize”.

Machine learning can come in a variety of shapes. Perhaps the most basic classification of machine learning systems is into *supervised* and *unsupervised*. The former relies on the training dataset being labeled (eg a pool of malware labeled as “malicious” or “benign”) while the latter, generally considered a more advanced form of machine learning, relies on the algorithm understanding relationships between various parts of the dataset by itself. Other classifications of machine learning systems can be done by type of data (structured / unstructured / semi-structured), type of algorithm (linear, non-linear, composite), and even type of learning (traditional learning, one-shot learning, reinforcement learning).

Machine learning can also be applied to a variety of tasks. The three broad categories are *regression*, *classification* and *clustering*. Regression and classification are examples of supervised learning and differ in that the former outputs a continuous variable (such as a stock price), while the latter outputs a discrete variable (such as a “cats” or “dogs” class). Clustering is an example of unsupervised learning and occurs when the algorithm tried to automatically break the dataset into “clusters” with similar properties. Clustering is especially helpful in data analysis where the analyst may not initially know what the “shape” of the data is or what they might be looking for.

One of the most exciting recent developments in machine learning has been the emergence of *deep learning*. Although neural networks, the algorithms used in deep learning, have existed since the 1950s [51], it is only recently that they were able to achieve best in class results on various classification and regression tasks.

Perhaps the pivotal moment in the evolution of deep learning has been the 2012 ImageNet competition, where for the first time ever a team using deep learning came first, outperforming the next team by 41% [52]. The next year, most teams relied on deep learning [53], and just 5 years after the competition had to be shut down because the accuracy results achieved left little room for further improvement.

2.2 Static and Dynamic Analysis

Machine learning can be used to perform two kinds of analysis on malware – static and dynamic. *Static analysis* attempts to classify a file as benign or malicious without actually executing it [29]. Instead a static classifier will look at the file’s features – from individual strings, functions and API calls to metadata and file’s size – and try to make a decision if the file is suspicious. It is typically the first line of defense in any anti-virus or endpoint protection product, as its cheap and fast.

In contrast, *dynamic analysis* generates its classification based on the run-time behavior of the file [29], [34], [35], [36]. Here the file will typically be placed inside a sandbox and executed. The analyzer then monitors the behavior of the file looking for malicious indicators, such as a file

attempting to reach out to a remote server and start a download process. Dynamic analysis tends to be more expensive and hence is typically reserved for files already marked as “suspicious” by static analysis.

From an attacker’s perspective it is generally safe to assume that the malware, if it is to be evasive, needs to evade both static and dynamic classification. Hence over the years attackers have developed techniques to evade both [29], [34], [35], [36], [37], [38], [39], [40].

Evasion of the dynamic layer typically takes the form of recognizing the environment the malware is placed in (that it’s a sandbox) and suppressing the file’s malicious attributes. Evasion of the static layer, however, is much more reliant on understanding the classifier at hand and what features it’s looking for. In this survey, because we are primarily concerned with the security of the machine learning classifiers themselves, we will only focus on static evasion.

3. TAXONOMY OF ADVERSARIAL MACHINE LEARNING

3.1 Types of Adversarial Attacks

In order to understand the kinds of attacks an adversary might mount on a machine learning system it is important to understand the main steps involved in machine learning, of which there are two (Figure 1). First, the machine learning model ingests carefully prepared and labelled training data in order to learn its representation, like we discussed in Section 2. Second, the newly trained model can now be used to make predictions about new inputs hitting its API.

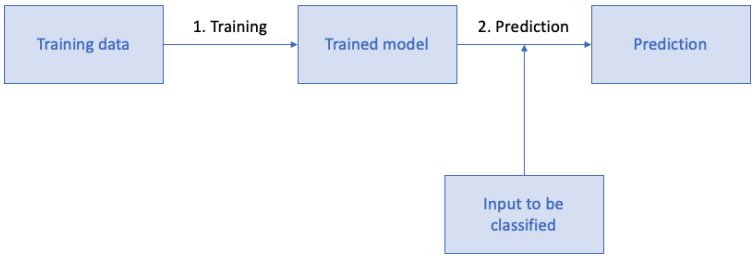

FIGURE 1: Simplified diagram depicting two steps of the machine learning process.

Each of the two steps described opens up possibilities for the adversary to interfere with model’s normal behavior, and hence also its output. At training, the attacker can inject bad data into the model’s training pool to make it learn an incorrect representation – an attack known as *data poisoning*. During poisoning, an attacker can do one of two things:

1. They can inject few carefully selected inputs with specific labels, which the model will learn to recognize and respond to (*backdoor poisoning*)
2. They can inject many random inputs to the point that the boundary that the model learns basically becomes useless (*availability poisoning*)

At test time, the attacker can make use of what seems to be a common vulnerability across virtually every type of machine learning system – susceptibility to *adversarial examples* [11]. Adversarial examples are carefully perturbed inputs that mislead the model into making an incorrect prediction, while seemingly only changing a few features (Figure 2) [54].

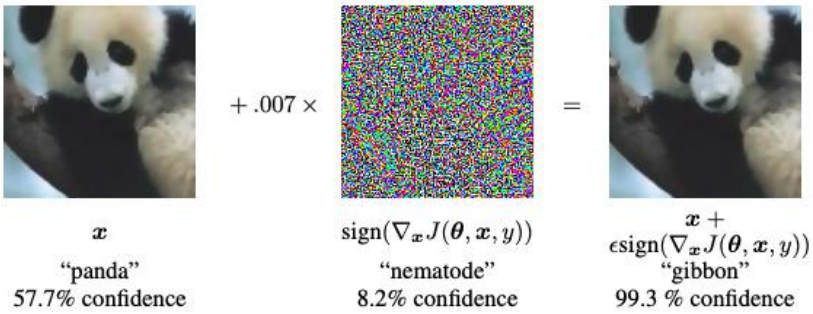

FIGURE 2: Perturbation applied to an image that although makes no difference to the human observer – still successfully throws of the classifier. Taken from the seminal paper by Goodfellow et al. [54].

Adversarial examples are also often simply referred to as *evasion attacks*. Due to their popularity in research literature and seeming much higher potential for damage, we will focus the rest of the survey on these evasion attacks.

3.2 Adversarial Capability

An adversary's capability refers to the amount of knowledge he or she has about the target classifier. On one end of the spectrum, the attacker might have full knowledge of the system. This is known as a *WhiteBox* attack. In a WhiteBox attack, the attacker knows what type of learning algorithm is used, the distribution of the training data, and what features the algorithm uses [43].

On the other end of the spectrum is a *BlackBox* attack. In a BlackBox attack, the adversary knows nothing about the internals of the system, however they are able to query the system freely [43]. If the attacker can get back both the class labels and the confidence score it is referred to as a *confidence-score* attack, otherwise, if the attacker can only obtain the label we call it a *hard-label* attack.

In the middle of these two extremes are *GrayBox* attacks. For example, the attacker might have access to the distribution of the training data, but not know specifics about the model used [42], [43].

3.3 Difference Between Malware and Images

Much of the research on evasion attacks has been done in the image domain, in the context of computer vision. We believe there are a number of reasons for that. Firstly, images by definition are visual and easy to understand for any person, hence it's an attractive research subject. Secondly, since the success of ImageNet competition in 2012, images and computer vision have taken center stage in the AI community; it's only natural then that once accuracy is solved the community turns its attention to security and robustness. But thirdly, and perhaps most importantly, images are significantly easier to perturb from an attacker's perspective than malware is.

There are two important differences between images and malware. Firstly, images present the attacker with a much bigger "playground" or perturbation space than malware does. In a 1000x1000 colored image the attacker has 3 million values to choose from for perturbation, and 255 dimensions to perturb each of those values in. They can tweak the color, opacity, lightning, saturation and many other properties until an evasive variant is found. Malware, on the other hand, is binary (it either has a particular API call or it does not), with far fewer options for the attacker to choose from [20], [42].

Secondly, images come with effectively unbounded perturbation space, while perturbing malware is a delicate process that needs to first and foremost preserve the file's functionality. There is no way to "break" an image by perturbing it, unless some of the pixel values are set outside the possible 0 to 255 range. Malware, on the other hand, can easily be made useless if even a single

function or API call in the file ceases to function correctly. Within the image realm, research typically relies on distance metrics to cap the possible perturbation space, however, even that has been brought into question from a practical security standpoint [50].

In the next two sections, we will review in more detail evasion attacks and defenses in the context of machine learning based malware classifiers.

4. EVASION ATTACKS ON MALWARE CLASSIFIERS

We will now discuss evasion attacks in more detail, split into WhiteBox and BlackBoxin-line with taxonomy presented in Section 3.

4.1 WhiteBox Attacks

Snrdic and Laskov [12] attempted attacks on a PDF malware classifier, PDFRate. They executed 2 attacks: mimicry (malicious file is made to look benign) and gradient-based (the data sample is manipulated in the reverse direction of the loss function) under 4 distinct knowledge scenarios (ranging from just knowing highlights to knowing everything about the objective classifier). Their attacks were successful and managed to get theaccuracyof the classifier down from 100% to 28-33%.

Biggio et al. [10] conducted a simple yet effective gradient-based attack on a PDF classifier, achieving successful evasion with as few as 5 to 10 modifications to the PDF file structure. It was found that /Linearized, /OpenAction, /Comment, /Root and /PageLayoutcould be added to PDF documents to mislead the classifier. The major limitations of their work is that they only perturbed the input in feature space and never actually checked if proposed perturbations could lead to real, functional malware in problem space.

Grosse et al. [20] mounted an evasion attack on a neural network model as described by McDaniel et al. [44]. The authors adapted an attack algorithm initially developed in computer vision [17] and successfully generated adversarial examples causing up to 85% misclassification rate of the target model.

Another work that attacked a neural network model using a gradient-based attack was that by Kolosnjaji et al. [1]. In order to prevent the malware from breaking, the authors restricted perturbations to just adding padding bytes, generate both randomly and using the model's gradients. As a result the authors managed to decrease the accuracy of MalConv [7] by 50%.

4.2 BlackBox Confidence Score Attacks

Xu et al. [19] proposed an attack method that doesn't require access to a model's gradients – however, does still require access to the predicted confidence scores. They implemented a system that makes use of genetic algorithms and a sandbox to verify the malware's functionality and used it to successfully evade PDFrate [17] and Hidost [30]. It is worth pointing out that this work is one of the few where the authors have put in the extra effort to ensure the perturbed samples remain functional malware (in other words, they attacked the model in "problem space", not "feature space" [28]).

4.3 BlackBox Hard-Label Attacks

Maiorca et al. [13] proposed a methodthey termed "reverse mimicry" to generate evasive variants of PDF malware. The intuition is to inject malicious code into a benign sample rather than doing the reverse, thus preserving most of the benign features used for classification. The authors were successful in their experiments, although their approach is overly manual.

Hu and Tan [14] proposed a generative adversarial network (GAN)-based approach, which takes original samples as inputs and outputs adversarial examples. Authors motivate their work by saying that intrinsic non-linear structure of neural networks enables them to generate more complex and flexible adversarial examples to fool the target model. The detector part of the GAN

is a surrogate model built from the original classifier in a BlackBox fashion (only assumes knowledge of the feature space). Experimental results showed that virtually all of the adversarial examples generated by MalGAN could successfully bypass the detection algorithm (true positive rate drops close to 0%).

Dang et.al. [15] proposed a new method for attacking malware classifiers with no knowledge of their internals and no confidence scores. They constructed a system with three components - the detector (BlackBox API for the malware classifier), the tester (confirms maliciousness), and the morpher (perturbs the file). They then proposed a scoring mechanism that assigns real-value scores to samples based on the binary outputs obtained from the tester and detector. The intuition is to measure the number of morphing steps required to change the detector and tester's decisions, and derive the score based on these values.

Finally, they empirically demonstrated that their approach can find evasive variants 100% of the time, outperforming random perturbation by as much as 80x in terms of execution cost. The approach is interesting because unlike other most BlackBox approaches that require a substitute model - this approach is direct and makes no assumptions about the transferability property of adversarial attacks [55].

Rosenberg et al. [46] not only developed a system to launch a BlackBox attack on malware classifiers, but also built a generalized and end-to-end framework to do it. The effectiveness of their framework was tested against 14 different models achieving successful evasion in 70-90% of the time, depending on the model. Apart from the original training dataset, they also proved the effectiveness of the proposed framework on previously unseen malware, including WannaCry.

Yang. et al. [47] took a unique approach to perturbing malware to generate evasive variants. First, instead of taking vanilla static or dynamic features the authors extracted features from Android benign and malicious samples based on the RTLD (Resource, Temporal, Locale, and Dependency) model. Then they proposed 2 types of attacks. The *evolution attack* looks at malware family trees, finds the most popular mutations that the malware authors have used, and tries to re-apply them to malware samples that don't yet have them. The *confusion attack* tries to replace features that are unique to malware samples with features that can be found in both malware and benign samples, but that lead to the same functionality. The authors investigated the success of the two attacks each individually and together (under the "MRV" umbrella) and found that they were able to evade VirusTotal 80% of the time, Drebin 60% of the time, and AppContext 20% of the time.

Hu and Tan [48] argued that malware detection will move in the direction of sequential classifiers based on RNNs. Hence they mounted an attack on 6 different RNN classifiers. To be able to inject whole API sequences they developed their own unique attack algorithm that relies on a generative RNN to perturb the files. Eventually they successfully reduced the accuracy of 6 RNN models from 90% to low single digits, even when the adversarial examples were prepared on a substitute RNN with a different architecture. Thus they also reaffirmed adversarial example transferability in the malware space [55].

Finally, Anderson et.al. [29] wanted to perform an attack as closely grounded in reality as possible. This meant both a)only relying on hard label output of the classifier and b) holding no assumptions about the features space used by the model. They built a reinforcement learning system that stochastically modified PE binaries with features sampled from a benign feature list, capped at 10 mutations per file. Depending on the dataset used (authors tried 4), they managed to generate evasive malware in 10-24% of cases. Because of the extremely limited threat model the results appear modest compared to other papers; what's more their results seem to have done no better than random perturbation would have, as the authors themselves point out in their paper. This highlights just how challenging adversarial attacks on malware classifiers under realistic, 0-knowledge threat models are.

Clearly, there is no shortage of methods to evade malware classifiers – be itBlackBox or WhiteBox. It is thus paramount that such classifiers are hardened and made more robust, which is exactly what we review in the next section.

5. DEFENSES FOR MALWARE CLASSIFIERS

In this section we will review a number of defensive strategies that can help make the classifiers more robust to evasion attacks.

5.1 Information Hiding

Information hiding is perhaps the most obvious place to start when thinking about classifier defenses, because many attacks rely on access to model's gradients or at least confidence scores. Such defenses were discussed and attempted in previous literature [16].

A somewhat related method is injecting randomness into the classifier system through for example multiple classifier ensembling [22]. This makes it more difficult for the attacker to reverse engineer the classifier and mathematically prepare adversarial examples.

The biggest downside of this approach is that it doesn't fix the underlying problem (classifier making decisions based on non-robust features) but simply hides behind a curtain of obscurity. The security community in general is rightly skeptical of such *security through obscurity* approach and it indeed can backfire [12], [17], [19].

5.2 Adapting to Evasive Variants

To be able to develop an evasive sample, the attackers would first need to test multiple samples on the classifier. Therefore, in an online scenario, the classifier can learn and adapt to the attempted alternates by storing and training again on every sample recognized by the API. It should be noted, however, that retraining is usually cost ineffective and potentially makes the classifier vulnerable to alternate evasion strategies such as poisoning [18], [19].

In [18], Chinavle et al. the authors presented an approach to retrain the classifier using pseudo labels as soon as the evasive variants were first detected from an ensemble model. This type of approach was particularly effective in solving a spam detection task [18].

5.3 Preventing Overfitting

Xu et al. [19] argue that the problem of evasion is really a problem of overfitting. In this case, common machine learning strategies generally used to combat overfitting can be implemented. For example, methods such as training on a significantly larger dataset or using strategies such as model averaging to reduce the variance may be employed. It is unlikely, however, that these strategies would confer true robustness – it is impossible to collect a complete data set that would include all future malware variants, and hence some degree of overfitting should always be expected.

5.4 Feature Selection

Xu et al. [19] demonstrated that even features used by popular classifiers such as PDFRate and Hidost are non-robust (they are meaningless in the context of malware). Instead, in order to build better classifiers the authors proposed to use deeper features that resist evasion. Such features could be crafted from higher-level semantic analysis of the input file and would target properties of the file difficult to change without disrupting the malicious behavior.

Grosse et al. [20] considered manual feature selection, restricting possible features only to AnroidManifest.xml and discarding those with the highest cardinality. Unfortunately, this turned out to make the neural network less secure. Other works have similarly found that feature selection can in fact make the classifier less robust, not more [21], [22], [23], [24], [25].

Nevertheless, Zhang et al. [26] took a different approach to feature selection that proved to be effective. The authors first added a regularization term to the loss function that represents robustness to adversarial attacks. They then conducted forward selection and backward elimination of features, until only robust features were left. It is noteworthy that this approach, in contrast to previous ones [21], [22], [23], [24], [25], didn't require any assumptions about the model's feature space.

Finally, in [27] the authors successfully combinedfeature selection and ensembling to build arobust classifier for Android malware. Their system enjoyed true positive classification rates of greater than 80% under heavy attacks, compared to 20-60% for an undefended model. Unattacked, their system performed on par with undefended classifiers in terms of accuracy.

5.5 Distillation

Distillation, first proposed by Hinton et al. [31], refers to the transfer of knowledge from a large, teacher neural network to a smaller, student network. Papernot et al. [32] suggested distillation could be used to defend classifiers against evasion attacks.

Grosse et al. [20] found that malware classifiers trained via distillation did show enhanced robustness against attacks. However, improvements in accuracy were significantly lower than what was originally observed in the image domain (misclassification rates decreased only to 40% vs to 5% in images) [32].

5.6 Adversarial Training

Adversarial training was originally proposed by Szegedy at al. [11] and involves the following three steps [20]:

1. Train a classifier on the original, benign dataset
2. Craft adversarial examples for the classifier trained in (1)
3. Iterate additional training epochs of the original classifier, this time including correctly labeled samples from (2)

Grosse et. al. [20] showed that re-training the classifier on the newly created training data improves its effectiveness against evasion attacks. However, the gains were minimal, topping out at around a 6% improvement in misclassification rate.

Hu and Tan [14] explored using adversarial training against GAN-based attacks.While the re-trained model did show improvement, once its new weights became publicly known, the GAN was able to always generate new adversarial examples, with 100% success rate.

Tong et al. [28] showed that it took ten rounds of adversarial re-training to finally confer robustness against the attack method by Xu et al. [19]. They also, however, drew a distinction between two types of attacks - those constructed in problem space (actual code is modified and the file is then checked for maliciousness through a sandbox) vs those constructed in feature space (feature vectors are modified using gradients, and then translated back into problem space, if possible). They pointed out that adversarial retraining with feature space-based attacks doesn't always confer resistance to problem space-based attacks, and hence robustness against [19] doesn't necessarily imply "actual" robustness.

In [29], the authors successfully improved classification accuracy by 33% by using adversarial training. However, the authors noted that the library used for binary manipulation seems to be leaving unique fingerprints in the code, and suspected that adversarial training simply learnt to detect those fingerprints, rather than to generalize to detect evasive malware samples.

Finally, researchers from MIT re-trained a classifier using adversarial samples from four different attacks [49]. The authors managed to increase the robustness of the model, including against

attacks from [20], reducing fooling rate from near 100% on undefended model to 35% on the defended one. Other works have also found success with adversarial re-training [47].

5.7 Non-linearity

In [30] the authors found that using an RBF kernel reduced the effectiveness of evasion attacks on SVMs from 50% down to practically 0%. The same paper found decision trees to be infinitely vulnerable to evasion, to the point where the authors strongly recommended not to use them in security-conscious applications. This work is in-line with one of the most important research papers in the space, written by Goodfellow et al. [54], where the authors postulate that the entire reason adversarial examples exist is excessive linearity of modern machine learning models.

5.8 Micro-detectors

The final defensive approach seen in the literature utilizes multiple smaller machine learning models (termed "micro-detectors") working together to confer classifier robustness [33]. Each of the micro-detectors is trained on a sub-portion of data and learns to recognize a particular malware type or family. A key benefit of such a design is its adaptability. If one model becomes obsolete or needs to be re-trained, it can be done so quickly and inexpensively, without having to affect other models.

6. CONCLUSION

This survey looked at machine learning in the malware classification domain, and more specifically how such classifiers can be attacked by and protected from adversaries wishing to evade detection. We have provided a background on machine learning and adversarial attacks and dived deep into a particular class of attacks - evasion attacks. Finally, we have reviewed the available research that can help make malware classifiers more resistant to such attacks.

Machine learning is destined to take central place in the future of our cybersecurity systems, and we believe it is absolutely paramount that we learn to build classifiers that are not only effective, but also robust. We would hate a future where machine learning itself becomes the weakest link in the security chain– and hope this survey inspires new research that will prevent that from happening.

7. REFERENCES

[1] 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 *Proc. 2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 533-7.

[2] "Internet security threat report (ISTR 22)." Internet: https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf, Apr. 2017 [Jul. 24, 2019].

[3] "Machine learning for malware detection." Internet: https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf, nd [Jul. 24, 2019].

[4] T. Abou-Assaleh, N. Cercone, V. Keselj and R. Sweidan. "N-gram-based detection of new malicious code," in *Proc. 28th Annual International Computer Software and Applications Conference 2004 (COMPSAC 2004),* 2004, pp. 41-2.

[5] K. Rieck, T. Holz, C. Willems, P. Düssel and P. Laskov. "Learning and Classification of Malware Behavior, " in *Proc. DIMVA'08: Detection of Intrusions and Malware, and Vulnerability Assessment, 2008, pp. 108-25.*

[6] G. Yan, N. Brown and D. Kong. “Exploring Discriminatory Features for Automated Malware Classification,” in *Proc. DIMVA’13 - 10th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2013, pp. 41-61.

[7] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro and C. Nicholas. “Malware Detection by Eating a Whole EXE,” in *Proc. 32nd AAAI Conference on Artificial Intelligence,* 2017.

[8] E. Raff, J. Sylvester and C. Nicholas. “Learning the PE Header, Malware Detection with Minimal Domain Knowledge,” in *Proc. AISec’17 - 10th ACM Workshop on Artificial Intelligence and Security,* 2017, pp. 121-32.

[9] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder and L. Cavallaro. “TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time,” in *Proc. SEC'19 - 28th USENIX Conference on Security Symposium,* 2018, pp. 729-46.

[10] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto and F. Roli. "Evasion attacks against machine learning at test time," in *Proc. ECML/PKDD'13 – 2013 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III,* 2013, pp. 387-402.

[11] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus. (2013, Dec.). “Intriguing properties of neural networks.” *arXiv Preprint - arXiv:1312.6199*. [Online]. Available: https://arxiv.org/abs/1312.6199 [Jul. 24, 2019].

[12] N. Šrndić and P. Laskov. “Practical Evasion of a Learning-Based Classifier: A Case Study,” in *Proc. 2014 IEEE Symposium on Security and Privacy*, 2014, pp. 197-211.

[13] D. Maiorca, I. Corona and G. Giacinto. “Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious PDF files detection,” in *ASIACCS’13 - 8th ACM SIGSAC symposium on Information, computer and communications security,* 2013, pp. 119-30.

[14] W. Hu and Y. Tan. (2017, Feb.). “Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN.” *arXiv Preprint – ArXiv:1702.05983*. [On-line]. Available: https://arxiv.org/abs/ 1702.05983 [Jul. 24, 2019].

[15] H. Dang, Y. Huang and E. Chang. “Evading Classifiers by Morphing in the Dark,” in *Proc. CCS’17 – 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 119-33.

[16] M. Barreno, B. Nelson, R. Sears, A.D. Joseph and J.D. Tygar. “Can machine learning be secure?,” in *Proc. ASIACCS'06 - 2006 ACM Symposium on Information, computer and communications security*, 2006, pp. 16-25.

[17] C. Smutz and A. Stavrou. “Malicious PDF detection using metadata and structural features,” in *Proc. ACSAC’12 - 28th Annual Computer Security Applications Conference*, 2012, pp. 239-48.

[18] D. Chinavle, P. Kolari, T. Oates and T. Finin. “Ensembles in adversarial classification for spam,” in *Proc. CIKM’09 - 18th ACM Conference on Information and Knowledge Management,* 2009, pp. 2015-8.

[19] W. Xu, Y. Qi and D. Evans. “Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers,” in *Proc. NDSS’16 - Network and Distributed System Security Symposium*, 2016.

[20] K. Grosse, N. Papernot, P. Manoharan, M. Backes and P.D. McDaniel. (2016, Jun.). “Adversarial Perturbations Against Deep Neural Networks for Malware Classification.” *arXiv Preprint – ArXiv:1606.04435*. [Online]. Available: https://arxiv.org/abs/1606.04435 [Jul. 24, 2019].

[21] B. Biggio, G. Fumera and F. Roli. “Security Evaluation of Pattern Classifiers under Attack.” *IEEE Transactions on Knowledge and Data Engineering,* vol. 26(4), pp. 984-96, Apr. 2014.

[22] B. Biggio, G. Fumera and F. Roli. “Multiple classifier systems for robust classifier design in adversarial environments.” *International Journal of Machine Learning and Cybernetics,* vol. 1(1-4), pp. 27-41*,* 2010.

[23] B. Biggio, G. Fumera and F. Roli. "Evade hard multiple classifier systems." in *Applications of Supervised and Unsupervised Ensemble Methods*, vol. 245. O. Okun and G. Valentini, Eds. Berlin: Springer, 2009, pp. 15-38.

[24] B. Li and Y. Vorobeychik. “Feature Cross-Substitution in Adversarial Classification,” in *Proc. NIPS'14 - 27th International Conference on Neural Information Processing Systems - Volume 2,* 2014, pp. 2087-95.

[25] F. Wang, W. Liu and S. Chawla. “On Sparse Feature Attacks in Adversarial Learning,” in *Proc. 2014 IEEE International Conference on Data Mining (ICDM)*, 2014, pp. 1013-8.

[26] F. Zhang, P.P.K. Chan, B. Biggio, D.S. Yeung and F. Roli. “Adversarial Feature Selection Against Evasion Attacks.” *IEEE Transactions on Cybernetics,* vol. 46(3), pp. 766-77, Mar. 2016.

[27] L. Chen, S. Hou and Y. Ye. “SecureDroid: Enhancing Security of Machine Learning-based Detection against Adversarial Android Malware Attacks,” in *Proc. ACSAC 2017 - 33rd Annual Computer Security Applications Conference*, 2017, pp. 362-72.

[28] L. Tong, B. Li, C. Hajaj, C Xiao and Y. Vorobeychik. (2017, Nov.). "Hardening Classifiers Against Evasion: the Good, the Bad, and the Ugly." *arXiv Preprint – ArXiv:1708.08327v2*. [Online]. Available: https://arxiv.org/abs/1708.08327v2 [Jul. 26, 2019].

[29] H.S. Anderson, A. Kharkar, B. Filar, D. Evans and P. Roth. (2018, Jan.). “Learning to Evade Static PE Machine Learning Malware Models via Reinforcement Learning.” *arXiv Preprint – ArXiv:1801.08917*. [Online]. Available: https://arxiv.org/abs/1801.08917 [Jul. 26, 2019].

[30] N. Šrndić and P. Laskov. “Detection of Malicious PDF Files Based on Hierarchical Document Structure,” in *Proc. NDSS 2013 - Network and Distributed System Security Symposium*, 2013.

[31] G.E. Hinton, O. Vinyals and J. Dean. (2015, Mar.). “Distilling the Knowledge in a Neural Network.” *arXiv Preprint – ArXiv:1503.02531*. [Online]. Available: https://arxiv.org/abs/1503.02531 [Jul. 26, 2019].

[32] N. Papernot, P.D. McDaniel, X. Wu, S. Jha and A. Swami. “Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks,” in *Proc. 2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 582-97.

[33] S. Saad, W. Briguglio and H. Elmiligi. "The Curious Case of Machine Learning in Malware Detection,” in *Proc. ICISSP 2019 - 5th International Conference on Information Systems Security and Privacy,* 2019.

[34] B. Athiwaratkun and J.W. Stokes. “Malware classification with LSTM and GRU language models and a character-level CNN,” in *Proc. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2482-6.

[35] G.E. Dahl, J.W. Stokes, L. Deng and D. Yu. “Large-scale malware classification using random projections and neural networks,” in *Proc. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 3422-6.

[36] R. Pascanu, J.W. Stokes, H. Sanossian, M. Marinescu and A. Thomas. “Malware classification with recurrent networks,” in *Proc. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 1916-20.

[37] T. Raffetseder, C. Kruegel and E. Kirda. “Detecting System Emulators,” in *Proc. ISC’07 - 10th international conference on Information Security*, 2007, pp. 1-18.

[38] T. Garfinkel, K. Adams, A. Warfield and J. Franklin. “Compatibility Is Not Transparency: VMM Detection Myths and Realities,” in *Proc. HOTOS'07 - 11th USENIX workshop on Hot topics in Operating Systems,* 2007.

[39] M. Carpenter, T. Liston and E. Skoudis. "Hiding Virtualization from Attackers and Malware." *IEEE Security and Privacy,* vol. 5(3), pp. 62–5, May-Jun. 2007.

[40] C. Rossow, C.J. Dietrich, C. Grier, C. Kreibich, V. Paxson, N. Pohlmann, H. Bos and M. van Steen. “Prudent Practices for Designing Malware Experiments: Status Quo and Outlook,” in *Proc.* SP’12 - *2012 IEEE Symposium on Security and Privacy*, 2012, pp. 65-79.

[41] N. Papernot, P. McDaniel, A. Sinha and M. Wellman. (2016, Nov.).“Towards the Science of Security and Privacy in Machine Learning.” *arXiv Preprint – ArXiv:1611.03814*. [Online]. Available: https://arxiv.org/abs/1611.03814 [Jul. 26, 2019].

[42] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay and D. Mukhopadhyay. (2018, Sep.). “Adversarial Attacks and Defences: A Survey.” *arXiv Preprint - arXiv:1810.00069*. [On-line]. Available: https://arxiv.org/abs/1810.00069 [Jul. 24, 2019].

[43] I. Moisejevs. “Will My Machine Learning Be Attacked?.” Internet: https://towardsdatascience.com/will-my-machine-learning-be-attacked-6295707625d8, Jul. 14, 2019 [Jul. 24, 2019].

[44] P. McDaniel, N. Papernot and Z.B. Celik. “Machine Learning in Adversarial Settings.” *IEEE Security & Privacy*, vol. 14(3), pp. 68-72, May-Jun. 2016.

[45] J.S. Cross and M.A. Munson. "Deep pdf parsing to extract features for detecting embedded malware," in *Technical report: SAND2011-7982.* California: Sandia National Laboratories, Sept. 2011.

[46] I. Rosenberg, A. Shabtai, L. Rokach and Y. Elovici. “Generic Black-Box End-to-End Attack Against State of the Art API Call Based Malware Classifiers,” in *Proc. RAID 2018: Research in Attacks, Intrusions, and Defenses,* 2018, pp. 490-510.

[47] W. Yang, D. Kong, T. Xie and C.A. Gunter. “Malware Detection in Adversarial Settings: Exploiting Feature Evolutions and Confusions in Android Apps,” in *Proc. ACSAC 2017 - 33rd Annual Computer Security Applications Conference,* 2017, pp. 288-302.

[48] W. Hu and Y. Tan. (2017, May.). “Black-Box Attacks against RNN based Malware Detection Algorithms.” *arXiv Preprint - arXiv:1705.08131*. [On-line]. Available: https://arxiv.org/abs/1705.08131 [Jul. 24, 2019].

[49] A. Al-Dubjaili, A. Huang, E. Hemberg and U. O'Reilly. “Adversarial Deep Learning for Robust Detection of Binary Encoded Malware,” in *Proc. 2018 IEEE Symposium on Security and Privacy Workshops (SPW)*, 2018, pp. 76-82.

[50] J. Gilmer, R.P. Adams, I. Goodfellow, D. Andersen and G.E. Dahl. (2018, Jul.). “Motivating the Rules of the Game for Adversarial Example Research.” *arXiv Preprint - arXiv:1807.06732*. [On-line]. Available: https://arxiv.org/abs/1807.06732 [Jul. 24, 2019].

[51] “Neural Networks – History: The 1940's to the 1970's.” Internet: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html, nd [Jul. 24, 2019].

[52] A. Krizhevky, I. Sutskever and G.E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” *Communications of the ACM,* vol. 60(6)*,* pp. 84-90, Jun. 2017.

[53] “Imagenet Large Scale Visual Recognition Challenge 2013 (ILSVRC2013).” Internet: http://www.image-net.org/challenges/LSVRC/2013/index.php, 2013 [Jul. 24, 2019].

[54] I.J. Goodfellow, J. Shlens and C. Szegedy. “Explaining and Harnessing Adversarial Examples,” in *Proc. 3rd International Conference on Learning Representations (ICLR 2015),* 2015.

[55] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh and P. McDaniel. (2017, May.). “The Space of Transferable Adversarial Examples.” *arXiv Preprint - ArXiv, abs/1704.03453*. [On-line]. Available: https://arxiv.org/abs/1704.03453 [Jul. 24, 2019].