IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 13, 2020, accepted January 25, 2020, date of publication February 18, 2020, date of current version February 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2974752

Adversarial Machine Learning Applied to Intrusion and Malware Scenarios: A Systematic Review

NUNO MARTINS[1], JOSÉ MAGALHÃES CRUZ[1], TIAGO CRUZ[2], AND PEDRO HENRIQUES ABREU[2]

[1]Faculty of Engineering, University of Porto, 4200-465 Porto, Portugal
[2]Faculty of Sciences and Technology, University of Coimbra, 3030-290 Coimbra, Portugal

Corresponding author: Nuno Martins (up201405079@fe.up.pt)

ABSTRACT Cyber-security is the practice of protecting computing systems and networks from digital attacks, which are a rising concern in the Information Age. With the growing pace at which new attacks are developed, conventional signature based attack detection methods are often not enough, and machine learning poses as a potential solution. Adversarial machine learning is a research area that examines both the generation and detection of adversarial examples, which are inputs specially crafted to deceive classifiers, and has been extensively studied specifically in the area of image recognition, where minor modifications are performed on images that cause a classifier to produce incorrect predictions. However, in other fields, such as intrusion and malware detection, the exploration of such methods is still growing. The aim of this survey is to explore works that apply adversarial machine learning concepts to intrusion and malware detection scenarios. We concluded that a wide variety of attacks were tested and proven effective in malware and intrusion detection, although their practicality was not tested in intrusion scenarios. Adversarial defenses were substantially less explored, although their effectiveness was also proven at resisting adversarial attacks. We also concluded that, contrarily to malware scenarios, the variety of datasets in intrusion scenarios is still very small, with the most used dataset being greatly outdated.

INDEX TERMS Cybersecurity, adversarial machine learning, intrusion detection, malware detection.

I. INTRODUCTION

Protecting computer systems and networks from digital attacks is a rising concern in the recent years [72]. Although most systems today are built with improved security characteristics, there is still a vast amount of vulnerabilities, mainly due to outdated software, insecure protocols/systems and human error. Cyber-attacks can target any infrastructure, from cloud systems to Internet of things (IoT) devices, in the most various forms [72].

Intrusion detection systems (IDS) typically use signatures or system misuse patterns to identify cyber-attacks, but with the growth of the diversity of attacks in recent years, machine learning approaches are being widely employed [64].

Malware detection follows a similar approach, classically using signature based and reverse engineering techniques to detect malicious code in files, such as trojan horses, spyware and rootkits [70].

A downside of using machine learning techniques to perform classifications is the possibility of adversaries that try to circumvent the classifiers. The field that studies these types of attacks is called ''adversarial machine learning'', and has been extensively explored in some areas such as image classification and spam detection; its exploration is still small in other areas though, such as intrusion detection [46]. Basically, adversarial examples are inputs to a classifier specifically crafted to deceive the model, causing misclassification.

Models are many times trained with assumptions in mind for convenience or ease of computation, such as feature independence and linear separability of the data, but these types of assumptions can often open possibilities for adversarial attacks [23].

The aim of this survey is to explore applications of adversarial machine learning to cyber-security scenarios, specifically intrusion and malware detection, since with the growth

The associate editor coordinating the review of this manuscript and approving it for publication was Aniello Castiglione.

of machine learning applied to this area, adversaries can attempt to circumvent detection systems. Critical systems, which must be highly reliable, often deal with sensitive data. Therefore, achieving the maximum amount of security is a top priority, including resilience to adversarial attacks which are proven effective in the various studies explored here.

To elaborate this survey, a search was performed on multiple science databases, such as **IEEE Xplore**, **Springer**, **arXiv**, **ScienceDirect** and **Research Gate**, as well as regular search engines such as **Google**, using keywords *"Intrusion Detection"*, *"Malware Detection"* and *"Adversarial Machine Learning"*. We found in total twenty papers on these topics, with nine being related to intrusion detection, and eleven being related to malware scenarios. We also found other papers that apply adversarial concepts to intrusion and malware scenarios, but we specifically focus on adversarial machine learning. Several adversarial machine learning surveys and books were used to find multiple adversarial attack and defense strategies [12], [15], [18], [21], [23], [24], [26], [50]. After exploring these studies, we concluded that adversarial attack techniques were proven effective in both malware and intrusion scenarios, with a high diversity of techniques being tested. We also concluded that adversarial defense techniques are still not thoroughly explored, with few studies testing their application, and that the variety of datasets on intrusion scenarios is still very small.

The article is organized as follows: in section 2, some background concepts on machine learning and adversarial machine learning are presented; in section 3 we explore state-of-the-art adversarial attack strategies; in section 4, we take a look at defensive strategies, which aim to detect adversarial attacks; in section 5 we explore works in the field of intrusion and malware detection that apply adversarial machine learning concepts; in section 6, we present our conclusions, and propose new directions for future research.

II. BACKGROUND KNOWLEDGE

In this section we explore the fundamentals of machine learning and adversarial machine learning, along with several attack and defense strategies that have been applied both to malware/intrusion detection and computer vision security scenarios.

A. MACHINE LEARNING ALGORITHMS

Machine learning, a narrower field of artificial intelligence, comprises the study of algorithms that computer systems use to mainly perform classification tasks without external instructions, by using trained models.

Pedro Domingos [22] distinguishes five tribes of algorithms:

- **Symbolists** - These algorithms essentially focus on inverse deduction. Instead of starting with the premise and looking for conclusions, symbolists start with some premises and conclusions, and try to fill the gaps in between. Among the symbolists, the most notorious are decision trees and random forests.

  A decision tree is a tree-structure resembling a flowchart where every node represents a test to an attribute, each branch represents the possible outcomes of that test, and the leaves represent the class labels. The paths from root to leaves represent the decision rules.

  A random forest is an ensemble learning method that builds a large group of independent decision trees, and outputs the mode of the label predictions of all the trees. This method has higher computing costs, but tends to reduce overfitting of the data, which happens when a model adapts too strictly to a particular set of data, having poor capabilities for generalization.
- **Connectionist** - This tribe focuses on reverse engineering the human brain. This approach involves creating artificial neurons and connecting them in neural networks. All connectionist algorithms revolve around the usage of neural networks.

  Neural networks are a learning framework that aims to mimic animals brains. The main concept is that a group of neurons are organized among layers, with each layer connected to surrounding layers. The training process consists of adjusting the weights of the connections among layers, until the output of the final layer represents the correct labels. The process of adjusting weights is typically back-propagation, a method that, given an objective function in the last layer, adjusts the weights of the entire network given the gradient of this function in respect to every weight.

  Autoencoders are a type of neural network that aims to reconstruct data from the input layer into the output layer with a minimal amount of distortion [65]. Since the objective function is calculated with a comparison between the input and output, autoencoders are an unsupervised learning algorithm, since they do not require class labels to be trained.
- **Evolutionaries** - The evolutionaries focus on applying the idea of genomes and DNA in the processing of data. The survival and offspring of units is essentially considered the performance data.

  Genetic algorithms are a set of evolutionary algorithms which take inspiration from genetic evolution observed in living beings. The algorithms typically start with a set of individuals, and through processes of mutation and reproduction between multiple individuals, new populations are generated. With the progress of the algorithms, the most fit individuals have a higher chance of reproducing, leading to a more fitting population on each iteration.
- **Bayesians** - Focuses on applying probabilistic inference, such as the Bayes theorem. Applies 'a priori' thinking, with the belief that different outcomes have different probabilities.

  Naive Bayes is a machine learning algorithm that consists of applying the Bayes theorem in order to find a distribution of conditional probabilities among class labels, with the assumption of independence

between features. It has the advantage of being highly scalable.

- **Analogizers** - Focuses of the idea that close elements are more strongly related, essentially matching related pieces of data. Among the analogizers, the most popular ones are k-nearest-neighbours (KNN) and support vector machines (SVM).
  KNN is a classification algorithm that uses a distance function (e.g. Euclidean distance) in order to find the k closest elements in the feature space. The final label will depend on the mode of the distribution of classes in those neighbours.
  SVM is a binary classification algorithm which creates an hyper plane that separates the data. On each side of the hyper plane are the classes used for classification, and the objective is to maximize the gap perpendicular to the plane, allowing better generalization. Through the usage of kernel functions (e.g. gaussian), it is possible to generate hyper planes in higher dimensions, when the classes are not linearly separable.

B. FUNDAMENTALS OF ADVERSARIAL MACHINE LEARNING

Adversarial machine learning is the field that studies a class of attacks which aim to deteriorate the performance of classifiers on specific tasks. Adversarial attacks can be mainly classified as **poisoning** attacks, if the attacker influences the training data or its labels to cause the model to under-perform during deployment, or **evasion** attacks, if the attacker manipulates the data during deployment to deceive previously trained classifiers [67]. This paper explores evasion attacks, because: they are the most researched types of attacks; they are the most practical types of attacks, since they are performed during the deployment phase; they are the most used types of attack on intrusion and malware scenarios.

Huang *et al.* [67] proposed a formal taxonomy to model an adversary, according to the following aspects:

- **Influence** - refers to the capabilities the attacker has over its interaction with the target. It can either be a **causative** attack if the attacker can tamper the training data of the classifier, or **exploratory** if the attacker can not influence the training process, but can use other techniques to probe for useful information;
- **Security violation** - refers to the type of violation of the attack. These can be **integrity** attacks when the aim is for the attack to be classified as normal (false negative), **availability** attacks, when the aim is to cause misclassifications of any type (false negative or false positive), rendering the model useless, or **privacy** attacks, when the aim is to obtain information from the learner;
- **Specificity** - refers to the broadness of misclassification. These can be **targeted** attacks when the intent is for the attacks to be misclassified into a certain class/group of classes, or **indiscriminate** when there is no specific class to be targeted, and the objective is only to cause a misclassification.

III. ADVERSARIAL ATTACK STRATEGIES

In this section, we explore adversarial attack techniques that have been applied to intrusion and malware attack scenarios. Several techniques have been proposed, with a trade-off on performance, complexity, computational efficiency [46] and application scenario (black-box and white-box). In white-box attacks, the attacker has knowledge over the training data, model parameters, and other useful information about the classifier. In black-box scenarios, the attacker has little to no knowledge of the classifier, and thus is severely limited.

Gradient based methods typically introduce perturbations optimized for certain distance metrics between the original and perturbed samples. The three mainly used distance metrics in literature are:

- L_{inf} - minimize the maximum amount of perturbation introduced to any feature;
- L_0 - minimize the amount of features perturbed;
- L_2 - minimize the Euclidean distance between original and adversarial samples.

Szegedy *et al.* proposed one of the first gradient methods to generate adversarial examples applied to the imaging field, using **box constrained limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS)** optimization [63]. Given an input image, it searches for a different image that is similar to the first, under L_2 distance, that is mislabeled by the classifier. By adding noise to the original image, the problem is represented as an optimization problem, where the objective is to minimize the perturbations r added to the original image under L_2 distance:

$$min\ ||r||^2 \quad subject\ to:\ f(x+r) = l$$

Here, x is the original image, r is the perturbation, f is the loss function of the classifier and l the incorrect prediction/label. Since this is a non-trivial problem, the authors approximate it by using L-BFGS optimization algorithm to solve it (the box constraint is needed to limit the possible values of the variables, which are pixels in image scenarios).

Although this method is effective at producing adversarial examples, it is not very practical, since it uses a computationally expensive algorithm to search for an optimal solution.

Goodfellow *et al.* proposed a simple and fast gradient based method to generate adversarial examples called **Fast Gradient Sign method (FGSM)**, with the aim of minimizing the maximum amount of perturbation added to any pixel (L_{inf} distance metric) of the image to cause misclassification [58]. The rationale of this technique is to compute the gradient of the loss function with respect to the input (e.g. using backpropagation), and add perturbations to each feature (each pixel in the case of images) with the sign of the gradient by a flat amount. The formula for the attack is the following:

$$\eta = \epsilon * sign(\nabla x\ J(x, y))$$

Here, η is the perturbed sample, ϵ is a hyper-parameter controlling the amount of perturbation added to each feature (which is a flat value), J is the cost function, x is the original input and y the target label (figure 1).

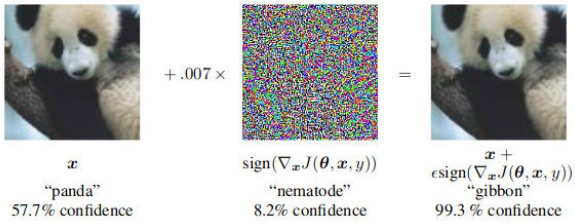


FIGURE 1. FGSM - By using a ϵ of 0.007, which only allows at most 1 bit to be changed in a 8 bit representation of each channel of a pixel, the perturbation can lead to a misclassification (in [58]).

Although this method is less effective than other state of the art techniques for generating adversarial attacks, while also introducing more perturbations than others, it is one of the most efficient at computing time, allowing fast generation of adversarial examples.

Papernot *et al.* proposed a new adversarial sample generation technique called **Jacobian based Saliency Map Attack (JSMA)** that, unlike the previous method, uses feature selection, with the aim of minimizing the number of features modified (L_0 distance metric) while causing misclassification [55]. This method revolves around the computation of saliency maps for an input sample, which contain the saliency values for each feature. These values indicate how much the modification of each feature will affect the classification process. The features are then selected in decreasing order of saliency value, and each is perturbed according by the value θ. The process finishes when misclassification occurs, or a threshold number of modified features is reached.

This is a more computationally intensive method compared to FGSM, since it requires the computation of the saliency values, but drastically reduces the amount of features perturbed, allowing the generation of adversarial examples seemingly closer to the original sample (figure 2).

Moosavi-Dezfooli *et al.* proposed an untargeted adversarial sample generation technique called **Deepfool**, with the aim of minimizing the euclidean distance between perturbed samples and original samples (L_2 distance metric) [59]. The generation of the attack consists of the analytical calculation of a linear decision boundary that separates samples from different classes, followed by the addition of a perturbation perpendicular to that decision boundary (figure 3). In neural networks, these decision boundaries are almost always non linear, so they add the perturbations iteratively by performing the attack multiple times, finishing when an adversary is found. The overshoot parameter is used as a termination criterion to prevent vanishing updates.

Although this method allows the generation of adversarial samples with less perturbations than FGSM and JSMA (using Euclidean distance as a metric for comparison) and with higher misclassification rates, it is more computationally intensive than both.

Carlini and Wagner created a new attack based on the L-BFGS attack, called **Carlini & Wagner attack (C&W)**, by representing the attack as an optimization problem [37].

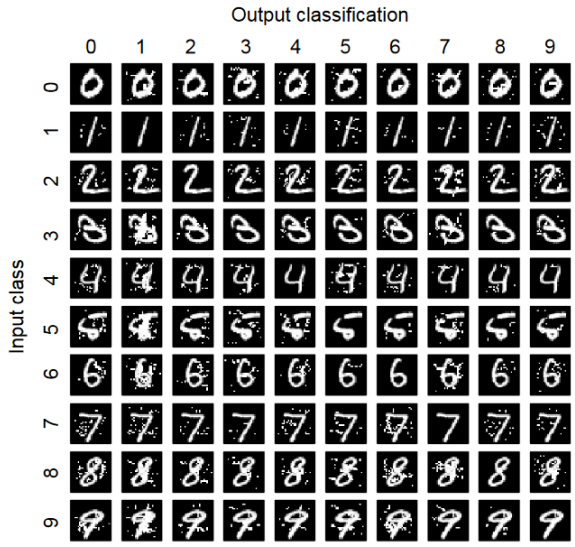


FIGURE 2. JSMA on MNIST dataset. The original samples are found in the diagonal, with all other cells containing adversarial examples with different target classes (in [55]).

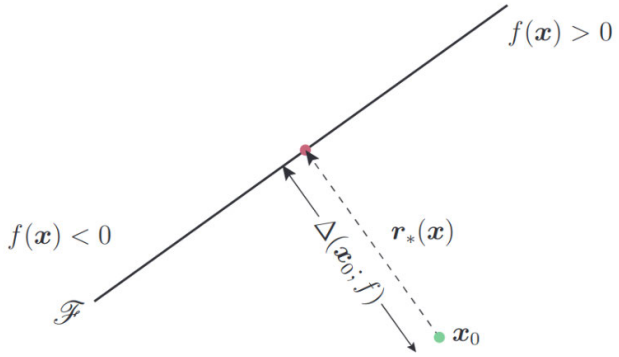


FIGURE 3. Deepfool - a perturbation r is added to a sample x0 in the direction of the hyperplane that separates the original class and the target class (in [59]).

Instead of using the same loss function as in L-BFGS, the authors proposed the usage of other loss functions, such as hinge loss instead of cross entropy loss, and added a new variant w to the minimization problem to avoid the box constraints, making the problem more efficient to solve.

This method is more efficient than L-BFGS at generating adversarial examples, and was shown to be able to defeat state of the art defenses, such as defensive distillation and adversarial training.

Generative adversarial networks (GAN) have also been used to generate adversarial attacks. GANs were initially proposed by Goodfellow *et al.* [62], and can be described as a machine learning system, where two neural networks compete with each other, with one acting as a generator, and the other behaving as the discriminator. The two networks play a zero-sum game, where the generator tries to produce samples that will be misclassified by the discriminator, while the latter will try to distinguish real samples from ones created

by the generator. This process trains the generator to synthesize data from a similar distribution of the real data, with the goal of deceiving the discriminator. Ideally, in the end, a Nash equilibrium between the generator and the discriminator will be found, where one of the networks cannot improve without changing the opposing networks parameters (figure 4).

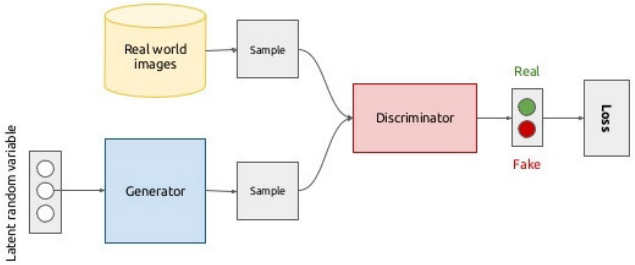


FIGURE 4. The general flow of a GAN. Only one loss function is used, which is be used to train both the generator and the discriminator ([62]).

One of the biggest problems with regular GANs is that using exclusively gradient descent to train the networks often leads to mode collapse to a parameter setting, always emmitting the same point [56]. So, the optimization can become highly unstable. **Wasserstein GANs (WGAN)** were introduced as a means to solve this problem. Instead of using the gradient of the loss function of the neural networks to train the models, they used the Wasserstein distance, also known as the earth-mover distance, which measures the distance between two distributions [35]. This technique was shown to eliminate many of the shortcomings of vanilla GANs, and provide a smoother training [35].

Although the generation of adversarial samples with GANs is effective, the distribution of the perturbations is more unpredictable than the previously mentioned gradient based methods, which all introduce perturbations under a certain distance metric, and the generation can also be highly unstable, even for WGANs [17].

Zeroth-order optimization attack (ZOO) was proposed by Chen *et al.*, and allows the estimation of the gradient of the classifiers without access to the classifier, which is ideal for a black box attack [39]. The method uses zeroth order stochastic coordinate descent to optimize the malicious samples, by iteratively adding perturbations to each feature of the samples and querying the classifier to estimate the gradient and hessian of the different features. The authors then use Adam optimizer to find the optimal perturbations for the target sample under first order using the estimated gradient, or Newton's method using both the gradient and hessian. This makes it an oracle based technique, not requiring the training of substitute models, and the attacker does not require information on the classifier.

Although this method was proven effective at estimating the gradient and hessian, showing results similar to the C&W attack, it requires a large amount of queries to the oracle, which can be used to detect the attacker in real scenarios.

Goodfellow *et al.* explored a property of adversarial inputs, which is **transferability** across models [58]. The authors observed that, when an adversarial input is successfully misclassified by a model, it will often be misclassified by other models, even when they have different arquitectures or were trained in different datasets. It was also observed that they often agree on the predicted classes for multi-class scenarios. This property is particularly useful when performing black-box attacks, where an attacker creates a substitute model trained on data following the same distribution, and is able to generate samples that are misclassified on the target model.

Most of the attacks presented were initially tested on image domains, where nearly negligible perturbations were introduced to existing images, which caused the models to misclassify these malicious samples.

Although these attacks were initially proposed to target pixels in images, they can equally be applied to other types of data, such as tabular datasets with a limited number of features, since these attacks are not data type dependant. In this case, a normalization of the features would create a similar scenario to the ones of images, where every pixel has a set range of possible values.

This poses as a security threat, as an attacker can virtually target any type of data used by a classifier to produce adversarial examples, such as modifying existing malware to prevent their detection, or manipulating denial of service attacks to go undetected.

Some limitations exist for the attackers in these situations, such as potentially not knowing the features used by the classifiers, or having limited access to modify certain features. Nonetheless, the threat of adversarial attacks still exists.

A summary of the attacks explored in this section can be explored in table 1.

IV. ADVERSARIAL DEFENSES

To counteract many of the attack strategies illustrated in the previous section, many approaches have been proposed to improve the resilience of classifiers to adversaries.

Goodfellow *et al.* proposed **adversarial training** on the imaging field, the first method to improve resilience to adversaries which consists of including adversarial examples in the training set [58]. Instead of directly including adversarial examples in the training set, the authors propose the usage of an adversarial objective function, by introducing the FGSM attack to the loss function of the neural network, which replicates the same effect. The authors concluded that this technique was effective at regularizing the classifier, although misclassifications still occurred. The original authors were able to reduce the error rate from 89.4% to 17.9% using adversarial training against FGSM on the MNIST dataset [58]. Swami *et al.* further explored this technique by showing superior effectiveness when introducing perturbations on intermediate layers of deep neural networks instead of the input layers [47]. The main disadvantage of this method is that it needs to be trained against specific types of attacks to be

TABLE 1. Summary of most common and explored adversarial attacks.

Attack	Distance metric	Description	Advantages	Disadvantages
L-BFGS	L_2	L-BFGS optimization used to minimize the amount of perturbations added to images	-Effective at producing adversarial examples	-Very computationally intensive, as it is an optimization method with has box constraints
FGSM	L_{inf}	Flat perturbations are added to every feature, in the sign of the gradient	-Computing time efficient	-Perturbations are added to every feature
JSMA	L_0	Flat perturbations are added to features iteratively according to saliency value, by decreasing order	-Very few features are perturbed	-More computationally intensive than FGSM
Deepfool	L_2	Decision boundaries between classes are estimated, and perturbations are added iteratively	-Effective at producing adversarial examples	-More computationally intensive than FGSM and JSMA -Adversarial examples are likely not optimal
C&W	L_0, L_2 or L_{inf}	Similar to L-BFGS attack (optimization problem), but without box constraints and different objective functions	-Very effective at producing adversarial examples -Can defeat some adversarial defenses	-More computationally intensive than FGSM, JSMA and Deepfool
GAN/ WGAN	*none*	Two neural networks are trained: one to generate samples, and the other to distinguish real and generated samples	-Can generate samples that are different from the ones used in training	-Training a GAN is very computationally intensive, and can be highly unstable
ZOO	L_2	Estimate gradient and hessian by querying the target model with modified individual features, and use Adam or Newton's method to optimize perturbations	-Similar performance to C&W attack	-Requires large amount of queries to the target classifier

resilient to them, being mostly effective against L_{inf} attacks, such as FGSM [44].

Gradient masking comprises a group of defensive techniques which assume that "if the model is non-differentiable or if the model's gradient is zero at data points, then gradient based attacks are ineffective" [23]. One form of gradient masking is **gradient hiding**, which consists on using non differentiable models to perform classification, such as decision trees. This prevents the adversary from estimating the gradient and using it to generate adversaries. Gradient smoothing comprises a range of techniques that smooth out the model's gradient, causing numerical instabilities which hamper the estimation of the gradient.

Although the objective of preventing the adversary of estimating the gradient was achieved, Papernot *et al.* concluded that, both under black-box and white-box scenarios, the training of substitute classifiers to estimate the gradient is an effective strategy to generate adversarial attacks against gradient masking because of the transferability of the attacks, making gradient masking an ineffective defense [53], [54].

Another defense technique was proposed by Papernot *et al.* called **defensive distillation** [52]. The authors propose the usage of distillation as a means to improve the robustness of a classifier to adversarial inputs. Instead of using distillation in the conventional way to train a different, simpler model from a teacher model to allow deployment on resource constrained devices, it is proposed to use the knowledge extracted during the distillation process to improve the classifier itself at detecting adversarial samples. By increasing the temperature at which a neural network is trained on the softmax layer, the teaching network outputs have less confidence on the ground truth label and show a higher distribution across classes. It is then proposed to use this output to train a second network with the same structure as the first. The structure is not changed since the objective is to make the network more resilient, and not more computationally efficient. The authors observed that, by using the entropy generated in the softmax layer to train the same network, it will become more resilient to adversarial samples, as it prevents the network from fitting too tightly to the ground truth class labels. An overview on the proposed technique is in figure 5.

To test the proposed method, the authors used MNIST [7] and CIFAR10 [2] datasets, and the results showed that distillation reduces the success of adversarial samples from 95.89% to 0.45% on the MNIST dataset, and 78.9% to 5.11% on the CIFAR10, with variations of accuracy in the detection of normal samples in the range of -2% and 2%.

Although this technique was proven effective against adversarial attacks, it requires the training of an entire new model, and was shown to be ineffective against C&W attacks.

Xu *et al.* proposed **feature squeezing** as a means to combat adversarial examples [48]. The intuition behind this technique is to compress the features of the sample (the pixels of the image in their case) and perform classification on the compressed sample. If the prediction by the classifier on the compressed sample is substantially different from the prediction of the original sample, it is considered an adversarial example. The authors tested several compression methods, namely bit depth compression, median smoothing and non local means, but found that the best method was largely dependent on the dataset. An overview of the feature squeezing framework can be seen in figure 7.

This technique was proven effective on MNIST, CIFAR10 and ImageNet [69] datasets against many state of the art adversarial attacks, such as FGSM, JSMA, Deepfool and C&W under different distance metrics, showing an improvement between 20% and 70% on the detection rate for all datasets, with bit depth squeezing being the most effective on the MNIST dataset and median smoothing the most effective in CIFAR10 and ImageNet.

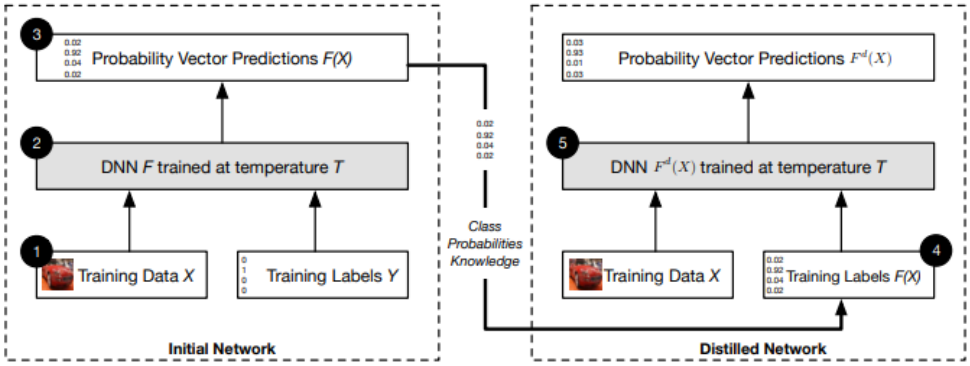


FIGURE 5. Overview of defensive distillation technique. A neural network F was initially trained and performed predictions under a certain temperature T. The predicted labels distribution is then used to train a second neural network with the same structure, which is shown to be more resilient to adversaries (in [52]).

The main limitations of this technique reside in the choice of compression methods, as the best ones were different for each dataset. This strategy is also limited to the field of images, as applying the compression methods tested in this study to tabular data would result in the loss of a substantial amount of data.

Hosseini *et al.* proposed a new technique to block the transferability of adversarial examples across different models on black-box environments [42].

It was previously shown that adversarial examples crafted to target a specific classifier often work on different classifiers, even if their training data or architecture are different. Instead of trying to create a defense mechanism that assigns adversarial examples to their original label, the authors propose discarding adversarial examples, by creating a new class called NULL, which indicates if the input is suspicious.

The authors started by training a classifier exclusively with normal data. After the initial training, they computed a function that outputs the NULL probabilities, which represent the probability of a sample being adversarial based on the amount of perturbations present. They introduced perturbations using a brute-force method, which iteratively modified certain features, and repeated this process for multiple number of features perturbed. The third and final step was to perform adversarial training, and deciding the label based on the NULL probabilities (figure 6).

Although this method is not intended to identify the true label of a certain sample, it is one of the most effective techniques in literature aimed at identifying adversarial examples.

The results show the transferability rate of different attacks on the MNIST and GTSRB [4] datasets reduced from over 55% to less than 6%.

The main limitation of this technique is not being able to find the original label of the adversarial samples, since these are labeled as NULL.

Naveed Akhtar and Jian Liu proposed the **Universal perturbation defense method** as a defense method against adversaries [32]. The method consists of placing a *perturbation rectifying network* (PRN) before the input layer of the target classifier. This network is trained on images with and without perturbations, having the layers of the classifying network frozen. This process creates a network that is effectively denoising the inputs and, by evaluating the difference on the inputs and the outputs of the PRN, extracts discriminative features, which are used to train a binary classifier that is able to identify adversarial samples. The results on a GoogleNet dataset [60] show that the detection rate of adversarial attacks was between 91.6% and 97.5%, both against L_{inf} and L_2 attacks.

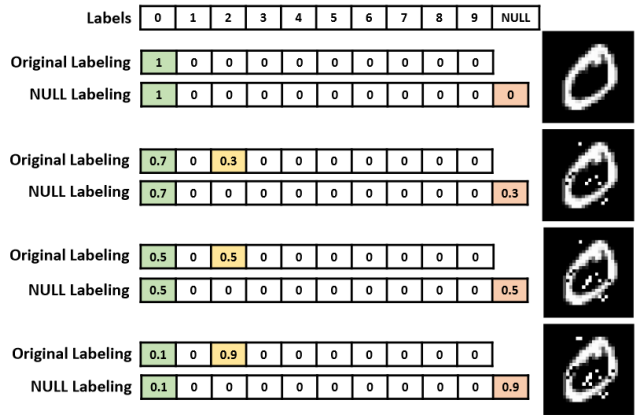


FIGURE 6. Transferability defense: instead of assigning a larger credibility to the target class during training, the network is trained to assign it to the NULL label, proportionally to the magnitude of the perturbations (in [42]).

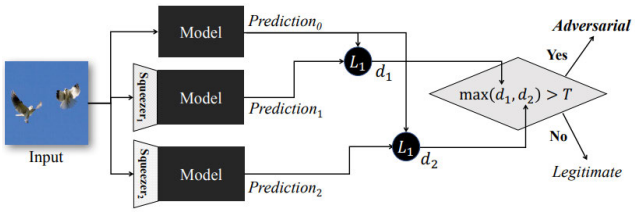


FIGURE 7. Framework of feature squeezing. When the difference of predictions on different models using different/no squeezers exceeds a threshold, the input is considered adversarial.

This technique is especially useful in practical scenarios since it offers defenses without need of modifying a previously trained classifier, although a PRN still needs to be trained.

Dongyu Meng and Hao Chen proposed a defensive framework called **MagNet** [45]. The framework consists of two main modules: a detector and a reformer.

The authors consider that a model misclassifies an adversarial attack for two reasons: the example is distant from the boundary of the manifold of the normal examples, or the example is close to the manifold of normal examples, but the classifier does not generalize well.

The purpose of the detector is to defend against the first case (adversarial examples distant from the manifold), and it is built using an autoencoder. The autoencoder is trained on normal samples, and during deployment, it rejects samples that deviate substantially from the manifold. It does this by verifying the loss function, which is the mean squared error, and rejects samples with an error above a set threshold. The reformer then receives samples that were classified as normal by the detector, and ''denoises'' the inputs, to remove small perturbations that were not caught by the detector, and approximate the adversarial samples to normal samples. The output of the reformer is then input on a classifier, which will perform classification among the normal classes (figure 8).

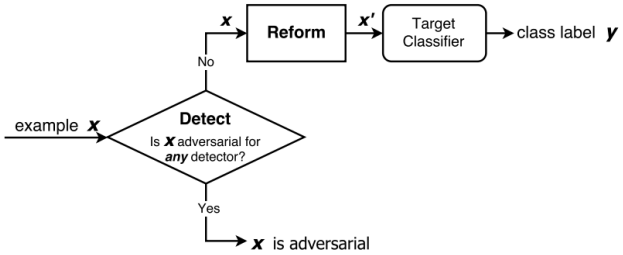


FIGURE 8. MagNet defense strategy. Using an ensemble of detectors, if any considers a sample to be adversarial, it is considered to be adversarial, removing samples with high magnitude perturbations. Regular samples and adversarial samples with small perturbations are then directed to the reformer, which denoises the data and directs it to a classifier that performs predictions (in [45]).

The authors tested the proposed technique on the MNIST and CIFAR datasets, using FGSM, IGSM (an iterative variation of FGSM), Deepfool and C&W. Initially, the attacks were effective at decreasing the performance of the classifiers, with C&W being the most effective by reducing the accuracy to 0%, and Deepfool to 19.1%. After deploying the defense, all accuracies improved, with all being above 92% on the MNIST dataset, and above 77.5% on the CIFAR dataset.

This strategy also has the advantage of not requiring the training of a new model to perform classification, although the training of autoencoders can be costly.

A summary of the defenses seen in this section can be explored in table 2.

V. APPLICATIONS TO INTRUSION AND MALWARE SCENARIOS

In this section, we explore different works that have applied adversarial machine learning to intrusion and malware detection scenarios. We selected all articles found on intrusion detection, and selected the most cited on malware detection.

Maria Rigaki and Ahmed Elragal first tested the effectiveness of adversarial attacks in an intrusion detection scenario [46]. The authors performed tests on the NSL-KDD dataset [8], by using **FGSM** and **JSMA** to generate Targeted attacks, and used 5 models to perform classification: decision tree, random forest, linear SVM, voting ensembles of the previous three classifiers and a multi-layer perceptron neural network (MLP). The results on JSMA showed that all classifiers accuracy was affected, with linear SVM being the most substantial one with a drop of 27%. The drop on F1-score and AUC was also notable, especially on the Linear SVM and Voting ensemble. Overall, the most resilient classifier was random forest, which suffered smaller performance drops across all metrics, with a drop of 18% on the accuracy and 6% on the F1 score and AUC. The authors made an important remark on the percentage of features modified by the attacks: FGSM modifies 100% of the features on every sample, while JSMA only modified on average 6% of the features. This makes JSMA a more realistic attack, since in the field of intrusion detection, there are domain specific limitations to the attacker relating to what features he can modify, which is coherent with the taxonomy of Huang *et al.* [67]. It was also noted that, although the explored attacks proved their effectiveness against unknown classifiers, they still required knowledge about the preprocessing of the data, such as the features used for classification, and the effectiveness of attacks under other circumstances was not tested.

Zheng Wang also tested the performance of adversarial attacks on the NSL-KDD dataset [8]. Similarly to Rigaki [46], multiple adversarial attack strategies were used to modify malicious samples in the dataset, this time using four different adversarial attack techniques to attack a MLP neural network: **FGSM, JSMA, Deepfool** and **C&W** [29]. On the original dataset, an AUC of 0.94 was achieved by the classifier on the normal data. The results on the attacks showed that they were effective at decreasing AUC, with C&W being the least effective attack, with an AUC of 0.80, and targeted FGSM being the most effective one, with an AUC of 0.44. JSMA was again identified as the most realistic attack, since a small range of features was modified, producing an AUC of approximately 0.5 for all classes. The authors also compared different attacks on the modified features, having found that there are mainly seven features used by almost all attacks, also discussing how an attacker can manually modify each one to perform adversarial attacks on a real scenario.

Zilong Lin et al. also performed adversarial attacks on the NSL-KDD dataset. Unlike the previous two studies [29], [46], where existing malicious samples were slightly modified by introducing perturbations, the authors created new

TABLE 2. Summary of most common and explored adversarial defenses.

Attack	Description	Advantages	Disadvantages
Adversarial training	Include adversarial examples in the training set or in the loss function of the model	-Easy to implement -Effective when attacks during deployment are similar to ones in training	-Requires retraining of the classifier -Can be ineffective against attacks different from the ones in the training set
Gradient masking	Consists of using techniques that make it hard for an attacker to estimate the gradient of a target classifier	-Can be an effective defense when the attacker needs to estimate the gradient	-Transferability of attacks across different models make this defense useless
Defensive distillation	Use neural network distillation to train a new network with the same structure as an original one	-One of the most effective defenses on most popular datasets	-Was shown to be defeated by C&W attack -Requires the training of a new model
Feature squeezing	Group of techniques that use compressed features to perform classification	-Very effective in image scenarios	-Only applicable to specific scenarios where compression is possible without large loss of information, such as images -Hard to choose best compression method
Transferability block	Train model with new label (NULL) with values proportional to the amount of noise in the sample	-One of the most effective techniques at identifying adversarial examples	-Can not identify original label of adversarial example
Universal perturbation defense method	Train a network that extracts features of adversarial examples, and use these to train another model that identifies adversarial examples	-Does not require training of a new model -Shows results similar to other defenses	-Attacks with different metrics produce different results
MagNet	Train an autoencoder to identify adversarial examples by measuring reconstruction error, and use another to reconstruct clean samples to perform classification on a separate classifier	-Does not require the retraining of the original model -Shows comparable results to other defenses	-Requires usage of autoencoders, which can be hard to train

malicious samples by using a variation of a WGAN called **IDSGAN** [25]. These samples follow a similar distribution to the ones in the dataset, but cause misclassification. By training a Generator to produce adversarial malicious traffic from real malicious traffic, and a discriminator that distinguishes normal traffic from adversarial, the authors were able to generate adversarial network traffic. The effectiveness of the attacks was tested against a multitude of classifiers, namely SVM, naive Bayes, MLP neural network, logistic regression, decision tree, random forest and k-nearest neighbors. By applying this concept to the NSL-KDD dataset, the results showed that the detection rates of adversarial examples on all classifier dropped from over 70% on the original data to less than 1% for all classifiers and types of attacks. To make the attacks more realistic, the authors prevented the IDSGAN from modifying functional features of the attacks, which are features that would affect the performance of the intrusions, such as time-based features for denial of service attacks, or content based features for user-to-root attacks. However, they observed no notorious difference from the performance with the inclusion of functional features, proving thus the effectiveness of IDSGAN at creating adversarial examples, although there were no limitations to the amount of noise introduced by the technique.

The authors concluded that IDSGAN is effective in generating adversarial attacks by creating new malicious samples, even when limiting the number of perturbed features to realistically and functionally modifiable ones, reducing the detection rate to close to 0%. For future work, it was proposed to test the technique on more types of attacks.

Qiao Yan et al. followed a similar approach to Lin *et al.* [25] and applied WGANs to the NSL-KDD dataset, this time exploring exclusively the generation of adversarial examples for denial of service samples only [17]. To decide what features are realistically modifiable, the authors split them into four categories: *basic features* refer to most elementary properties of the connection; *content features* refer to the content of the connection, and are not important for DoS attacks; *traffic features over a two second window* refer to statistical features over all the connections performed in

the last two seconds; *traffic features over 100 connections window* are statistics for the 100 previous connections, and are also considered not important to DoS attacks. Therefore, the authors only allow the modification of content features and traffic features of 100 connections windows. To prevent the modifications of these features, a converter was added between the generator and the discriminator, which replaces the unmodifiable features with their original values. After training the WGAN by generating adversarial DoS samples by approximating them to the distribution of normal traffic samples, the authors were able to decrease the true positive rate of a convolutional neural network classifier (CNN) from 97.3% to 47.6%, showing the effectiveness of the attack while maintaining the integrity of the denial of service attack by not modifying its functional features.

It is concluded that DoS-WGAN is effective at generating adversarial examples against a CNN based IDS. Information entropy was calculated for all scenarios and its stability through the training confirms that WGAN is more stable than other models. The diversity of attacks was also observed to be greater when using WGAN.

Yang et al. used three attack strategies on the NSL-KDD, against a simple neural network classifier. Two of those strategies were already explored in previous works (C&W and GAN), while ZOO was never previously tested [30]. For the first scenario, a substitute classifier was trained and used to generate adversarial samples using C&W to attack a separately trained model. On the second scenario, to attack the target classifier without training a substitute one, ZOO was used by querying the classifier to estimate the gradient and produce adversarial attacks. For the third scenario, a WGAN was trained to generate adversarial samples. The results achieved were the following: with the C&W attack, a deterioration of 24% in the F1 score was achieved, from 0.898 to 0.687; with the ZOO attack, the deterioration was more notorious, with 70% on the F1 score, from 0.898 to 0.273; with the WGAN attack, the F1 score reduced by 62%, from 0.989 to 0.350. The authors concluded that, although ZOO was the most effective technique, it was also the least realistic, as it required an excessive amount of queries to generate adversarial examples, which would be detected in a real scenario. They also concluded that the GAN technique was very powerful, although the training procedure was still unstable, even when using WGAN, and suffered from model collapse and convergence failure.

Their work showed the practicality of crafting adversarial attacks using more varied techniques against deep neural networks, with all attacks being effective at deteriorating the performance of the classifiers. It is also noted that the attacks are effective when the attackers have no access to the classifier used by the IDS (black-box attack), but, for greater effectiveness, knowledge of the parameters of classifier can be advantageous.

Nuno Martins et al. tested the effectiveness of four adversarial attacks which modified existing malicious samples, similarly to Wang [29], on the NSL-KDD and CICIDS2017 datasets (only on denial of service records), with the aim of analyzing the footprint of multiple classifiers [13]. Four adversarial strategies were used: FGSM, JSMA, Deepfool and C&W, and the classifiers that were used were: decision tree, random forest, SVM, naive Bayes, neural network and DAE. The DAE was used for classification by first training the autoencoder, and then freezing those layers and training new layers attached to the end of the autoencoder, which performed classification. It was found that all techniques were able to deteriorate the performance of the classifiers, with the average AUC decreasing by 13% on the NSL-KDD and by 40% on the CICIDS. This is mainly due to the larger proportion of modifiable features available on the CICIDS dataset, which exposes more vulnerabilities to an adversary. It was observed that the DAE was the most resilient classifier, showing only a drop of 1% of AUC on both datasets. Adversarial training was then tested as a defensive technique on all classifiers, and it was observed that the performance of all classifiers improved, with random forest being on average the best performing classifier on both datasets, which only suffered a 0.1% AUC loss from the baseline tests on normal data for both datasets.

Apruzzese *et al.* performed an evaluation of integrity attacks against network intrusion detectors [10]. Three different models were attacked: random forest, MLP neural network and KNN. The testbed consisted on the CTU-13 dataset. The detectors were split in multiple instances, each devoted to a specific botnet family, and using a 80/20% ratio as training and testing sets. The recall measure of all detectors was between 0.93 and 0.97, with random forest being the highest performing and KNN the worst. The authors then implement a custom adversarial attack, which targets three features: *exchanged_bytes* (number of bytes exchanged), *duration* (duration of the connections) and *total_packets* (number of packets exchanged). The attack works by adding random values within a specific interval to each feature, leading to new adversarial samples. The recall lowered to between 0.34 and 0.31, with the highest performing classifier being random forest, and the lowest the MLP neural network. The authors then tested two defenses: adversarial training and feature removal. When using adversarial training, the recall was increased to between 0.49 for KNN and 0.60 for random forest. When using feature removal, which consists on training detectors with every feature except the ones being perturbed, the recall was between 0.76 for MLP neural network and 0.89 for random forest.

Giovanni Apruzzese and Michele Colajanni also tested the effectiveness of adversarial attacks on a scenario where an attacker aims to spread a botnet malware inside a large network, while going undetected, by exclusively modifying the network flows [19]. This experiment was performed by using the CTU13 dataset, which contains 7 malware variants, while using a random forest detector trained on existing botnets with features related to network flows, such as protocol of the connection, IP addresses, ports and packets transmitted. In total, 7 classifiers were used (1 for each attack type),

since the authors defend that classifiers wield superior results when focusing on a single problem. The classifiers were trained on training sets, of which 95% were benign samples and 5% malicious. The detection rate of most attacks was over 0.95, with two exceptions, one of which should be mentioned, because it was so low (0.14): the Sugou attack, that had the least number of samples available. To generate adversarial samples, the authors randomly manipulate combinations of up to 4 features of the original malicious flows, namely duration of the flow, number of packets exchanged and number of bytes exchanged. The magnitude of the modifications was random, and these features were selected because they do not alter the logic of the attack. The detection rates were greatly reduced for every type of attack, proportionally to the number of features modified and magnitude of modification, reducing the detection rates to between 0.84 and 0.0.

Wu *et al.* applied deep reinforcement learning to generate adversarial attacks on botnet attacks [16]. In this scenario, the attacker has no knowledge of the detector (black-box attack), and only receives a boolean feedback on each query of the classifier, which states if the sample is detected as malicious or not. The framework for the attack consists on deep Q-learning, where the agent is rewarded when misclassification occurs, and each action is a modification to the sample, which must maintain the integrity of the attack. Since the representation of each flow is too large for the deep Q-learning framework, the authors used autoencoders to produce smaller feature vectors (1024 bytes). The authors then defined a limited set of actions to modify the sample, such as adding random values to the timestamps of the flow, adding length to the packets and appending benign packets.

Using the CTU-13 dataset, the authors trained two different models for validation: a decision tree and a CNN. The models were trained on 100000 botnet and benign flows, using a feature vector of 12 network related features, such as IPs, ports, number of bytes and duration. Both models achieved an accuracy of 0.99. After training an agent on 40000 flows with a maximum of 10 actions/modifications to existing flows, the authors achieved evasion rates between 0.35 and 0.50 on the CNN, and between 0.02 and 0.1 on the decision tree. The authors theorize that the CNN is more affected because it is trained on the compact representation of the autoencoder, and small perturbations can generate greater distortions on this representation, while some features that are not perturbed, such as IP addresses, are not modified on the original representation, which is used to train the decision tree.

Jin-YoungKim et al. proposed a technique based on the usage of **autoencoders** and **GANs** to detect zero-day attacks in malware [31]. Zero-day attacks are attacks which are unknown prior to their detection, and thus resemble adversarial examples.

The authors began the preprocessing of the data by modeling the code used for the attack as an image matrix, and trained a **denoising autoencoder** (DAE) with this data, extracting relevant features from the bottleneck layer of the DAE. Then, they used the decoder of the DAE as the data generator of the GAN, given a known probability distribution, and trained the discriminator of the GAN to detect the adversarial examples generated by the decoder. The discriminator was then used as the malware detector of the IDS, and trained using malware data. By using the dataset from the **Kaggle Microsoft Malware Classification Challenge** [6] to train and test the models, the proposed model achieved an accuracy of 98% at detecting zero-day attacks, outperforming every other tested conventional models (SVM, decision trees, random forest, adaboost, KNN, MLP neural network, naive Bayes, linear discriminant analysis (LDA) and quadratic discriminant analysis).

Joseph Clements et al. used four adversarial attacks on the Mirai botnet dataset [34], and used Kitsune [11], a network IDS (NIDS) system recently proposed by Mirsky *et al.* [27] to perform detection of intrusions. The IDS is based on an ensemble of autoencoders, which receives a feature vector of preprocessed packets. The ensemble was trained exclusively on regular data, making the autoencoders rebuild regular instances of data, using root-mean-squared-error (RMSE) as an objective function. The IDS then performed classification during deployment based on the RMSE of input samples, triggering an alarm when it was above a threshold value. Four adversarial attack techniques were used: FGSM, JSMA, C&W and Elastic Net Method (ENM [40]). When performing integrity attacks, all techniques achieved 100% false negative rate (FNR). Under availability attacks, only C&W and ENM achieved 100% FNR, with FGSM achieving 4% and JSMA 0%. It was concluded that the attacks were effective against Kitsune, but the adversary required knowledge of the data representation to perform the attacks. The authors also predict that, as feature representations in NIDSs become more automated and less human knowledge will be necessary in the future, adversarial attacks will become a larger concern.

Biggio *et al.* proposed a technique to generate adversarial examples and test it on malware classifiers for PDF files [36] using the Contagio dataset [3]. A black-box scenario is assumed, where the attackers only have knowledge of the distribution of the training data and the feature space used by the target. The method proposed consists of optimizing the following formula:

$$x^* = \min_x g(x)$$
$$subject\ to\ d(x, x^0) < d_{max}$$

Here $g(x)$ is the loss of the replicated classifier, x is the original sample, x^0 is the desired adversarial sample and x^* is the generated sample. Since this is a non-trivial problem the authors propose using optimization methods to solve it by using gradient descent. A problem with using gradient descent in this scenario is that it might lead to local minimums, or unsupported regions since the distribution of the data known by the attacker might not be exact. As such, the authors modified the formula, by minimizing $\hat{g}(x)$ while maintaining the generated samples in highly populated regions (of regular samples). The authors then

applied this technique to generate adversarial malware on PDF files, by using a feature representation proposed by Maiorca *et al.* [66] which consists of the tally of occurrences of keywords. The results showed that False Negative rates of up to 100% were achieved for both neural network and SVM classifiers.

The authors believed that these results could be extended to non-differentiable classifiers, such as k-nearest-neighbors and decision trees, and also proposed the usage of other techniques such as bagging or random subspace method.

Weiwei Hu and Ying Tan generated adversarial malware examples in a black-box oracle scenario using a new system based on GANs called MalGAN [43] on a malware dataset [5]. The system consisted of a generator, a substitute detector (discriminator) and a black box detector. The generator received real malware samples, along with a noise vector which works as a seed for the output (an adversarial sample). These adversarial samples, along with benign samples, were then input to the substitute model, which predicted if they were malware or not. The main difference between this system and other GANs is that the ground truth labels are not used to train the system, but, instead, the output of the black box model (the target model) is used, since the objective is to deceive that classifier. The authors attacked six classifiers from different architectures: random forest, linear regression, decision tree, SVM, MLP neural network and a Voting Ensemble of the previous classifiers. On normal data, all classifiers achieved true positive rates of over 90%, with decision tree showing the best performance of 98%. After generating adversaries, the true positive rate dropped to below 0.20% for all classifiers, both for the training and testing sets.

Grosse et al. attacked a binary neural network classifier trained on the Drebin dataset, an Android malware dataset [41]. Since adding perturbations to dynamically gathered features is hard, the authors took the simpler approach of only considering static features. The authors used a feature vector exclusively with binary features which represented mainly system calls, and applied an attack strategy similar to JSMA, but instead only allowed the perturbations to cause extreme values (0 or 1) since the features were binary. After testing the effectiveness of the attack, the authors reached misclassification rates between 63% and 69% based on the proportion of malware samples on the training set, having lower misclassification rates when the malware ratio was higher.

Two defensive techniques were also tested: distillation and adversarial training. When applying distillation, a drop of up to 2% in the accuracy was observed on the normal data, but the misclassification rate dropped drastically up to 38% when under attack. Adversarial training was also proven effective, but largely depended on the proportion of adversarial samples used to train the classifiers.

Andersonn *et al.* took a different approach at attacking a malware dataset by using deep Q-learning, a reinforcement learning technique [33]. The environment was defined as a feature vector comprised of 2350 categorical features on the metadata of the samples, and the action space was defined as a set of valid modifications to the samples. Rewards were provided when a sample successfully misclassified the target model. The authors attacked a gradient boosted decision tree, and achieved an evasion rate of 16%, which is a very modest result, but serves as a proof of concept, and the results give insights on which features were used to attack the model.

Weilin Xu et al. performed evasion attacks on the Contagio PDF malware dataset using a genetic programming approach [57], with the aim of evading two malware detection methods. The attack strategy requires knowledge of the feature representation by the defense, and an oracle indicating if a given sample is classified as malign or benign. The attack generation process consisted of applying genetic algorithms on existing samples until evasion was achieved. Only mutation operations were performed, by either inserting, removing, or replacing objects in the PDF file's tree with low probability. On each iteration, only samples which retain malicious behavior according to the oracle are kept, and the fitness function corresponds to the evasiveness of the attack. This attack strategy was tested against two detectors: PDFrate and Hidost. PDFrate consists on a random forest classifier which uses PDF metadata and object metadata as features, while Hidost is a SVM with a RBF kernel, which uses structural paths of objects in the PDF file as features.

The results showed that an evasion rate of 100% was achieved when attacking both defenses, while maintaining the integrity of all attacks. In total, 16,985 evasive variants were found for PDFrate, and 2,859 for Hidost.

A similar approach was followed by Calleja *et al.* [20], where the authors use genetic programming techniques to modify malicious Android apps to be mislabeled to different malware families in the DREBIN dataset, using RevealDroid to perform classification, which consists of a decision tree with various static features from the app. The main objective was to cause the malicious apps to be misclassified to the wrong malware family. The authors achieved this by performing feature addition, which consists of adding new features to existing apps, such as API calls and new permissions. Feature removal was more difficult to perform, since the integrity of the attacks could be compromised. This addition was achieved by using four genetic operators: selection, reproduction, crossover and mutation.

By performing these genetic operations on a population with different malware families, the authors were able to achieve misclassification in 28 out of the 29 malware families, by only adding one new feature.

Menéndez *et al.* [14] used **entropy time series analysis (EnTS)** to perform detection of malware hidden using different forms of polymorphism, such as compression and encryption, by assuming these processes change the entropies of their binaries. The process consisted of separating the file as a stream of chunks, and the entropy is calculated from the byte frequencies on each chunk, followed by a denoising step using wavelet analysis. The resulting feature

TABLE 3. Summary of works on Adversarial Machine Learning applied to intrusion and malware scenarios.

Paper	Attack Techniques	Classifiers	Datasets	Metrics	Results & Conclusions
Intrusion Detection					
[46]	JSMA	Decision tree Random forest Linear SVM Voting ensemble of 3 previous techniques Neural network (MLP)	NSL-KDD	Accuracy F1 AUC	- Random forest most resilient (10% accuracy drop) - SVM most affected classifier (27% accuracy drop)
[29]	FGSM JSMA Deepfool C&W	Neural network	NSL-KDD	Accuracy F1 AUC	- C&W least effective attack (0.8 AUC) - FGSM most effective attack (0.44 AUC)
[25]	WGAN	Neural network	NSL-KDD	Detection rate	- Detection rates dropped from over 70% to less than 1% for all classifiers - Limitation on the range of modifiable features did not improve the performance of the classifiers
[17]	WGAN	CNN	NSL-KDD	True positive rate	- TPR reduced from 97.3% to 47.6%
[30]	C&W ZOO GAN	Naive Bayes Random forest SVM Neural network	NSL-KDD	F1	- ZOO was the most effective attack with a drop of 70% on F1 - C&W was the least effective attack with a drop of 24% on F1. - JSMA would be the most practical in real scenarios, (modifies less features) and ZOO the least realistic (requires too many queries)
[13]	FGSM JSMA Deepfool C&W	Decision tree Random forest RBF SVM Naive Bayes Neural network Denoising autoencoder	NSL-KDD CICIDS2017	AUC	- All techniques were effective at generating adversarial attacks, with the mean AUC decreasing by 13% (NSL-KDD) and 40% (CICIDS) - Denoising autoencoder was the most resilient classifier on adversarial data, with only 1% decrease of AUC - With adversarial training used, the most resilient classifier is random forest, with a decrease of 0.1% of the mean AUC from regular data
[10]	Random noise added to selected features	Random forest Multi-layer perceptron KNN	CTU-13	Recall	- Best baseline classifier was random forest (0.97 recall) and worst was KNN (0.93 recall) - Best classifier under attack was random forest(0.34 recall) and worst was MLP NN (0.31 recall) - Adversarial training improved recall to between 0.49 and 0.60, and feature removal improved recall to between 0.76 and 0.89
[19]	Random noise added to selected features	Random forest	CTU-13	Detection rate	- Detection rate on most attacks over 0.95, with one exception of 0.14 due to lack of samples - Detection rates reduced proportionally to number of features attacked and magnitude of perturbations
[16]	Deep Q-learning	Decision tree CNN	CTU-13	Accuracy Evasion rate	- Both models had a baseline accuracy of 99% - Evasion rates between 0.35 and 0.5 on CNN (mainly due to autoencoder representation) - Evasion rates between 0.02 and 0.1 on decision tree
Malware Detection					
[31]	GAN	Decision tree Random forest SVM KNN Naive Bayes LDA	Kaggle Malware VirusShare	Accuracy	- 98% accuracy reached, when other techniques reached between 66% and 96% accuracy
[11]	FGSM JSMA C&W ENM	Kitsune NIDS (ensemble of autoencoders)	Mirai botnet	False negative rate	- All attacks achieved 100% FNR under Integrity attack scenario - Under availability attacks, FGSM and JSMA only achieved 0% and 4% FNR, with C&W and ENM achieving 100%
[36]	*Gradient Based Technique*	SVM Neural network	Contagio	False negative rate	- False negative rates went up to 100% for Neural networks and SVM classifiers - The attacks are effective in a black-box scenario
[43]	GAN	Neural network	malwr	True positive rate	- All classifiers performed with over 90% TPR, with decision tree being the best performing - TPRs dropped below 0.20% for all classifiers
[41]	Modified JSMA	Neural network	DREBIN	Misclassification rate	- Misclassification rate between 63% and 69% when under attack - Distillation deteriorates the accuracy by 2% but improves MR to 36% - Adversarial training effectiveness largely depends on the proportion of adversarial samples used used
[33]	Deep Q-learning	Gradient boosted DT	*Custom*	False negative rate	- Evasion rate of 16% was achieved for black box attack
[57]	Genetic Algorithm	PDFrate (Random forest) Hidost (RBF SVM)	Contagio	Evasion rate	- Evasion rate of 100% against both defenses - 16,985 evasive variants found for PDFrate, and 2,859 for Hidost
[20]	Genetic Algorithm	RevealDRoid (Decision tree)	DREBIN	Evasion rate	- Evasion rate of 100% achieved for 28 out of the 29 families of malware
[14]	*El Empaquetador Evolutivo*	Entropy time series analysis	Kaggle Malware VirusShare	Precision Accuracy False negative rate	- EnTS achieved 82% accuracy when maximizing precision, and 94% when maximizing accuracy - False negative rate of 90.8% to 98.7% when attacking with EEE, up from 0% to 9.4% without attacks
[38]	*EvnAttack*	unspecified	Comodo Cloud Security Center Dataset (Private)	F1 Score False negative rate	- *EvnAttack* reduced F1 from 0.96 to 0.43 and increased FNR from 0.05 to 0.70 - *SecDefender* increased F1 from 0.43 to 0.95 and also reduced the FNR substantially
[49]	*GAN*	Random forest	Alexa top 1M domains)	AUC	- AUC decreased from 0.99 against regular attacks to 0.94 against GAN

vector is then used to train a random forest classifier which distinguishes benign from malware samples. The authors tested the technique on multiple datasets: Kaggle malware competition dataset, packet (Pck) malware from VirusShare, and Mix, a dataset built from the previous two. The results showed that EnTS achieved 82% accuracy when maximizing precision (100%), and 93.9% accuracy when maximizing accuracy, surpassing all other tested state of the art detection

strategies and being the most scalable because of linear complexity.

To further evaluate the effectiveness of the defense, the authors presented a new attack called **"El Empaquetador Evolutivo" (EEE)**. This attack manipulates the entropy signature of the malware sample by injecting controlled entropy regions (CERs) without modifying the semantics of the malware (so it remains effective). To optimize the placement of the CERs, the authors used genetic algorithms as a learning process to optimize the misclassification of the target defense (EnTS). As a result, the false negative rate went from [0%-9.4%] to [90.8%-98.7%] after the attack, while maintaining the functionality of the malware.

Chen *et al.* [38] proposed both an evasion attack model (*EvnAttack*) on malware present in portable Windows executable files, and a defense for that attack (*SecDefender*). The feature vectors used for the experiments were binary features, each representing an API call to Windows (1 for used, 0 for not used). The authors used Max-Relevance algorithm [71] to compute the importance of each feature on a labeled dataset, and separate them into two sets: one of features relevant to malware (M), and another with features relevant to benign samples (B). To perform the attack *EvnAttack*, they then used a wrapper method which iteratively selects features to be added on a sample from the subset B and removed from the subset M, with the aim of maximizing the loss function of the target classifier. To implement a defense to this attack, the authors exploit this type of attack to retrain the classifier in a progressive way, by training the classifier both in the original dataset and generated adversarial samples. They also added a regularization step to penalize adversarial samples with high evasion costs (many features modified), since these attacks are typically not feasible, as they damage the malicious capability of the sample.

The authors performed tests on a private dataset obtained from the Comodo Cloud Security Center, with 10.000 samples, of which 5.000 were malicious and 5.000 were benign, with 3,503 features (API calls). The original classifier achieved an F1 score of 0.96 on the testing set, and when faced with *EvnAttack* with a maximum evasion cost of 22, it reduced to 0.43, while the FNR increased from 0.05 to 0.70. When deploying SecDefender, the F1 score was raised back to 0.95, while the FNR greatly reduced.

Anderson *et al.* applied GANs to domain generation algorithms (DGA), which are typically used by malware, allowing updates to the malware from multiple generated domains [49]. The authors initially trained an auto-encoder using the Alexa top 1M dataset, which contains the domain names of the top one million websites listed at [1]. The decoder of the autoencoder is then used as a generator for the GAN, receiving a seed of 20 random numbers as seed and outputting a pseudorandomly generated domain. The encoder of the autoencoder was then used to distinguish real samples from the original Alexa 1M dataset and generated samples by training a logistic regression model on the output of the encoder. Once the training of the GAN was complete, the authors used a random forest classifier with manually crafted features used in other state of the art works, such as length, entropy character distribution and vowel to consonant ratio. Against other state of the art DGAs, the random forest classifier always achieved an AUC 0.99 or above, but only achieved an AUC of 0.93 against samples generated by the generator of DeepDGA.

It is presented in table 3 an overview of the different works explored in this section, and in table 4 a summary of all the datasets that were used.

TABLE 4. Summary of datasets used in the literature review (section V).

Ref.	Dataset	Description
[8]	NSL-KDD	Intrusion dataset with DoS, Probe, User 2 Root and Remote to User attacks
[28]	CICIDS2017	Intrusion dataset numerous types of attacks (DoS, Heartbleed, XSS, SQL Injection, Botnet...)
[34]	Mirai Botnet	Dataset with Mirai malware botnet, which affected Linux systems to perform DDoS attacks
[3]	Contagio	Database containing multiple types of malware, both for mobile, IoT and desktop devices
[5]	malwr	A program sharing site which evaluates software as malware or benign
[61]	DREBIN	Dataset containing Android apps from 179 malware families
[6]	MS Kaggle Malware	Malware dataset published by Microsoft in the Kaggle website
[9]	VirusShare	Repository of many different malware samples for live testing by researchers
[51]	CTU-13	Labeled dataset generated at CTU university containing botnet and benign traffic.
[1]	Alexa top 1M	Dataset containing top 1 million domains at alexa.com

In the case of intrusion detection, several adversarial attack techniques have been explored against multiple classifiers from different families and structures, all showing effectiveness at deteriorating the performance, with Deepfool, WGAN and Zoo being the most effective, but JSMA the most realistic, since it perturbs less features. All basic classifiers performed approximately at the same level, with the exception of decision trees and naive Bayes that generally performed the worst. Although many attack strategies were explored, defenses were only explored in few studies (adversarial training based strategies and DAE), which proved their effectiveness at improving the resilience of the detection system. The variety of datasets is also scarce, with almost all works using exclusively the NSL-KDD dataset.

On malware detection scenarios, there is a greater variety of datasets, although not all are publicly accessible. There were also a great variety of explored attack strategies, with similar results to intrusion scenarios, although in malware scenarios there is more freedom on what features can be modified, making JSMA no more the only realistic attack. Adversarial defenses were tested in some studies (distillation, adversarial training, and attack specific defenses), which, again, were proven effective with little performance drops on regular data.

VI. CONCLUSION

In this paper we explored several works which apply adversarial machine learning concepts to intrusion and malware

detection scenarios. We first showed various fundamental concepts that can help the understanding of the basics of adversarial machine learning, along with adversarial attack and defense strategies. We then explored the application of these techniques to intrusion detection and malware detection scenarios and concluded that:

- Adversarial attacks can deteriorate the performance of malware and intrusion classifiers, even if they follow different architectures or are from different families due to the transferability of adversarial attacks across different classifiers;
- Various adversarial attack strategies have been explored for both scenarios, with some strategies being more effective than others depending on the situation. Most attacks are generated using the gradient of a neural network to identify weaknesses in the classifiers, or by using genetic algorithms to optimize adversarial samples. There is a trade-off when choosing adversarial attacks, such as in the number of features modified, computing power required and magnitude of the perturbations;
- All classifiers displayed relatively similar results on normal data, but under adversarial attacks, decision trees, naive Bayes and linear SVMs were the most affected classifiers, while neural networks with a different architecture from the attacker, random forests and RBF SVMs were the most resilient;
- Adversarial defenses were very little explored, but their effectiveness was proven, with adversarial training being the most effective defense, both in an intrusion and malware scenarios;
- The number of datasets that were used on intrusion detection scenarios was small, with only NSL-KDD being used in six studies, CICIDS2017 being used in one study and CTU-13 in three studies. This is mainly due to the small amount of public labeled intrusion datasets. On the case of malware detection, there is a greater variety of datasets used across different studies due to higher availability of public malware datasets.

To further extend the studies that were analyzed in this paper, we propose for future work:

- Further testing of adversarial defense techniques: in total, four defense strategies were tested, with defensive distillation being used in malware detection, denoising autoencoders in intrusion detection and adversarial training in both scenarios, as well as other defenses specific to malware scenarios. Their effectiveness was proven, but there is a wider variety of defensive strategies that were proposed to image recognition that can be applied to intrusion and malware detection (also presented in section IV) as well as other deep learning based anomaly detection techniques [68];
- The usage of more recent and standardized datasets: in the case of intrusion detection, the main dataset used is NSL-KDD, which is greatly outdated, but is the mainly studied one due to lack of available alternatives. The growth of IoT environments in the recent years poses as a potential source for generating new test beds for intrusion detection;
- Performing live attacks against time based intrusion detection systems, such as recurrent neural network classifiers, as all attacks and detections were performed on independent samples.

REFERENCES

[1] *Alexa Top 1m*. Accessed: Jun. 2019. [Online]. Available: https://www.alexa.com

[2] *CIFAR-10 (Canadian Institute for Advanced Research)*. Accessed: Jun. 2019. [Online]. Available: https://www.cs.toronto.edu/~/cifar.html

[3] *Contagio Dataset*. Accessed: Jun. 2019. [Online]. Available: http://contagiodump.blogspot.it

[4] *GTSRB German Traffic Sign Dataset*. Accessed: Jun. 2019. [Online]. Available: http://benchmark.ini.rub.de/?section=gtsrb&subsection=news

[5] *Malwr Dataset*. Accessed: Jun. 2019. [Online]. Available: https://malwr.com

[6] *Microsoft Malware Kaggle Dataset*. Accessed: Jun. 2019. [Online]. Available: https://www.kaggle.com/c/malware-classification

[7] *MNIST Handwritten Digit Database*. Accessed: Jun. 2019. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[8] *NSL-KDD Dataset*. Accessed: Jun. 2019. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[9] *Virusshare Dataset*. Accessed: Jun. 2019. [Online]. Available: https://virusshare.com/

[10] G. Apruzzese, M. Colajanni, L. Ferretti, and M. Marchetti, "Addressing adversarial attacks against security systems based on machine learning," in *Proc. 11th Int. Conf. Cyber Conflict (CyCon)*, May 2019, pp. 1–18.

[11] J. Clements, Y. Yang, A. Sharma, H. Hu, and Y. Lao, "Rallying adversarial techniques against deep learning for network security," 2019, *arXiv:1903.11688*. [Online]. Available: http://arxiv.org/abs/1903.11688

[12] T. S. John, *Adversarial Attacks and Defenses in Malware Detection Classifiers*. Harrisburg, PA, USA: IGI Global, 2019, pp. 127–150.

[13] N. Martins, "Analysing the footprint of classifiers in adversarial denial of service contexts," *Proc. EPIA Conf. Artif. Intell.*, 2019, pp. 256–267.

[14] H. D. Menéndez, S. Bhattacharya, D. Clark, and E. T. Barr, "The arms race: Adversarial search defeats entropy used to detect malware," *Expert Syst. Appl.*, vol. 118, pp. 246–260, Mar. 2019.

[15] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput. Secur.*, vol. 86, pp. 147–167, Sep. 2019.

[16] D. Wu, B. Fang, J. Wang, Q. Liu, and X. Cui, "Evading machine learning botnet detection models via deep reinforcement learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[17] Q. Yan, M. Wang, W. Huang, X. Luo, and F. R. Yu, "Automatically synthesizing DoS attack traces using generative adversarial networks," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 12, pp. 3387–3396, Dec. 2019.

[18] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.

[19] G. Apruzzese and M. Colajanni, "Evading botnet detectors based on flows and random forest with adversarial samples," in *Proc. IEEE 17th Int. Symp. Netw. Comput. Appl. (NCA)*, Nov. 2018, pp. 1–8.

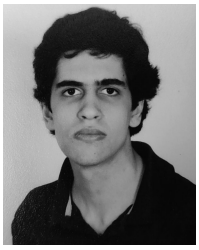
[20] A. Calleja, A. Martín, H. D. Menéndez, J. Tapiador, and D. Clark, "Picking on the family: Disrupting android malware triage by forcing misclassification," *Expert Syst. Appl.*, vol. 95, pp. 113–126, Apr. 2018.

[21] C. Chio and D. Freeman, *Machine Learning and Security: Protecting Systems with Data and Algorithms*. Newton, MA, USA: O'Reilly Media, 2018.

[22] P. Domingos, *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. New York, NY, USA: Basic Books, Inc., 2018.

[23] V. Duddu, "A survey of adversarial machine learning in cyber warfare," *Defence Sci. J.*, vol. 68, no. 4, pp. 356–366, Aug. 2018.

[24] G. Li, P. Zhu, J. Li, Z. Yang, N. Cao, and Z. Chen, "Security matters: A survey on adversarial machine learning," 2018, *arXiv:1810.07339*. [Online]. Available: http://arxiv.org/abs/1810.07339
[25] Z. Lin, Y. Shi, and Z. Xue, "IDSGAN: Generative adversarial networks for attack generation against intrusion detection," 2018, *arXiv:1809.02077*. [Online]. Available: http://arxiv.org/abs/1809.02077
[26] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. M. Leung, "A survey on security threats and defensive techniques of machine learning: A data driven view," *IEEE Access*, vol. 6, pp. 12103–12117, 2018.
[27] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018, *arXiv:1802.09089*. [Online]. Available: http://arxiv.org/abs/1802.09089
[28] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.
[29] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, vol. 6, pp. 38367–38384, 2018.
[30] K. Yang, J. Liu, C. Zhang, and Y. Fang, "Adversarial examples against the deep learning based network intrusion detection systems," in *Proc. MILCOM - IEEE Military Commun. Conf. (MILCOM)*, Oct. 2018, pp. 559–564.
[31] J.-Y. Kim, S.-J. Bu, and S.-B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Inf. Sci.*, vols. 460–461, pp. 83–102, Sep. 2018.
[32] N. Akhtar, "Defense against universal adversarial perturbations," 2017, *arXiv:1711.05929*. [Online]. Available: https://arxiv.org/abs/1711.05929
[33] H. S. Anderson, "Evading machine learning malware detection," *Black Hat*, 2017. [Online]. Available: https://blackhat.com/
[34] M. Antonakakis, "Understanding the Mirai Botnet," in *Proc. 26th USENIX Conf. Secur. Symp.*, 2017, pp. 1093–1110.
[35] M. Arjovsky, "Wasserstein GAN," 2017, *arXiv:1701.07875*. [Online]. Available: https://arxiv.org/abs/1701.07875
[36] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," 2017, *arXiv:1708.06131*. [Online]. Available: http://arxiv.org/abs/1708.06131
[37] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," 2016, *arXiv:1608.04644*. [Online]. Available: http://arxiv.org/abs/1608.04644
[38] L. Chen, Y. Ye, and T. Bourlai, "Adversarial machine learning in malware detection: Arms race between evasion attack and defense," in *Proc. Eur. Intell. Secur. Informat. Conf. (EISIC)*, Sep. 2017, pp. 99–106.
[39] P. Y. Chen, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," 2017, *arXiv:1708.03999*. [Online]. Available: https://arxiv.org/abs/1708.03999
[40] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "EAD: Elastic-net attacks to deep neural networks via adversarial examples," 2017, *arXiv:1709.04114*. [Online]. Available: http://arxiv.org/abs/1709.04114
[41] K. Grosse, "Adversarial examples for malware detection," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2017, pp. 62–79.
[42] H. Hosseini, "Blocking transferability of adversarial examples in black-box learning systems," 2017, *arXiv:1703.04318*. [Online]. Available: https://arxiv.org/abs/1703.04318
[43] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," 2017, *arXiv:1702.05983*. [Online]. Available: https://arxiv.org/abs/1702.05983
[44] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*. [Online]. Available: http://arxiv.org/abs/1706.06083
[45] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," 2017, *arXiv:1705.09064*. [Online]. Available: http://arxiv.org/abs/1705.09064
[46] M. Rigaki, "Adversarial deep learning against intrusion detection classifiers," M.S. thesis, Inf. Secur., Luleå Univ. Technol., Luleå, Sweden, 2017.
[47] S. Sankaranarayanan, A. Jain, R. Chellappa, and S. Nam Lim, "Regularizing deep networks using efficient layerwise adversarial training," 2017, *arXiv:1705.07819*. [Online]. Available: http://arxiv.org/abs/1705.07819
[48] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," 2017, *arXiv:1704.01155*. [Online]. Available: http://arxiv.org/abs/1704.01155
[49] H. S. Anderson, J. Woodbridge, and B. Filar, "DeepDGA: Adversarially-tuned domain generation and detection," in *Proc. ACM Workshop Artif. Intell. Secur.*, 2016, pp. 13–21.
[50] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware c&c detection: A survey," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 59:1–59:39, 2016.
[51] F. Haddadi, D.-T. Phan, and A. N. Zincir-Heywood, "How to choose from different botnet detection systems?" in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2016, pp. 1079–1084.
[52] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," 2015, *arXiv:1511.04508*. [Online]. Available: http://arxiv.org/abs/1511.04508
[53] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," 2016, *arXiv:1611.03814*. [Online]. Available: http://arxiv.org/abs/1611.03814
[54] N. Papernot, "Practical black-box attacks against machine learning," 2016, *arXiv:1602.02697*. [Online]. Available: https://arxiv.org/abs/1602.02697
[55] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS P)*, Mar. 2016, pp. 372–387.
[56] T. Salimans, "Improved techniques for training GANs," 2016, *arXiv:1606.03498*. [Online]. Available: https://arxiv.org/abs/1606.03498
[57] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on PDF malware classifiers," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2016.
[58] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*. [Online]. Available: http://arxiv.org/abs/1412.6572
[59] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," 2015, *arXiv:1511.04599*. [Online]. Available: http://arxiv.org/abs/1511.04599
[60] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[61] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "DREBIN: Effective and explainable detection of Android malware in your pocket," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 23–26.
[62] I. Goodfellow, "NIPS 2016 Tutorial: Generative adversarial networks," 2014, *arXiv:1701.00160*. [Online]. Available: https://arxiv.org/abs/1701.00160
[63] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*. [Online]. Available: http://arxiv.org/abs/1312.6199
[64] M. Zamani and M. Movahedi, "Machine learning techniques for intrusion detection," 2013, *arXiv:1312.2177*. [Online]. Available: https://arxiv.org/abs/1312.2177
[65] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proc. ICML Workshop Unsupervised Transf. Learn.*, 2012, pp. 37–49.
[66] D. Maiorca, "A pattern recognition system for malicious PDF files detection," in *Proc. Mach. Learn. Data Mining Pattern Recognit.*, 2012, pp. 510–524.
[67] L Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proc. 4th ACM Workshop Secur. Artif. Intell.*, 2011, pp. 43–58.
[68] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.
[69] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[70] P. Vinod, R. Jaipur, V. Laxmi, and M. Gaur, "Survey on malware detection methods," in *Proc. 3rd Hackers' Workshop Comput. Internet Secur.*, 2009, pp. 74–79.
[71] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, Aug. 2005.
[72] R. A. Kemmerer, "Cybersecurity," in *Proc. 25th Int. Conf. Softw. Eng.*, 2003, pp. 705–715.



NUNO MARTINS received the integrated master's degree in computer engineering from the Faculty of Engineering, University of Porto (FEUP). His interests include areas such as cybersecurity, machine learning, and adversarial machine learning.

JOSÉ MAGALHÃES CRUZ was born in Portugal, in 1960. He received the Graduated degree in physics from the Faculty of Sciences, University of Porto (branch of optics and applied electronics), in 1983, and the Ph.D. degree in electrical and computer engineering from FEUP, in 1999. He is currently an Assistant Professor with the Department of Informatics Engineering, Faculty of Engineering, University of Porto (FEUP). He taught multiple courses in the areas of physics, electronics and informatics throughout his academic career. For several years, he has also focused in the areas of operating systems, distributed systems and computer security, having participated in several projects and supervised multiple master's theses.

TIAGO CRUZ received the Ph.D. degree in computer science from University of Coimbra, in 2012. He has been an Auxiliary Professor with the Department of Informatics Engineering, University of Coimbra, since December 2013. He is currently a Senior Researcher with the Center for Informatics and Systems of UC, having started his research activity, in 2001. His research interests include areas such as management systems for communications infrastructure and services (operator and data center environments), embedded computing, critical infrastructure security, broadband access network device, and service management. He has been involved in various European- and industry-funded research projects related with cyber-security, autonomous management, content delivery infrastructures or wireless WAN technologies (CockpitCI FP7, WEIRD FP7, ACROSS COST, CONTENT FP6), as well as several national projects, in partnership with telecommunications operators and research agencies, with technical and management activities.

PEDRO HENRIQUES ABREU received the Informatics Engineering degree and the Ph.D. degree in soccer team modeling from the University of Porto, in 2006 and 2011, respectively. He is currently an Assistant Professor with the Department of Informatics Engineering, University of Coimbra. He has authored more than 60 publications in international conferences and journals. His interests include medical informatics and personal healthcare systems applied to oncology.