Adversarial Machine Learning Attacks and Defense Methods in the Cyber Security Domain

ISHAI ROSENBERG, ASAF SHABTAI, YUVAL ELOVICI, and LIOR ROKACH,
Ben-Gurion University of the Negev

In recent years, machine learning algorithms, and more specifically deep learning algorithms, have been widely used in many fields, including cyber security. However, machine learning systems are vulnerable to adversarial attacks, and this limits the application of machine learning, especially in non-stationary, adversarial environments, such as the cyber security domain, where actual adversaries (e.g., malware developers) exist. This article comprehensively summarizes the latest research on adversarial attacks against security solutions based on machine learning techniques and illuminates the risks they pose. First, the adversarial attack methods are characterized based on their stage of occurrence, and the attacker' s goals and capabilities. Then, we categorize the applications of adversarial attack and defense methods in the cyber security domain. Finally, we highlight some characteristics identified in recent research and discuss the impact of recent advancements in other adversarial learning domains on future research directions in the cyber security domain. To the best of our knowledge, this work is the first to discuss the unique challenges of implementing end-to-end adversarial attacks in the cyber security domain, map them in a unified taxonomy, and use the taxonomy to highlight future research directions.

CCS Concepts: • **Computing methodologies** → *Machine learning*; *Learning paradigms*; *Reinforcement learning*; *Adversarial learning*; • **Security and privacy** → *Intrusion/anomaly detection and malware mitigation*; • **Computing methodologies** → *Machine learning approaches*;

Additional Key Words and Phrases: Adversarial learning, adversarial machine learning, evasion attacks, poisoning attacks, deep learning, adversarial examples, cyber security

ACM Reference format:
Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. 2021. Adversarial Machine Learning Attacks and Defense Methods in the Cyber Security Domain. *ACM Comput. Surv.* 54, 5, Article 108 (May 2021), 36 pages.
https://doi.org/10.1145/3453158

1 INTRODUCTION

The growing use of machine learning, and particularly deep learning, in fields like computer vision and **natural language processing (NLP)** has been accompanied by increased interest in the domain of adversarial machine learning (a.k.a. adversarial learning)—that is, attacking and defending machine learning models algorithmically (Huang et al. [50]). Of special interest are adversarial examples, which are samples modified to be misclassified by the classifier attacked.

Authors' address: I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, Ben-Gurion University of the Negev; emails: ishairos@post.bgu.ac.il, elovici@bgu.ac.il, elovici@bgu.ac.il, liorrk@post.bgu.ac.il.

0360-0300/2021/05-ART108 $15.00
https://doi.org/10.1145/3453158

Most of the research in adversarial learning has focused on the computer vision domain, and more specifically in the image recognition domain. This research has centered mainly on **convolutional neural networks (CNNs)**, which are commonly used in the computer vision domain (Akhtar and Mian [3], Qiu et al. [79]). However, in recent years, adversarial example generation methods have increasingly been utilized in other domains, including NLP (e.g., Gao et al. [34]). These attacks have also been used recently in the cyber security domain (e.g., Rosenberg et al. [89]). This domain is particularly interesting because it is rife with adversaries (e.g., malware developers who want to evade machine and deep learning based **next generation antivirus (NGAV)** products and spam filters). Adversarial learning methods have already been executed against static analysis **deep neural networks (DNNs).**[1]

The main goal of this work is to illuminate the risks posed by adversarial learning to cyber security solutions that are based on machine learning techniques. This article contains (1) an in-depth discussion about the unique challenges of adversarial learning in the cyber security domain (Section 2); (2) a summary of state-of-the-art adversarial learning research papers in the cyber security domain, categorized by application (Sections 4 and 6.6) and characterized by our unified taxonomy (defined in Section 3); (3) a discussion of the challenges associated with adversarial learning in the cyber security domain and possible future research directions (Section 6), including issues relating to existing defense methods (and the lack thereof); and (4) a summary of the theoretical background on the adversarial methods used in the cyber security domain (Appendix B in the supplemental materials). We focus on adversarial attacks and defenses against classifiers used in the cyber security domain, and not on other topics, such as attacks on models' interpretability Kuppa and Le-Khac [63] or methods to assist the model's interpretability Ross and Doshi-Velez [90].

The main contributions of this work are as follows:

1. We focus on a wide range of adversarial learning applications in the cyber security domain (e.g., malware detection, speaker recognition, **cyber-physical systems (CPSs)**), introduce a new unified taxonomy, and illustrate how existing research fits into this taxonomy, providing a holistic overview of the field. In contrast, previous work focused mainly on specific domains, such as malware detection or network intrusion detection.

2. Using our taxonomy, we highlight research gaps in the cyber security domain that have already been addressed in other adversarial learning domains (e.g., Trojan neural networks in the image recognition domain) and discuss their impact on current and future trends in adversarial learning in the cyber security domain.

3. We discuss the unique challenges that attackers and defenders face in the cyber security domain, challenges which do not exist in other domains (e.g., image recognition). For instance, in the cyber security domain, the attacker must verify that the original functionality of the modified malware remains intact. Our discussion addresses the fundamental differences between adversarial attacks performed in the cyber security domain and those performed in other domains.

2 PRELIMINARY DISCUSSION: THE DIFFERENCES BETWEEN ADVERSARIAL ATTACKS IN THE COMPUTER VISION AND CYBER SECURITY DOMAINS

Most adversarial attacks published, including those published at academic cyber security conferences, have focused on the computer vision domain (e.g., generating a cat image that would be classified as a dog by the classifier). However, the cyber security domain (e.g., malware detection) seems to be a more relevant domain for adversarial attacks because in the computer vision domain, there is no real adversary (with a few exceptions, e.g., terrorists who want to tamper with autonomous cars' pedestrian detection systems (Eykholt et al. [32]), deepfakes (Liu et al. [70]) that

[1] https://skylightcyber.com/2019/07/18/cylance-i-kill-you/.

might cause fake news or financial fraud, etc.). In contrast, in the cyber security domain, there are actual adversaries with clear targeted goals. Examples include ransomware developers who depend on the ability of their ransomware to evade anti-malware products that would prevent both its execution and the developers from collecting the ransom money, and other types of malware that need to steal user information (e.g., keyloggers), spread across the network (worms), or perform any other malicious functionality while remaining undetected.

A key step in defining the proper taxonomy for the cyber security domain is answering this question: Given the obvious relevance of the cyber security domain to adversarial attacks, why do most adversarial learning researchers focus on computer vision? In addition to the fact that image recognition is a popular machine learning research topic, another major reason for the focus on computer vision is that performing an end-to-end adversarial attack in the cyber security domain is more difficult than performing such an attack in the computer vision domain. The differences between adversarial attacks performed in those two domains and the challenges that arise in the cyber security domain are discussed in the sections that follow.

2.1 Keeping (Malicious) Functionality Intact in the Perturbed Sample

Any adversarial executable file must preserve its malicious functionality after the sample' s modification. This might be the main difference between the image classification and malware detection domains, and pose the greatest challenge. In the image recognition domain, the adversary can change every pixel' s color (to a different valid color) without creating an "invalid picture" as part of the attack. However, in the cyber security domain, modifying an API call or arbitrary executable' s content byte value might cause the modified executable to perform a different functionality (e.g., changing a *WriteFile()* call to *ReadFile()*) or even crash (if you change an arbitrary byte in an opcode to an invalid opcode that would cause an exception). The same is true for network packets; perturbing a network packet to evade the **network intrusion detection system (NIDS)** while keeping a valid packet structure is challenging.

To address this challenge, adversaries in the cyber security domain must implement their own methods (which are usually feature specific) to modify features in a way that will not break the functionality of the perturbed sample, whether it is an executable, a network packet, or something else. For instance, the adversarial attack used in Rosenberg et al. [89] generates a new malware **portable executable (PE)** with a modified API call trace in a functionality-preserving manner.

2.2 Small Perturbations Are Not Applicable for Discrete Features

In the computer vision domain, gradient-based adversarial attacks, such as the **fast gradient sign method (FGSM)** (see Appendix B in the supplemental materials), generate a minimal random modification to the input image in the direction that would most significantly impact the target classifier prediction. A " small modification' (a.k.a. perturbation) can be, for example, changing a single pixel' s color to a very similar color (a single pixel' s color can be changed from brown to black to fool the image classifier).

However, the logic of a "small perturbation" cannot be applied to many cyber security features. Consider a dynamic analysis classifier that uses API calls. An equivalent to changing a single pixel' s color would be to change a single API call to another API call. Even if we disregard what such a modification would do to the executable' s functionality (mentioned in the previous section), would one of the following be considered a small perturbation of the *WriteFile()* API call: (1) modifying it to *ReadFile()* (a different operation for the same medium) or (2) modifying it to *RegSetValueEx()* (a similar operation for a different medium)? The use of discrete features (e.g., API calls) that are not continuous or ordinal severely limits the use of gradient-based attack methods (Appendix B in the supplemental materials). The implications of this issue will be discussed in Section 6.6.1.

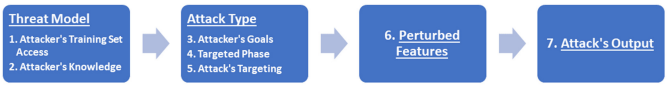

Fig. 1. Chronological overview of the taxonomy.

2.3 Executables Are More Complex Than Images

An image used as input to an image classifier (usually a CNN) is represented as a fixed-size matrix of pixel colors. If the actual image has different dimensions than the input matrix, the picture will usually be resized, clipped, or padded to fit the dimension limits.

An executable, however, has a variable length: executables can range in size from several kilobytes to several gigabytes. It is also unreasonable to expect a clipped executable to maintain its original classification. Let us assume that we have a 100-MB benign executable into which we inject a shellcode in a function near the end-of-file. If the shellcode is clipped to fit the malware classifier' s dimensions, there is no reason that the file would be classified as malicious, because its benign variant would be clipped to the exact same form.

In addition, the execution path of an executable may depend on the input, and thus the adversarial perturbation should support any possible input that the malware may encounter when executed in the target machine.

Finally, in the cyber security domain, classifiers usually use more than a single feature type as input (e.g., phishing detection using both **uniform resource locators (URLs)** and connected server properties as was done in Shirazi et al. [95]). A non-exhaustive list of features used in the cyber security domain is presented later in Figure 2. Some feature types are easier to modify without harming the executable's functionality than others. For instance, in the adversarial attack used in Rosenberg et al. [89], appending printable strings to the end of a malware PE file is much easier than adding API calls to the PE file using a dedicated framework built for this purpose. In contrast, in an image adversarial attack, modifying each pixel has the same level of difficulty. The implications of this issue are discussed in Section 6.6.1.

Although this is a challenge for malware classifier implementation, it also affects adversarial attacks against malware classifiers. For instance, attacks in which you have a fixed input dimension (e.g., a 28*28 matrix for MNIST images) are much easier to implement than attacks for which you need to consider the variable file size.

3 TAXONOMY

Adversarial learning in cyber security is the modeling of non-stationary adversarial settings like spam filtering or malware detection, where a malicious adversary can carefully manipulate (or perturb) the input data, exploiting specific vulnerabilities of learning algorithms to compromise the (targeted) machine learning system's security.

A taxonomy for the adversarial domain in general exists (e.g., Barreno et al. [15]) and inspired our taxonomy. However, the cyber security domain has a few unique challenges, described in the previous section, necessitating a different taxonomy to categorize the existing attacks, with several new parts, such as the attack's output, the attack's targeting, and perturbed features.

Our proposed taxonomy is shown in Figure 1. The attacks are categorized based on seven distinct attack characteristics, which are sorted by four chronological phases of the attack:

(1) *Threat model*: The attacker's knowledge and capabilities, known prior to the attack. The threat model includes the training set access and the attacker's knowledge.

(2) *Attack type*: These characteristics are a part of the attack implementation. The attack type includes the attacker's goals, the targeted phase, and the attack's targeting.

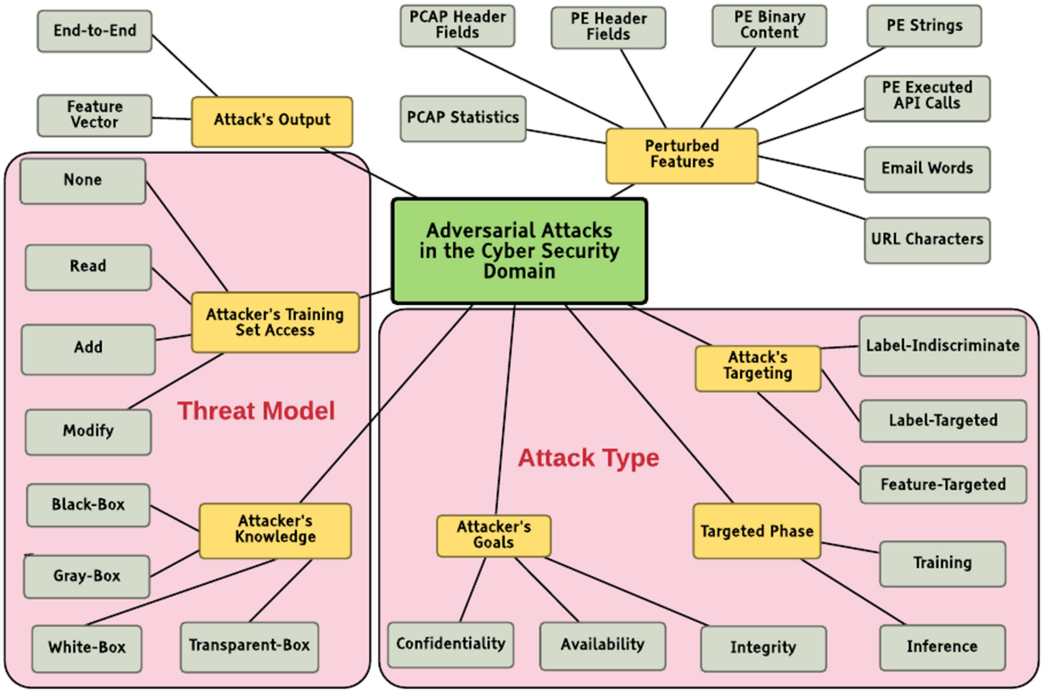

Fig. 2. Detailed overview of the taxonomy.

(3) The features modified (or perturbed) by the attack.

(4) The attack's output.

A more detailed overview of our proposed taxonomy, including possible values for the seven characteristics, is shown in Figure 2. The seven attack characteristics (attacker's goals, attacker's knowledge, attacker's training set access, targeted phase, attack's targeting, perturbed features, and attack's output) are described in the sections that follow.

We include these characteristics in our taxonomy for the following reasons:

(1) These characteristics are specific to the cyber domain (e.g., perturbed features and the attack' s output).

(2) These characteristics are especially relevant to the threat model, which plays a much more critical role in the cyber security domain, where white-box attack are less valuable than in other domains, since the knowledge of adversaries in the cyber security domain about the classifier architecture is usually very limited (e.g., the attacker' s knowledge, the attacker' s training set access, and the targeted phase).

(3) These characteristics highlight missing research in the cyber security domain, which exists in other domains of adversarial learning. Such research is specified in Section 6 (e.g., the attack's targeting).

(4) These characteristics exist in many domains but have a different emphasis (and are therefore more important) in the cyber security domain (e.g., if we analyze the attacker' s goal characteristic, availability attacks are of limited use in other domains, but they are very relevant in the cyber security domain).

3.1 Attacker's Goals

This characteristic of the attack is sometimes considered part of the attack type. An attacker aims to achieve one or more of the following goals (a.k.a. the CIA triad). First, *Confidentiality*—acquire

private information by querying the machine learning system, such as stealing the classifier's model (Tramèr et al. [109]). Second, *Integrity*—cause the machine learning system to perform incorrectly for some or all input (e.g., to cause a machine learning based malware classifier to misclassify a malware sample as benign (Srndic and Laskov [101])). Third, *Availability*—cause a machine learning system to become unavailable or block regular use of the system (i.e., to generate malicious sessions that have many of the features of regular traffic, causing the system to classify legitimate traffic sessions as malicious and block legitimate traffic (Chung and Mok [25])).

3.2 Attacker's Knowledge

This attack characteristic is sometimes considered part of the threat model. Attacks vary based on the amount of knowledge the adversary has about the classifier he/she is trying to subvert. First, *Black-Box attack*—requires no knowledge about the model beyond the ability to query it as a black-box (a.k.a. the oracle model), such as inserting an input and obtaining the output classification. Second, *Gray-Box attack*—requires some (limited) degree of knowledge about the targeted classifier. Although usually this consists of the features monitored by the classifier, sometimes it is other incomplete pieces of information like the output of the hidden layers of the classifier or the confidence score (and not just the class label) provided by the classifier. Third, *White-Box attack*—the adversary has knowledge about the model architecture and even the hyperparameters used to train the model. Fourth, *Transparent-Box attack*—in this case, the adversary has complete knowledge about the system, including both white-box knowledge and knowledge about the defense methods used by defender (see Section 6.6). Such knowledge can assist the attacker in choosing an adaptive attack that would be capable of bypassing the specific defense mechanism (e.g., Tramer et al. [108]).

Although white-box attacks tend to be more efficient than black-box attacks (sometimes by an order of magnitude (Rosenberg and Gudes [84]), the knowledge required is rarely available in real-world use cases. However, white-box knowledge can be gained either through internal knowledge or by using a staged attack to reverse engineer the model beforehand (Tramèr et al. [109]). Each type of attack (black-box, gray-box, etc.) has a query-efficient variant in which the adversary has only a limited number of queries (in each query, the adversary inserts input into the classifier and obtains its classification label), and not an unlimited amount of queries, as in the variants mentioned earlier. A query-efficient variant is relevant in the case of cloud security services (e.g., Rosenberg et al. [87]). In such services, the attacker pays for every query of the target classifier and therefore aims to minimize the number of queries made to the cloud service when performing an attack. Another reason for minimizing the number of queries is that many queries from the same computer might arouse suspicion of an adversarial attack attempt, causing the cloud service to stop responding to those queries. Such cases require query-efficient attacks.

3.3 Attacker's Training Set Access

Another important characteristic of an attack, sometimes considered part of the threat model, is the access the adversary has to the training set used by the classifier (as opposed to access to the model itself, mentioned in the previous section). The attacker' s training set access is categorized as follows: (1) None (no access to the training set), (2)*Read* data from the training set (entirely or partially), (3)*Add* new samples to the training set, and (4) *Modify* existing samples (modifying either all features or just specific features, e.g., the label). For instance, poisoning attacks require *add* or *modify* permissions.

3.4 Targeted Phase

This attack characteristic is sometimes considered part of the attack type. Adversarial attacks against machine learning systems occur in two main phases of the machine learning process. First, *Training Phase attack*—this attack aims to introduce vulnerabilities (to be exploited in the classification phase) by manipulating training data during the training phase. For instance, a *poisoning attack* can be performed by inserting crafted malicious samples labeled as benign to the training set as part of the baseline training phase of a classifier. Second, *Inference Phase attack*—this attack aims to find and subsequently exploit vulnerabilities in the classification phase. In this phase, the attacker modifies only samples from the test set. For example, an *evasion attack* involves modifying the analyzed malicious sample's features to evade detection by the model. Such inputs are called *adversarial examples*.

Note that attacks on online learning systems (e.g., anomaly detection systems (Clements et al. [27])) combine both training phase and inference phase attacks: the attack is an evasion attack, but if it succeeds, the classifier learns that this traffic is legitimate, making additional such attacks harder to detect by the system (i.e., there is a poisoning effect). Such attacks would be termed *inference attacks* in this article, since in this case, the poisoning aspect is usually a by-product and is not the attacker's main goal. Moreover, even if the poisoning aspect is important to the attacker, it would usually be successful only if the evasion attack works, so evasion is the primary goal of the attacker in any case.

3.5 Attack's Targeting

This characteristic is sometimes considered part of the attack type. Each attack has a different targeting, defining the trigger conditions or the desired effect on the classifier. First, *Label-Indiscriminate attack*—always minimizes the probability of correctly classifying a perturbed sample (the adversarial example). Second, *Label-Targeted attack*—always maximizes the probability of a specific class to be predicted for the adversarial example (different from the predicted class for the unperturbed sample). Third, *Feature-Targeted attack*—the malicious behavior of these attacks is only activated by inputs stamped with an attack trigger, which might be the existence of a specific input feature or group of feature values in the adversarial example.

Attacks can be both feature and label targeted. Note that in the cyber security domain, many classifiers are binary (i.e., they have two output classes: malicious and benign, spam and ham, anomalous or not). For binary classifiers, label-indiscriminate and label-targeted attacks are the same, because in these cases, minimizing the probability of the current class (label-indiscriminate attack) is equivalent to maximizing the probability of the only other possible class.

3.6 Perturbed Features

As mentioned in Section 2.3, in the cyber security domain, classifiers and other machine learning systems often use more than one feature type. Thus, attackers who want to subvert those systems should consider modifying more than a single feature type. We can therefore characterize the different adversarial attacks in the cyber security domain by the features being modified/perturbed or added. Note that the same feature type might be modified differently depending on the sample's format. For instance, modifying a printable string inside a PE file might be more challenging than modifying a word within the content of an email content, although the feature type is the same. Thus, this classification is not simply a feature type but a tuple of feature type and sample format (i.e., printable strings inside a PE file). The following is a *partial* list (e.g., the work of Rosenberg et al. [86] contains 2,381 features, so the full list cannot be included) of such tuples used in the papers reviewed in our research: PCAP (packet capture; part of a network session) statistical

features (e.g., number of SYN requests in a certain time window), PCAP header (e.g., IP or UDP) fields, PE header fields, printable strings inside a PE file, binary bytes inside a PE file, PE executed API calls (during a dynamic analysis of the PE file), and words inside an email or characters inside a URL.

3.7 Attack's Output

As discussed in Section 2.1, in contrast to image-based attacks, most adversarial attacks in the cyber domain require the modification of a feature's values. Whereas in some domains, such as spam detection, modifying a word in an email is non-destructive, modifying, for example, a field in a PE header metadata might result in an unrunnable PE file. Thus, there are two type of attacks. First, *Feature Vector attack*—such attacks obtain a feature vector as an input and output another perturbed feature vector. However, such an attack does not generate a sample tha can be used by the attacker and is usually only a hypothetical attack that would not be possible in real life. Second, *End-to-End attack*—this attack generates a functional sample as an output. Thus, this is a concrete real-life attack. This category is further divided into many subgroups based on the sample type produced, such as a valid and runnable PE file, a phishing URL, or a spam email.

For instance, most traffic anomaly detection attacks reviewed in this work are feature vector attacks. They use statistical features aggregating packet metadata, but the authors do not show how to generate the perturbed packet. In contrast, the attack used by Rosenberg et al. [89] to add API calls to a malicious process uses a custom framework that generates a new binary that adds those API calls. Thus, this is an end-to-end attack. In some image-based domains, such as face recognition systems (Section 4.6.1), end-to-end attacks can be further categorized as those that generate images (e.g., Liu et al. [69]) and those that generate physical elements that can be used to generate multiple relevant images (e.g., Sharif et al. [94]).

4 ADVERSARIAL ATTACKS IN THE CYBER SECURITY DOMAIN

Our article addresses adversarial attacks in the cyber security domain. An overview of this section is provided in Tables 1 through 7. Target classifier abbreviations are specified in Appendix A in the supplemental materials. The attack type includes the attacker's goals, the targeted phase, and the attack's targeting. The threat model includes the attacker's knowledge and training set access. Unless mentioned otherwise, a gray-box attack requires knowledge of the target classifier's features, the attack's targeting is label indiscriminate, and the attacker's training set access is *none*. Some of the columns are not a part of our taxonomy (Section 3) but provide additional relevant information that may be helpful for understanding the attacks, such as the target classifiers.

Each of the following sections represents a specific cyber security domain that uses adversarial learning and discusses the adversarial learning methods used in this domain. Although there are other domains in cyber security, we focused only on domains in which substantial adversarial learning research has been performed. Due to space limits, this review article covers only the state of the art in the preceding areas and not all adversarial attacks, especially in large and diverse domains, such as biometric systems or CPSs. The strengths and weaknesses of each adversarial attack are analyzed throughout this section.

For the reader's convenience, we have summarized the analysis in Tables 1 through 7 using the following Boolean parameters:

(1) *Reasonable attacker knowledge*: Is the attack a gray-box or black-box attack, both of which require a reasonable amount of knowledge (+ value in Tables 1–7) or a white-box attack, which requires an unreasonable amount of knowledge (– value in Tables 1–7) in the cyber security domain?

Table 1. Summary of Adversarial Learning Approaches in Malware Detection (Part 1)

Citation	Year	Target Classifier	Attack Type	Attack's Output	Threat Model	Perturbed Features	**Reasonable attacker knowledge?**	**End-to-end attack?**	**Effective attack?**	**Representative dataset?**	**Representative features?**
Li et al. [66], Srndic and Laskov [101]	2020	RF	Inference integrity	PDF file (end-to-end)	Gray-box	Static structural PDF features	+	+	+	–	+
Ming et al. [71]	2015	SCDG	Inference integrity	PE file (end-to-end)	Gray-box	Executed API calls	+	+	+	–	–
Suciu et al. [105]	2018	SVM	Training integrity	Feature vector	Gray-box; *add* training set access	Static Android manifest features	–	–	+	–	+
Dang et al. [28]	2017	SVM, RF	Inference integrity	PDF file (end-to-end)	Query-efficient gray-box	Static structural PDF features	+	+	+	–	+
Anderson et al. [7]	2018	GBDT	Inference integrity	PE file (end-to-end)	Black-box	Operations (e.g., packing) performed on a PE file	+	+	–	+	+
Grosse et al. [41]	2017	FC DNN	Inference integrity	Feature vector	White-box	Static Android manifest features	–	–	+	–	+
Xu et al. [119]	2020	SCDG	Inference integrity	Feature vector	Gray-box	Static Android manifest features	+	–	+	–	–
Kolosnjaji et al. [57], Kreuk et al. [60]	2018	1D CNN	Inference integrity	PE file (end-to-end)	White-box	PE file's raw bytes	–	+	+	+	+
Suciu et al. [104]	2018	1D CNN	Inference integrity	PE file (end-to-end)	Black-box	PE file's raw bytes	+	+	+	+	+
Rosenberg and Meir [85], Rosenberg et al. [86]	2020	GBDT, FC DNN	Inference integrity	PE file (end-to-end)	Black-box	PE header metadata	+	+	+	+	+

(2) *End-to-end attack*: Does the attack have an end-to-end attack output (defined in Section 3.7)? Such attacks are considered more feasible attacks (+ value in Tables 1–7), whereas feature vector attacks (defined in Section 3.7) are considered less feasible (– value in Tables 1–7).

(3) *Effective attack*: Is the attack effective (success rate greater than 90%) for the attack use case (+ value in Tables 1–7) or ineffective (success rate lower than 90%; – value in Tables 1–7)?

(4) *Representative dataset*: Is the dataset used representative of the relevant threats in the wild (+ value in Tables 1–7), or is it just a subset (or old variants) of those threats (– value in Tables 1–7)?

(5) *Representative features*: Are the features used in the classifiers being attacked similar to those being used in real-life security products (+ value in Tables 1–7) or not (– value in Tables 1–7)?

The mathematical background for the deep learning classifiers is provided in Appendix A in the supplemental materials, and the mathematical background for the commonly used adversarial learning attacks in the cyber security domain is provided in Appendix B in the supplemental materials.

Note that although the classifiers the attacker tries to subvert are mentioned briefly to provide context helpful for understanding the attack, a complete list of the state-of-the-art prior work is not provided due to space limits. A more comprehensive list can be found, for example, in Berman et al. [17]. Cases in which an adversarial attack does not exist for a specific application type are omitted. This manuscript also does not review adversarial attacks in non-cyber domains, such as image recognition (with the exception of the face recognition domain that is addressed in Section 4.6.1, which is cyber related). It also does not cover cyber security-related papers that are not related to adversarial learning, such as the use of machine learning to bypass CAPTCHA.

4.1 Malware Detection and Classification

NGAV products, such as Cylance, CrowdStrike, SentinelOne, and Microsoft ATP, use machine and deep learning models instead of signatures and heuristics, allowing them to detect unseen and unsigned malware but also leaving them open to attacks against such models.

Malware classifiers can either use static features gathered without running the code (e.g., n-gram byte sequence, strings, or structural features of the inspected code) or dynamic features (e.g., CPU usage) collected during the inspected code execution.

Although using static analysis provides a performance advantage, it has a major disadvantage: since the code is not executed, the analyzed code might not reveal its "true features." For example, when looking for specific strings in the file, one might not be able to catch polymorphic malware, in which those features are either encrypted or packed, and decrypted only during runtime by a specific bootstrap code. Fileless attacks (code injection, process hollowing, etc.) are also a problem for static analysis. Thus, dynamic features, extracted at runtime, can be used. The most prominent dynamic features that can be collected during malware execution are the sequences of API calls (Kolbitsch et al. [56]), particularly those made to the OS, which are termed *system calls*. Those system calls characterize the software behavior and are harder to obfuscate during execution time without harming the functionality of the code. The machine learning techniques (and thus the attacks of them) can be divided into two groups: traditional (or shallow) machine learning and deep learning techniques. A summary of the attacks in the malware detection sub-domain is shown in Tables 1 and 2.

4.1.1 Attacking Traditional (Shallow) Machine Learning Malware Classifiers. Srndic and Laskov [101] implemented an inference integrity gray-box evasion attack against PDFRATE, a random forest classifier for static analysis of malicious PDF files, using PDF structural features, such as the number of embedded images or binary streams within the PDF. The attack used either a mimicry attack in which features were added to the malicious PDF to make it "feature-wise similar" to a

Table 2. Summary of Adversarial Learning Approaches in Malware Detection (Part 2)

Citation	Year	Target Classifier	Attack Type	Attack's Output	Threat Model	Perturbed Features	Reasonable attacker knowledge?	End-to-end attack?	Effective attack?	Representative dataset?	Representative features?
Hu and Tan [47]	2017	RF, LR, DT, SVM, MLP	Inference integrity	Feature vector	Gray-box	API calls' unigrams	+	–	+	–	–
Xu et al. [121]	2016	SVM, RF, CNN	Inference integrity	PDF file (end-to-end)	Gray-box	Static structural PDF features	+	+	+	–	+
Liu et al. [68]	2019	FC DNN, LR, DT, RF	Inference integrity	Feature vector	Gray-box	Static Android manifest features	+	–	+	–	+
Abusnaina et al. [2]	2019	CNN	Inference integrity	Feature vector	White-box	CFG features	–	–	+	–	–
Hu and Tan [46]	2017	LSTM	Inference integrity	Feature vector	Gray-box	Executed API calls	+	–	+	–	+
Rosenberg et al. [89]	2018	LSTM, GRU, FC DNN, 1D CNN, RF, SVM, LR, GBDT	Inference integrity	PE file (end-to-end)	Gray-box	Executed API calls, printable strings	+	+	+	+	+
Rosenberg et al. [87]	2018	LSTM, GRU, FC DNN, 1D CNN, RF, SVM, LR, GBDT	Inference integrity	PE file (end-to-end)	Query-efficient gray-box	Executed API calls, printable strings	+	+	+	+	+

benign sample, or created an SVM representation of the classifier and subverted it using a method that follows the gradient of the weighted sum of the classifier' s decision function and the estimated density function of benign examples. This ensures that the final result lies close to the region populated by real benign examples. The density function must be estimated beforehand, using the standard technique of kernel density estimation, and then the transferability property is used to attack the original PDFRATE classifier using the same PDF file. Li et al. [66] performed an inference integrity gray-box attack against the same classifier by using **generative adversarial network (GAN)**-generated feature vectors and transforming them back into PDF files. The main advantage of these attacks is the fact that they are end-to-end attacks that produce a valid (malicious) PDF file, which evade detection. The main problem is that very few malware have PDF file type. PE files, which are more common, were not covered in this work.

Ming et al. [71] used an inference integrity replacement attack, replacing API calls with different functionality-preserving API subsequences (so gray-box knowledge about the monitored

APIs is required) to modify the malware code. They utilized a **system-call dependence graph (SCDG)** with the graph edit distance and Jaccard index as clustering parameters of different malware variants and used several SCDG transformations on their malware source code to move it to a different cluster. Their transformations can cause similar malware variants to be classified as a different cluster, but they did not show that the attack can cause malware to be classified (or clustered) as a benign program, which is usually the attacker's main goal. Xu et al. [119] also implemented an inference integrity gray-box attack against an SCDG-based APK malware classifier, using n-strongest nodes and FGSM (see Appendix B in the supplemental materials) methods. The main advantage of these attacks is the fact that they are end-to-end attacks that produce a valid (malicious) binary (PE or APK, respectively) file, which evade detection. The main problem is that the features used (SCDG) are not used by real-life NGAV products.

Suciu et al. [105] and Chen et al. [23] used a training integrity poisoning attack against a linear SVM classifier trained on the Drebin dataset (Arp et al. [11]) for Android malware detection. This attack requires gray-box knowledge of the classifier's features and training set *add* access. The poisoning was done by adding static features (permissions, API calls, URL requests) from the target to existing benign instances. Muñoz González et al. [73] used a training integrity poisoning attack against logistic regression, MLP, and ADALINE classifiers, for spam and ransomware detection, by using back-gradient optimization. This attack requires gray-box knowledge of the classifier's features and training set *add* and *read* access. A substitute model is built and poisoned, and the poisoned samples are effective against the target classifier as well, due to the transferability property. The main problem with these poisoning attacks is that they require a powerful attacker who is able to inject samples into the training set of the malware classifier. Although such a scenario is possible in some cases (e.g., supply chain attacks), this is usually not a common case, making such attacks less feasible.

Dang et al. [28] utilized the rate of feature modifications from a malicious sample and a benign known sample as the score and used a hillclimbing approach to minimize this score, bypassing SVM and random forest PDF malware classifiers based on static features in a query-efficient manner. Thus, their inference integrity attack is a query-efficient gray-box attack. This attack was the first attempt to perform a query-efficient attack in the cyber security domain. However, the classifiers bypassed were not state-of-the-art deep classifiers.

In Anderson et al. [7, 8], the features used by the gradient boosted decision tree classifier included PE header metadata, section metadata, and import/export table metadata. In their work [7, 8], inference integrity black-box attack that trains a **reinforcement learning (RL)** agent was presented. The agent is equipped with a set of operations (e.g., packing) that it may perform on the PE file. The reward function was the evasion rate. Through a series of games played against the target classifier, the agent learns which sequences of operations are likely to result in detection evasion for any given malware sample. The perturbed samples that bypassed the classifier were uploaded to VirusTotal and scanned by 65 anti-malware products. Those samples were detected as malicious by 50% of anti-malware products that detected the original unperturbed samples. This means that this attack works against real-world security products (although the authors did not mention which ones were affected). However, unlike other attacks, this attack' s effectiveness is less than 25% (as opposed to 90% for most other adversarial attacks), showing that additional research is needed for this approach to be practical in real-life use cases.

4.1.2 Attacking DNN Malware Classifiers. Rosenberg et al. [85, 86] used the EMBER dataset and PE structural features (see Table 1 in the supplemental materials) to train a substitute FC DNN model and used explainability machine learning algorithms (e.g., integrated gradients) to detect which of the 2,381 features have high impact on the malware classification and can also be modified

without harming the executable's functionality (e.g., file timestamp). These features were modified in a gray-box inference integrity attack, and the mutated malware bypassed not only the substitute model but also the target GBDT classifier, which used a different subset of samples and features. The main advantage of these attacks are that they bypassed an actual, targeted, real-life NGAV. The main limitation of this attack is that it is not fully automatic—human intervention is required to select the features that are perturbed.

Grosse et al. [40, 41] presented a white-box inference integrity attack against an Android static analysis fully connected DNN malware classifier. The static features used in the DREBIN dataset (see Table 1 in the supplemental materials) were from the AndroidManifest.xml file and included permissions, suspicious API calls, and activities, among others. The attack is a discrete FGSM (see Appendix B in the supplemental materials) variant, which is performed iteratively in the following two steps until a benign classification is made: (1) compute the gradient of the white-box model with respect to the binary feature vector $\boldsymbol{x}$, and (2) find the element in $\boldsymbol{x}$ whose modification from 0 to 1 (i.e., only feature addition and not removal) would cause the maximum change in the benign score and add this feature to the adversarial example. The main advantage of this attack is that it provides a methodical way of dealing with discrete features, commonly used in the cyber security domain, and evaluates this attack against many DNN architectures. The main problem is that the white-box assumption is unrealistic in many real-life scenarios.

Kreuk et al. [60] implemented an inference integrity attack against MalConv, a 1D CNN, using the file's raw byte content as features (Raff et al. [80]). The additional bytes are selected by the FGSM method (see Appendix B in the supplemental materials) and are inserted between the file's sections. Kolosnjaji et al. [57] implemented a similar attack and also analyzed the bytes that are the most impactful features (and are therefore added by the attack), showing that a large portion of them are part of the PE header metadata. Suciu et al. [104] transformed this white-box gradient-based attack to a black-box decision-based attack by appending bytes from the beginning of benign files, especially from their PE headers, which, as shown in Kolosnjaji et al. [57], are prominent features. The main insight of these attacks is that even classifiers that use raw memory bytes as features (leveraging deep architecture's representation learning) are vulnerable to adversarial examples. The main disadvantage is that such classifiers are rarely used by real-life NGAVs.

Hu and Tan [47] perturbed static API call unigrams by performing a gray-box inference integrity attack. If n API types are used, the feature vector dimension is n. A GAN (Appendix A in the supplemental materials) was trained, where the discriminator simulates the malware classifier while the generator tries to generate adversarial samples that would be classified as benign by the discriminator, which uses labels from the black-box model (a random forest, logistic regression, decision tree, linear SVM, MLP, or an ensemble of all of these). However, this is a feature vector attack: the way to generate an executable with the perturbed API call trace was not presented, making this attack infeasible in real life.

Xu et al. [121] generated adversarial examples that bypass PDF malware classifiers by modifying static PDF features. This was done using an inference integrity genetic algorithm, where the fitness of the genetic variants is defined in terms of the target classifier' s confidence score. The genetic algorithm is computationally expensive and was evaluated against SVM, random forest, and CNN classifiers using static PDF structural features. This gray-box attack requires knowledge of both the classifier's features and the target classifier's confidence score. Liu et al. [68] used the same approach to bypass an IoT Android malware detector. The bypassed fully connected DNN, logistic regression, decision tree, and random forest classifiers were trained using the DREBIN dataset.

Abusnaina et al. [2] trained an IoT malware detection CNN classifier using graph-based features (e.g., shortest path, density, number of edges and nodes) from the **control flow graph (CFG)** of the malware disassembly. They used white-box attacks: C&W, DeepFool, FGSM, JSMA (see

Appendix B in the supplemental materials), the **momentum iterative method (MIM)**, **projected gradient descent (PGD)**, and **virtual adversarial method (VAM)**. They also added their own attack, graph embedding, and augmentation, which adds a CFG of a benign sample to the CFG of a malicious sample via source code concatenation. The problem with this attack is that CFG takes a long time to generate, and therefore graph-based features are rarely used by real-life malware classifiers.

Hu and Tan [46] proposed a gray-box inference integrity attack using a **recurrent neural network (RNN)** GAN to generate invalid APIs and inserted them into the original API sequences to bypass an LSTM classifier trained on the API call trace of the malware. A substitute RNN is trained to fit the targeted RNN. Gumbel-Softmax, a one-hot continuous distribution estimator, was used to smooth the API symbols and deliver gradient information between the generative RNN and the substitute RNN. Null APIs were added, but although they were omitted to make the adversarial sequence generated shorter, they remained in the loss function's gradient calculation. This decreases the attack's effectiveness, since the substitute model is used to classify the Gumbel-Softmax output, including the null APIs' estimated gradients, so it does not simulate the malware classifier exactly. The gray-box attack output is a feature vector of the API call sequence that might harm the malware's functionality (e.g., by inserting the *ExitProcess()* API call in the middle of the malware code), making this attack infeasible in real-life scenarios.

Rosenberg et al. [89] presented a gray-box inference integrity attack that adds API calls to an API call trace used as input to an RNN malware classifier, to bypass a classifier trained on the API call trace of the malware. A GRU substitute model was created and attacked, and the transferability property was used to attack the original classifier. The authors extended their attack to hybrid classifiers combining static and dynamic features, attacking each feature type in turn. The target models were LSTM variants, GRUs, conventional RNNs, bidirectional and deep variants, and non-RNN classifiers (including both feedforward networks, e.g., fully connected DNNs and 1D CNNs, and traditional machine learning classifiers, e.g., SVM, random forest, logistic regression, and gradient boosted decision tree). The authors presented an end-to-end framework that creates a new malware executable without access to the malware source code. The variety of classifiers and the end-to-end framework fits real-life scenarios, but the focus only on strings' static features is limiting.

A subsequent work (Rosenberg et al. [87]) presented two query-efficient gray-box inference integrity attacks against the same classifiers, based on *benign perturbations* generated using a GAN that was trained on benign samples. One of the gray-box attack variants requires the adversary to know which API calls are being monitored, and the other one also requires the confidence score of the target classifier to operate an evolutionary algorithm to optimize the perturbation search and reduce the number of queries used. This attack is generic for every camouflaged malware and does not require a per malware pre-deployment phase to generate the adversarial sequence (either using a GAN, as in Hu and Tan [46], or a substitute model, as in Rosenberg et al. [89]). Moreover, the generation is done at runtime, making it more generic and easier to deploy.

4.2 URL Detection

Webpages are addressed by a URL. A URL begins with the protocol used to access the page. The **fully qualified domain name (FQDN)** identifies the server hosting the webpage. It consists of a **registered domain name (RDN)** and prefix referred to as sub-domains. A phisher has full control of the sub-domains and can set them to any value. The RDN is constrained, since it has to be registered with a domain name registrar. The URL may also have a path and query components that also can be changed by the phisher at will.

Table 3. Summary of Adversarial Learning Approaches in URL Detection

Citation	Year	Target Classifier	Attack Type	Attack's Output	Threat Model	Perturbed Features	Reasonable attacker knowledge?	End-to-end attack?	Effective attack?	Representative dataset?	Representative features?
Bahnsen et al. [14]	2018	LSTM	Inference integrity	URL (end-to-end)	Gray-box	URL characters	+	+	+	+	–
Shirazi et al. [95]	2019	State-of-the-art phishing classifiers	Inference integrity	Feature vector	Gray-box	All features used by the classifiers	+	–	+	+	+
AlEroud and Karabatis [4]	2020	RF, NN, DT, LR, SVM	Inference integrity	URL (end-to-end)	Black-box	URL characters	+	+	+	–	–
Anderson et al. [9]	2016	RF	Inference integrity	URL (end-to-end)	Black-box	URL characters	+	+	+	+	–
Sidi et al. [97]	2019	CNN, LSTM, BLSTM	Inference integrity	URL (end-to-end)	Black-box	URL characters	+	+	+	–	–

Consider this URL example: https://www.amazon.co.uk/ap/signin?encoding=UTF8. We can identify the following components: protocol = https; FQDN = www.amazon.co.uk; RDN = amazon.co.uk; path and query = /ap/signin?encoding=UTF8. A summary of the attacks in the URL detection sub-domain is shown in Table 3.

Since URLs can be quite long, URL shortening services have started to appear. In addition to shortening the URL, these services also obfuscate them.

4.2.1 Phishing URL Detection. Phishing refers to the class of attacks where a victim is lured to a fake webpage masquerading as a target website and is deceived into disclosing personal data or credentials. Phishing URLs seem like legitimate URLs and redirect the users to phishing webpages, which mimic the look and feel of their target websites (e.g., a bank website), in the hopes that the user will enter his/her personal information (e.g., password).

Bahnsen et al. [14] performed an inference integrity attack to evade a character-level LSTM-based phishing URL classifier (Bahnsen et al. [13]) by concatenating the effective URLs from historical attacks (thus, this is a gray-box attack). Then, from this full text, sentences with a fixed length were created. An LSTM model used those sentences as a training set to generate the next character. Once the model generated a full prose text, it was divided by http structure delimiters to produce a list of pseudo URLs. Each pseudo URL was assigned a compromised domain such that the synthetic URLs take the form: http://+compromised_domain+pseudo_URL. Although this attack is indeed effective, the concatenation of benign URLs can be signed, making this attack less evasive for real-life classifiers then the generative attacks (e.g., AlEroud and Karabatis [4]) mentioned in the following.

Shirazi et al. [95] generated adversarial examples using all possible combinations of the values of the features (e.g., website reputation) used by state-of-the-art phishing classifiers, such as Verma and Dyer [111], making this a more realistic attack. However, the attack requires knowledge about the features being used by the classifier, making it a gray-box inference integrity attack. Such knowledge is not always accessible to the attacker, making this attack less feasible in real-life scenarios.

Phishing URLs were generated by a text GAN in Anand et al. [6] and Trevisan and Drago [110] to augment the phishing URL classifier's training set and improve its accuracy. AlEroud and Karabatis [4] used the phishing URLs generated as adversarial examples in an inference integrity attack to bypass the target classifier. It remains unclear whether the attack mentioned in AlEroud and Karabatis [4] is robust enough to bypass GAN-based defenses, such as the defense methods presented in Anand et al. [6] and Trevisan and Drago [110].

4.2.2 Domain Generation AlgorithmURL Detection. **Domain generation algorithms (DGAs)** are commonly used malware tools that generate large numbers of domain names that can be used for difficult-to-track communications with command and control servers operated by the attacker. The large number of varying domain names makes it difficult to block malicious domains using standard techniques such as blacklisting or sinkholing. DGAs are used in a variety of cyber attacks, including ransomware, spam campaigns, theft of personal data, and implementation of distributed denial-of-service attacks. DGAs allow malware to generate any number of domain names daily, based on a seed that is shared by the malware and the threat actor, allowing both to synchronize the generation of domain names.

Sidi et al. [97] used a black-box inference integrity attack, training a substitute model to simulate the DGA classifier on a list of publicly available DGA URLs. Then that attacker iterates over every character in the DGA URL. In each iteration, the results of the feedforward pass of the substitute model are used to compute the loss with regard to the benign class. The attacker performs a single backpropagation step on the loss to acquire the Jacobian-based saliency map, which is a matrix that assigns every feature in the input URL with a gradient value (JSMA; see Appendix B in the supplemental materials). Features (characters) with higher gradient values in the JSMA would have a more significant (salient) effect on the misclassification of the input, and thus each character would be modified in turn, making the substitute model's benign score higher. Finally, URLs that evade detection by the substitute model would also evade detection by the target DGA classifier due to the transferability property (see Appendix B in the supplemental materials). Despite the fact that this attack confirms the feasibility of transferability in the DGA URL detection sub-domain, the use of (only) character-level features does not accurately represent real-life classifiers.

Anderson et al. [9] performed an inference integrity black-box attack that used a GAN to produce domain names that current DGA classifiers would have difficulty identifying. The generator was then used to create synthetic data on which new models were trained. This was done by building a neural language architecture, a method of encoding language in a numerical format, using LSTM layers to act as an autoencoder. The autoencoder is then repurposed such that the encoder (which takes in domain names and outputs an embedding that converts a language into a numerical format) acts as the discriminator, and the decoder (which takes in the embedding and outputs the domain name) acts as the generator. Anderson et al. attacked a random forest classifier trained on features defined in Antonakakis et al. [10], Schiavoni et al. [92], and Yadev et al. [122, 123]. The features of the random forest DGA classifier are unknown to the attacker. They include the length of domain name; entropy of character distribution in domain name; vowel to consonant ratio; Alexa top 1M n-gram frequency distribution co-occurrence count, where n = 3, 4, or 5; n-gram normality score; and meaningful character ratio. The fact that this attack bypasses a classifier that uses many features, as specified eralier, makes it more suitable for real-life scenarios.

4.3 Network Intrusion Detection

A security system commonly used to secure networks is the NIDS, which is a device or software that monitors all traffic passing a strategic point for malicious activities. When such an activity is detected, an alert is generated. Typically an NIDS is deployed at a single point, such as the Internet gateway. A summary of the attacks in the network intrusion detection sub-domain is provided in Table 4.

Clements et al. [27] conducted a white-box inference integrity attack against Kitsune (Mirsky et al. [72]), an ensemble of autoencoders used for online network intrusion detection. Kitsune uses packet statistics which are fed into a feature mapper that divides the features between the autoencoders to ensure fast online training and prediction. The RMSE output of each autoencoder is fed into another autoencoder that provides the final RMSE score used for anomaly detection. This architecture can be executed on small, weak routers.

Clements et al. [27] used four adversarial methods: FGSM, JSMA, C&W, and ENM (see Appendix B in the supplemental materials). The attacker uses the L_P distance on the feature space between the original input and the perturbed input as the distance metric. Minimizing the L_0 norm correlates to altering a small number of extracted features. This method has two main limitations. First, the threat model assumes that the attacker knows the target classifier's features, architecture, and hyperparameters. This makes this attack a white-box attack rather than a black-box attack. This is a less realistic assumption in real-life scenarios. Second, modification is done at the feature level (i.e., modifying only the feature vector) and not at the sample level (i.e., modifying the network stream). This means that there is no guarantee that those perturbations can be performed without harming the malicious functionality of the network stream. The fact that some of the features are statistical makes the switch from vector modification to sample modification even more challenging.

Lin et al. [67] generated adversarial examples using a GAN, called *IDSGAN*, in which the GAN's discriminator obtains the labels from the black-box target classifier. The adversarial examples are evaluated against several target classifiers: SVM, naive Bayes, MLP, logistic regression, decision tree, random forest, and k-nearest neighbors classifiers. This attack assumes knowledge about the target classifier's features, making it a gray-box inference integrity attack. The features include individual TCP connection features (e.g., the protocol type), domain knowledge based features (e.g., a root shell was obtained), and statistical features of the network sessions (like the percentage of connections that have SYN errors within a time window). All features are extracted from the network stream (the NSL-KDD dataset was used; see Table 1 in the supplemental materials). This attack generates a statistical feature vector, but the authors do not explain how to produce a real malicious network stream that has those properties.

Yang et al. [125] trained a DNN model to classify malicious behavior in a network using the same features as Lin et al. [67], achieving performance comparable to state-of-the-art NIDS classifiers, and then showed how to add small perturbations to the original input to lead the model to misclassify malicious network packets as benign while maintaining the maliciousness of these packets. This attack assumes that an adversary without internal information on the DNN model is trying to launch black-box attack. Three different black-box attacks were attempted by the adversary: an attack based on zeroth-order optimization (ZOO; see Appendix B in the supplemental materials), an attack based on a GAN (similar to the one proposed by Lin et al. [67]), and an attack on which a substitute model is trained that is followed by a C&W attack (see Appendix B in the supplemental materials) performed against the substitute model. Applying the generated adversarial example against the target classifier succeeds due to the transferability property (see Appendix B in the supplemental materials). This work has the same limitations as Lin et al. [67]:

Table 4. Summary of Adversarial Learning Approaches in Network Intrusion Detection

Citation	Year	Target Classifier	Attack Type	Attack's Output	Threat Model	Perturbed Features	**Reasonable attacker knowledge?**	**End-to-end attack?**	**Effective attack?**	**Representative dataset?**	**Representative features?**
Clements et al. [27]	2019	Autoencoder ensemble	Inference integrity	Feature vector	White-box	Protocol statistical features	–	–	+	+	+
Lin et al. [67]	2018	SVM, NB, MLP, LR, DT, RF, KNN	Inference integrity	Feature vector	Gray-box	Statistical and protocol header features	+	–	+	–	+
Yang et al. [125]	2018	DNN	Inference integrity	Feature vector	Gray-box	Same as Lin et al.	+	–	+	–	+
Rigaki and Elragal [82]	2017	DT, RF, SVM	Inference integrity	Feature vector	Gray-box	Same as Lin et al.	+	–	+	–	+
Warzynski and Kolaczek [116]	2018	MLP	Inference integrity	Feature vector	White-box	Same as Lin et al.	–	–	+	–	+
Wang [115]	2018	MLP	Inference integrity	Feature vector	White-box	Same as Lin et al.	–	–	+	–	+
Kuppa et al. [62]	2019	DAGMM, AE, AnoGAN, ALAD, DSVDD, OC-SVM, IF	Inference integrity	PCAP file (end-to-end)	Query-efficient gray-box	Similar to Lin et al. but modifies only non-impactful features like send time	+	+	+	+	+
Ibitoye et al. [51]	2019	FC DNN, SNN	Inference integrity	Feature vector	Gray-box	Statistical and protocol header features	+	–	+	+	+
Huang et al. [49]	2019	MLP, CNN, LSTM	Inference integrity, training availability	Feature vector	White-box	Features from SDN messages	–	–	+	+	+

this gray-box inference integrity attack assumes knowledge about the target classifier's features and also generates only the feature vectors and not the samples themselves.

In their gray-box inference integrity attack, Rigaki and Elragal [82] used the same NSL-KDD dataset (see Table 1 in the supplemental materials). Both FGSM and JSMA (see Appendix B in the supplemental materials) attacks were used to generate adversarial examples against an MLP substitute classifier, and the results were evaluated against decision tree, random forest, and linear SVM classifiers. This work has the same limitations as Lin et al. [67]: this attack assumes knowledge about the target classifier's features and also generates only the feature vectors and not the samples themselves.

Warzynski and Kolaczek [116] performed a white-box inference integrity feature vector attack against an MLP classifier trained on the NSL-KDD dataset (see Table 1 in the supplemental materials). They used a white-box FGSM attack (see Appendix B in the supplemental materials). Wang [115] added three more white-box attacks: JSMA, DeepFool, and C&W (see Appendix B in the supplemental materials). The L_p distance and the perturbations are in the feature space in both cases. This attack is not an end-to-end attack, so again it cannot be used to generate malicious network streams that bypass real-life classifiers.

Kuppa et al. [62] proposed a query-efficient gray-box inference integrity attack against deep unsupervised anomaly detectors, which leverages a manifold approximation algorithm for query reduction and generates adversarial examples using spherical local subspaces while limiting the input distortion and KL divergence. Seven state-of-the-art anomaly detectors with different underlying architectures were evaluated: a deep autoencoding Gaussian mixture model, an autoencoder, anoGAN, adversarially learned anomaly detection, deep support vector data description, one-class support vector machines, and isolation forests (see Appendix A in the supplemental materials), which is a more diverse set of classifiers than other attacks mentioned in this section, making this attack more applicable, regardless of the classifier deployed. All classifiers were trained on the CSE-CIC-IDS2018 dataset and features (see Table 1 in the supplemental materials). This dataset is more recent than the NSL-KDD dataset used in much of the research mentioned in this section and better represent today's threats. Unlike other papers discussed in this section, the authors generated a full PCAP file (and not just feature vectors). They also only modified features that could be modified without harming the network stream (e.g., time-based features), so they actually created adversarial samples and not just feature vectors. However, they did not run the modified stream in order to verify that the malicious functionality remains intact.

Ibitoye et al. [51] attacked a fully connected DNN and a self-normalizing neural network classifier (an SNN is a DNN with a SeLU activation layer; Klambauer et al. [55]) trained on the BoT-IoT dataset and features (see Table 1 in the supplemental materials), using FGSM (see Appendix B in the supplemental materials), the basic iteration method, and the PGD at the feature level. They showed that both classifiers were vulnerable, although SNN was more robust to adversarial examples. This attack is unique in terms of the dataset and architectures used and demonstrates the susceptibility of IoT SNN-based classifiers to adversarial attacks.

Huang et al. [49] attacked port scanning detectors in a **software-defined network (SDN)**. The detectors were MLP, CNN, and LSTM classifiers trained on features extracted from Packet-In messages (used by port scanning tools like Nmap in the SDN) and **switch monitoring statistic messages (STATS)**. The white-box inference integrity attacks used were FGSM and JSMA (see Appendix B in the supplemental materials). The JSMA attack was also (successfully) conducted on *regular* traffic packets (JSMA reverse) to create false negatives, creating noise and confusion in the network (a white-box training availability attack). Although this attack requires permissions not available to a regular attacker (knowledge of the classifier's architecture, etc.), it shows the susceptibility of port scanning classifiers to adversarial attacks.

Table 5. Summary of Adversarial Learning Approaches in Spam Filtering

Citation	Year	Target Classifier	Attack Type	Attack's Output	Threat Model	Perturbed Features	Reasonable attacker knowledge?	End-to-end attack?	Effective attack?	Representative dataset?	Representative features?
Sethi and Kantardzic [93]	2018	SVM, kNN, DT, RF	Inference integrity and confidentiality	Feature vector	Gray-box	Email words or same as Lin et al.	+	−	+	+	+
Huang et al. [50], Nelson et al. [74]	2011	Bayesian spam filter	Training availability	Email (end-to-end)	Gray-box	Email words	+	+	+	+	+
Biggio et al. [18]	2014	SVM, LR	Inference integrity	Email (end-to-end)	White-box	Email words	−	+	+	+	+
Brückner et al. [20]	2012	NB, SVM	Inference integrity	Email (end-to-end)	Gray-box	Email words	+	+	+	+	+
Kuleshov et al. [61]	2018	NB, LSTM, 1D CNN	Inference integrity	Email (end-to-end)	Gray-box	Email words	+	+	+	+	+
Lei et al. [64]	2018	LSTM, 1D CNN	Inference integrity	Email (end-to-end)	Gray-box	Email words	+	+	+	+	+
Muñoz González et al. [73]	2017	LR, MLP	Training integrity	Feature vector	Gray-box; *add* and *read* training set access	Email words	−	−	+	+	+

4.4 Spam Filtering

The purpose of a spam filter is to determine whether an incoming message is legitimate (i.e., ham) or unsolicited (i.e., spam). Spam detectors were among the first applications to use machine learning in the cyber security domain and therefore were the first to be attacked. A summary of the attacks in the spam filtering sub-domain is shown in Table 5.

Huang et al. [50] attacked SpamBayes (Robinson [83]), which is a content-based statistical spam filter that classifies email using token counts. SpamBayes computes a spam score for each token in the training corpus based on its occurrence in spam and non-spam emails. The filter computes a message' s overall spam score based on the assumption that the token scores are independent, and then it applies Fisher' s method for combining significance tests to determine whether the email' s tokens are sufficiently indicative of one class or the other. The message score is compared against two thresholds to select the label: spam, ham (i.e., non-spam), or unsure.

Huang et al. [50] designed two types of training availability attacks. The first is an indiscriminate dictionary attack, in which the attacker sends attack messages that contain a very large set of tokens—the attack' s dictionary. After training on these attack messages, the victim' s spam filter will have a higher spam score for every token in the dictionary. As a result, future legitimate email

is more likely to be marked as spam, since it will contain many tokens from that lexicon. The second attack is a targeted attack—the attacker has some knowledge of a specific legitimate email that he/she targets for incorrect filtering. Nelson et al. [74] modeled this knowledge by letting the attacker know a certain fraction of tokens from the target email, which are included in the attack message. Availability attacks like these are quite rare in the adversarial learning landscape and open up interesting attack options, such as adversarial denial-of-service attacks (see Section 6.2).

Biggio et al. [18] evaluated the robustness of linear SVM and logistic regression classifiers to a white-box inference integrity attack where the attacker adds the most impactful good words and found that although both classifiers have the same accuracy for unperturbed samples, the logistic regression classifier outperforms the SVM classifier in robustness to adversarial examples. The white-box approach used in this attack is less feasible in real-life scenarios.

Brückner et al. [20] modeled the interaction between the defender and the attacker in the spam filtering domain as a static game in which both players act simultaneously (i.e., without prior information about their opponent' s move). When the optimization criterion of both players depends not only on their own action but also on their opponent' s move, the concept of a player' s optimal action is no longer well defined, and thus the cost functions of the learner (the defender) and the data generator (the attacker) are not necessarily antagonistic. The authors identified the conditions under which this prediction game has a unique Nash equilibrium and derived algorithms that find the equilibrial prediction model. From this equation, they derived new equations for the Nash logistic regression and Nash SVM using custom loss functions. The authors showed that both the attacker and the defender are better off respectively attacking and using the Nash classifiers. Although this attack took a different (more game theory focused) approach than other more "conventional" attacks, its effectiveness is no different.

Sethi and Kantardzic [93] trained several classifiers (linear SVM, k-nearest neighbors, SVM with RBF kernel, decision tree, and random forest) on several datasets, including SPAMBASE for spam detection and NSL-KDD (see Table 1 in the supplemental materials) for network intrusion detection. They presented a gray-box inference integrity and confidentiality attack and a query-efficient gray-box anchor point inference integrity attack that is effective against all models. The uniqueness of this study lies in its focus on query efficiency (thus making this attack more feasible) and on the confidentiality attack (trying to reverse engineer the attacked model, i.e., a model inversion attack), which are quite rare in the spam filtering domain.

Generalized attack methods, which are effective against several NLP classifiers, are a recent trend. Kuleshov et al. [61] implemented such a generalized black-box inference integrity attack to evade NLP classifiers, including spam filtering, fake news detection, and sentiment analysis. The greedy attack finds semantically similar words (enumerating all possibilities to find words with a minimal distance and score difference from the original input) and replacing them in sentences with a high language model score. Three classifiers were evaded: NB, LSTM, and a word-level 1D CNN.

Lei et al. [64] did the same while using a joint sentence and word paraphrasing technique to maintain the original semantics and syntax of the text. They attacked LSTM and a word-level 1D CNN trained on same datasets used by Kuleshov et al. [61], providing a more effective attack on many datasets, including a spam filtering dataset.

This interesting approach of generalization can be extended in the future by applying other NLP-based attacks in the domain of spam adversarial examples.

4.5 CPSs and Industrial Control Systems

CPSs and **industrial control systems (ICSs)** consist of hardware and software components that control and monitor physical processes, such as critical infrastructure, including the electric power

Table 6. Summary of Adversarial Learning Approaches in Cyber-Physical Systems

Citation	Year	Target Classifier	Attack Type	Attack's Output	Threat Model	Perturbed Features	Reasonable attacker knowledge?	End-to-end attack?	Effective attack?	Representative dataset?	Representative features?
Specht et al. [100]	2018	FC DNN	Inference integrity	Feature vector	White-box	Sensor signals	–	–	+	+	+
Ghafouri et al. [37]	2018	LR, DNN	Inference integrity	Feature vector	Gray-box	Sensor data	+	–	+	+	+
Clark et al. [26]	2018	RL Q-Learning	Inference integrity	Feature vector	White-box	Ultrasonic collision avoidance sensor data	–	–	+	+	+
Feng et al. [33]	2017	LSTM	Inference integrity	Feature vector	Gray-box	Sensor data	+	–	+	+	+
Erba et al. [31]	2019	Autoencoders	Inference integrity/ availability	Feature vector	Gray-box	Sensor data	+	–	+	+	+
Yaghoubi and Fainekos [124]	2019	RNN	Inference integrity	Feature vector	White-box	Continuous sensor data	–	–	+	+	+
Li et al. [65]	2020	FC DNN	Inference integrity	Feature vector	Gray-box	Sensor data	+	–	+	+	+

grid, transportation networks, water supply networks, nuclear plants, and autonomous car driving. A summary of the attacks in the CPS sub-domain is provided in Table 6.

Specht et al. [100] trained a fully connected DNN on the SECOM dataset, recorded from a semiconductor manufacturing process, which consists of 590 attributes collected from sensor signals and variables during manufacturing cycles. Each sensor data entry is labeled as either a normal or anomalous production cycle. They used the FGSM white-box inference integrity feature vector attack to camouflage abnormal/dangerous sensor data so it appears normal. This attack uses a white-box approach, making it less feasible in real-life scenarios.

Ghafouri et al. [37] conducted a gray-box inference integrity attack on a linear regression based anomaly detector, a neural network regression anomaly detector, and an ensemble of both, using the TE-PCS dataset, which contains sensor data describing two simultaneous gas-liquid exothermic reactions for producing two liquid products. There are safety constraints that must not be violated (e.g., safety limits for the pressure and temperature of the reactor), corresponding to the data. For the linear regression based anomaly detector, the problem of finding adversarial examples of sensor data can be solved using a **mixed-integer linear programming (MILP)** problem. To bypass the neural network regression and ensemble, an iterative algorithm is used. It takes small

steps in the direction of increasing objective function. In each iteration, the algorithm linearizes all of the neural networks at their operating points and solves the problem using MILP as before. The mathematical approach taken in this attack (MILP) can be used in the future to attack different models, such as auto-regressive models, with greater efficiency than current approaches.

Clark et al. [26] used a **Jaguar autonomous vehicle (JAV)** to emulate the operation of an autonomous vehicle. The driving and interaction with the JAV environment used the Q-learning RL algorithm. JAV acts as an autonomous delivery vehicle and transports deliveries from an origination point to a destination location. The attacker' s goal is to cause the JAV to deviate from its learned route to an alternate location where it can be ambushed. A white-box inference integrity attack was chosen with the goal of compromising the JAV' s RL policy and causing the deviation. The attack was conducted by inserting an adversarial data feed into the JAV via its ultrasonic collision avoidance sensor. This attack uses a white-box approach, making it less feasible in real-life scenarios. However, it attacks RL models that are frequently used in CPSs.

Feng et al. [33] presented a gray-box inference integrity attack against an LSTM anomaly detector using a GAN (see Appendix A in the supplemental materials) with a substitute model as a discriminator. Two use cases were evaluated: gas pipeline and water treatment plant sensor data. Li et al. [65] presented a gray-box inference integrity attack against the same dataset but used a constraint-based adversarial machine learning to adhere the intrinsic constraints of the physical systems, modeled by mathematical constraints derived from normal sensor data. This attack targets LSTM classifiers, which are commonly used in ICSs, making this an important attack.

Erba et al. [31] demonstrated inference integrity and availability attacks against an autoencoder-based anomaly detection system of water treatment plant sensor data. Access to both the ICS features and benign sensor data (to train a substitute model) is assumed, making this attack a gray-box attack. This attack enumerates all possible operations for every sensor. This attack targets autoencoder classifiers, which are commonly used to detect anomalies, as done in ICSs, making this an important attack.

Yaghoubi and Fainekos [124] presented a gray-box inference integrity attack against a steam condenser with an RNN-based anomaly detector that monitors continuous (e.g., signal) data. The attack uses gradient-based local search with either uniform random sampling or simulated annealing optimization to find the data to modify.

4.6 Biometric Systems

In this section, we focus on attacks that target the most commonly used biometric systems that leverage machine learning: face, speech, and iris recognition systems. Although these attacks can be viewed as computer vision (or audio) attacks, we focus only on such attacks that affect cyber security authentication thus allowing unauthorized users access to the protected systems. Many studies have focused on adversarial attacks against other types of biometric systems, such as handwritten signature verification (Hafemann et al. [43]), EEG biometrics (Özdenizci et al. [75]), and gait biometric recognition (Prabhu and Dewayne [78]). However, as mentioned previously, these are not discussed here due to space limitations. A summary of the attacks against biometric systems is presented in Table 7.

4.6.1 Face Recognition. Sharif et al. [94] presented an inference integrity attack against face recognition systems. The target classifier for the white-box attack was VGG-Face, a 39-layer CNN (Parkhi et al. [76]). The target classifier for the black-box attack was the Face++ cloud service. Instead of perturbing arbitrary pixels in the facial image, this attack only perturbed the pixels of eyeglasses that were worn by the attacker so the attacker would either be classified as another person (label-target attack) or not be classified as himself (indiscriminate attack). Both attacks

Table 7. Summary of Adversarial Learning Approaches in Biometric Systems

Citation	Year	Target Classifier	Biometric Application	Attack Type	Attack's Output	Threat Model	Perturbed Features	Reasonable attacker knowledge?	End-to-end attack?	Effective attack?	Representative dataset?	Representative features?
Sharif et al. [94]	2016	CNN	Face recognition	Inference integrity	Physical eyeglasses (end-to-end)	Feature-targeted white-box and black-box	Image's pixels	+	+	+	+	+
Liu et al. [69]	2018	CNN	Face recognition	Training integrity	Image (non-physical end-to-end)	Feature-targeted white-box; *add* training set access	Image's pixels	–	+	+	+	+
Chen et al. [24]	2017	CNN	Face recognition	Training integrity	Physical accessory (end-to-end)	Feature-targeted white-box; *add* training set access	Image's pixels	–	+	+	+	+
Kreuk et al. [59]	2018	LSTM/ GRU	Speaker recognition	Inference integrity	Feature vector	White-box or black-box	Mel-spectrum features and MFCCs	+	–	+	+	–
Gong and Poellabauer [38]	2017	Mixed CNN-LSTM	Speaker recognition	Inference integrity	Raw waveform (end-to-end)	Black-box	Raw waveforms	+	+	+	+	+
Du et al. [29]	2019	CNN	Speaker recognition	Inference integrity	Raw waveform (end-to-end)	Black-box	Raw waveforms	+	+	+	+	+
Cai et al. [21]	2018	CNN	Speaker recognition	Inference integrity	Feature vector	Gray-box	Mel-spectrum features	+	–	+	+	–
Wang et al. [114]	2018	CNN	Face recognition, Iris recognition	Inference integrity	Image (non-physical end-to-end)	Gray-box	Image's pixels	+	+	+	+	+
Taheri et al. [107]	2019	CNN	Fingerprint recognition, iris recognition	Inference integrity	Image (non-physical end-to-end)	White-box	Image's pixels	-	+	+	+	+
Soleymani et al. [98, 99]	2019	CNN	Iris recognition	Inference integrity	Feature vector	Gray-box	Iris codes	+	-	+	+	-

are feature targeted. Sharif et al. [94] used a commodity inkjet printer to print the front plane of the eyeglass frames on glossy paper, which they then affixed to the actual eyeglass frames when physically performing attacks. This makes this end-to-end attack interesting, as it can be used (in its black-box variant) in real-life scenarios.

Both Chen et al. [24] and Liu et al. [69] used a black-box training integrity poisoning attack against face recognition systems. The CNN target classifiers were VGG-Face (Parkhi et al. [76]) and DeepID (Sun et al. [106]), respectively. Liu et al. [69] used a non-physical image, such as a

square appearing in the picture, as the Trojan trigger in the picture to be labeled as a different person. In Chen et al. [24], the poisoned facial images contained a physical accessory as the key; a photo of a person taken directly from the camera can then become a backdoor when the physical accessory is worn; thus, this is both a feature-targeted and a label-targeted attack. Both attacks require training set *add* access. Despite the fact that these are white-box attacks, the usage of feature-targeting fits nicely with some forms of real-life attacks, such as supply-chain attacks, where a powerful adversary wants to exploit a very specific scenario (e.g., evade detection of a person only when he/she is holding a key).

4.6.2 Speaker Verification/Recognition. Note that in this section, we only discuss speaker recognition and verification system adversarial attacks.

Kreuk et al. [59] presented white-box inference integrity attacks on an LSTM/GRU classifier that was either trained on the YOHO or NTIMIT datasets using two types of features: Mel-spectrum features and MFCCs. They also presented two black-box inference integrity attacks, using the transferability property. In the first one, they generated adversarial examples with a system trained on the NTIMIT dataset and performed the attack on a system that was trained on YOHO. In the second attack, they generated the adversarial examples with a system trained using Mel-spectrum features and performed the attack on a system trained using MFCCs. All of the attacks used the FGSM attack, and the attack output was a feature vector and not a complete audio sample. This, and the fact that this attack uses a white-box approach, makes it less feasible in real-life scenarios.

Gong and Poellabauer [38] trained a WaveRNN model (a mixed CNN-LSTM model) on raw waveforms (the IEMOCAP dataset's utterances; see Table 1 in the supplemental materials) for speaker recognition, as well as emotion and gender recognition. They used a substitute WaveCNN model and performed a black-box inference integrity attack using FGSM on the raw waveforms rather than on the acoustic features, making this an end-to-end attack that does not require an audio reconstruction step. The use of a gray-box approach and raw waveform features makes this attack feasible in real-life scenarios.

Du et al. [29] used six state-of-the-art speech command recognition CNN models: VGG19, DenseNet, ResNet18, ResNeXt, WideResNet18, and DPN-92, all adapted to the raw waveform input. The models were trained for speaker recognition on the IEMOCAP dataset (see Table 1 in the supplemental materials) and for speech recognition, sound event classification, and music genre classification using different datasets. The black-box inference integrity attack used FGSM or **particle swarm optimization (PSO)** on the raw waveforms. The use of a gray-box approach and raw waveform features and the evaluation against many classifiers make this attack feasible in real-life scenarios.

Cai et al. [21] trained a CNN classifier that performs multi-speaker classification using Mel-spectrograms as input. They used a **Wasserstein GAN with gradient penalty (WGAN-GP)** to generate adversarial examples for an indiscriminate gray-box inference integrity attack and also used a WGAN-GP with a modified objective function for a specific speaker for a targeted attack. The attack output is a feature vector of Mel-spectrograms and not an audio sample. This attack uses a white-box approach, making it less feasible in real-life scenarios.

4.6.3 Iris and Fingerprint Systems. Wang et al. [114] performed an indiscriminate black-box inference integrity attack, leveraging the fact that many image-based models, including face recognition and iris recognition models, use transfer learning—that is, they add new layers on top of pre-trained layers that are trained on a different model (a teacher model with a known architecture) and are used to extract high-level feature abstractions from the raw pixels. For instance, the face recognition model's teacher model can be VGG-Face (Parkhi et al. [76]), whereas an iris model's teacher model can be VGG16. By attacking the teacher model using white-box attacks, such as

C&W, the target model (student model), for which the architecture is unknown, is also affected. This approach can be used to attack many classifiers, especially those in domains where transfer learning using a known model is common, like the computer vision domain.

Taheri et al. [107] trained a CNN classifier on the CASIA dataset of images of iris and fingerprint data. They implemented a white-box inference integrity attack using the FGSM, JSMA, DeepFool, C&W, and PGD methods (see Appendix B in the supplemental materials) to generate the perturbed images. This attack uses a white-box approach, making it less feasible in real-life scenarios.

Soleymani et al. [98], ,99] generated adversarial examples for code-based iris recognition systems, using a gray-box inference integrity attack. However, conventional iris code generation algorithms are not differentiable with respect to the input image. Generating adversarial examples requires backpropagation of the adversarial loss. Therefore, they used a deep autoencoder substitute model to generate iris codes that were similar to iris codes generated by a conventional algorithm (OSIRIS). This substitute model was used to generate the adversarial examples using FGSM, iGSM, and DeepFool attacks. The attack of iris codes can serve as the initial step toward end-to-end adversarial attacks against biometric systems.

5 ADVERSARIAL DEFENSE METHODS IN THE CYBER SECURITY DOMAIN

Our taxonomy focuses on the attack side, but every attack is accompanied by a corresponding defense method.

If adversarial attacks are equivalent to malware attacking a computer (a machine learning model), then defense methods can be viewed as anti-malware products. However, most defense methods have been evaluated in the image recognition domain for CNNs and in the NLP domain for RNNs. Due to space limitations, we cannot provide a complete list of the state-of-the-art prior work in those domains. A more comprehensive list can be found, for example, in Qiu et al. [79].

Several papers presenting attacks in the cyber security domain (e.g., Grosse et al. [41], Sidi et al. [97]) discuss the fact that the attack is effective even in the presence of well-known defense methods that were evaluated and found effective in the computer vision domain (e.g., distillation and adversarial retraining). However, only a few defense methods were developed specifically for the cyber security domain and its unique challenges, like those described in Section 2. Furthermore, cyber security classifiers usually have a different architecture than computer vision classifiers, against which most of the methods presented in prior research were evaluated.

Adversarial defense methods can be divided into two subgroups: (1) *detection methods,* methods used to detect adversarial examples, and (2) *robustness methods*, methods used to make a classifier more robust to adversarial examples, without explicitly trying to detect them.

Each defense method is either *attack specific* (i.e., it requires adversarial examples generated by the attack to mitigate the attack) or *attack agnostic* (i.e., it works against all types of attack methods, without the need to have a dataset of adversarial examples generated by those attacks). Attack-agnostic defense methods are more generic and are therefore preferable.

In the malware detection sub-domain, Chen et al. [23] suggested an attack-agnostic method to make an Android malware classifier robust to poisoning attacks. Their method includes two phases: an offline training phase that selects and extracts features from the training set, and an online detection phase that utilizes the classifier trained by the first phase. These two phases are intertwined through a self-adaptive learning scheme, in which an automated camouflage detector is introduced to filter the suspicious false negatives and feed them back into the training phase. Stokes et al. [103] evaluated three attack-agnostic robustness defense methods: weight decay, an ensemble of classifiers, and distillation (neural network compression, resulting in gradient masking) for a dynamic analysis malware classifier based on a non-sequence-based DNN. Rosenberg et al. [88] tried to defend an API call-based RNN classifier and compared their own RNN

attack-agnostic robustness defense method, termed *sequence squeezing*, to four robustness defense methods inspired by existing CNN-based defense methods: adversarial retraining (retraining the classifier after adding adversarial examples), defense GAN (generating several samples using a GAN and using the closest one to the original input as the classifier's input), nearest neighbor classification, and RNN ensembles. They showed that sequence squeezing provides the best trade-off between training and inference overhead (which is less critical in the computer vision domain) and adversarial robustness.

For DGA detection, Sidi et al. [97] evaluated the robustness defense methods of adversarial retraining (attack specific) and distillation (attack agnostic), showing that they improve the robustness against adversarial attacks.

For CPS defense, Kravchik and Shabtai [58] suggested detecting adversarial attacks in ICS data using 1D CNNs and undercomplete autoencoders, an attack-agnostic method. The authors demonstrate that attackers must sacrifice much of their attack's potential damage to remain hidden from the detectors. Ghafouri et al. [37] presented robust linear regression and neural network regression based anomaly detectors for CPS anomalous data detection by modeling a game between the defender and attacker as a Stackelberg game in which the defender first commits to a collection of thresholds for the anomaly detectors, and the attacker then computes an optimal attack. The defender aims to minimize the impact of attacks, subject to a constraint typically set to achieve a target number of false alarms without consideration of attacks.

For spam detection, Alzantot et al. [5] evaluated the attack-specific robustness defense methods of adversarial retraining against RNN classifiers, showing that such methods improve robustness against adversarial attacks.

For biometric systems defense methods, Taheri et al. [107] presented an attack-specific detection architecture that includes shallow networks and DNNs to defend against biometric adversarial examples. The shallow neural network is responsible for data preprocessing and generating adversarial samples. The DNN is responsible for understanding data and information, as well as for detecting adversarial samples. The DNN gets its weights from transfer learning, adversarial training, and noise training. Specht et al. [100] suggested an attack-specific robustness defense method using an iterative adversarial retraining process to mitigate adversarial examples for semi-conductor anomaly detection system of sensor data. Soleymani et al. [98] used an attack-agnostic robustness defense method involving wavelet domain denoising of the iris samples, by investigating each wavelet sub-band and removing the sub-bands that are most affected by the adversary.

In our opinion, other defense methods proposed for the computer vision domain could inspire similar defense methods in the cyber domain. However, their effectiveness in the cyber security domain would need to be evaluated, since as discussed earlier, the cyber security domain is different and has unique challenges. Furthermore, in the cyber security domain, further emphasis should be put on the defense method overhead (as done, e.g., in Rosenberg et al. [88]), since cyber security classifiers usually perform their classification in real time, meaning that low overhead is critical, unlike in the computer vision domain. Finally, we believe that the research attention should be given to attack-agnostic defense methods, as attack-specific defense methods (e.g., adversarial retraining) provide a very narrow defense against the growing variety of adversarial attacks presented in this study.

6 CURRENT GAPS AND FUTURE RESEARCH DIRECTIONS FOR ADVERSARIAL LEARNING IN THE CYBER SECURITY DOMAIN

In this section, we highlight gaps in our taxonomy, presented in Section 3, which are not covered by the applications presented in Sections 4 and 6.6, despite having a required functionality. Each such gap is presented in a separate section in the following. For each gap, we summarize the progress

made on this topic in other domains of adversarial learning, such as the computer vision domain, and extrapolate future trends in the cyber security domain from it.

6.1 Attack's Targeting Gap: Feature-Targeted Attacks and Defenses

Poisoning integrity attacks place mislabeled training points in a region of the feature space far from the rest of the training data. The learning algorithm labels such a region as desired, allowing for subsequent misclassifications at test time. However, adding samples to the training set might cause misclassification of many samples and thus would raise suspicion of an attack, whereas the adversary might want to evade just a specific sample (e.g., a dedicated APT).

In non-cyber security domains, a feature-targeted attack, also known as a Trojan neural network attack (Liu et al. [69]) or backdoor attack (Gu et al. [42], Chen et al. [24]), is a special form of poisoning attack, which aims to resolve this problem. A model poisoned by a backdoor attack should still perform well on most inputs (including inputs that the end user may hold out as a validation set) but cause targeted misclassifications or degrade the accuracy of the model for inputs that satisfy some secret, attacker-chosen property, which is referred to as the backdoor trigger. For instance, adding a small rectangle to a picture would cause it to be classified with a specific target label (Liu et al. [69]). Such attacks were performed on face recognition (Chen et al. [24], Liu et al. [69]), traffic sign detection (Gu et al. [42]), sentiment analysis, speech recognition, and autonomous driving (Liu et al. [69]) datasets.

However, such attacks have not been applied yet in the cyber security domain, despite the fact that such attacks have interesting use cases. For instance, such an attack might allow only a specific nation-state APT to bypass the malware classifier, while still detecting other malware, leaving the defender unaware of this feature-targeted attack.

Defenses against such attacks are also required. In the image recognition domain, Wang et al. [113] generates a robust classifier by pruning backdoor-related neurons from the original DNN. Gao et al. [35] detects a Trojan attack during runtime by perturbing the input and observing the randomness of predicted classes for perturbed inputs on a given deployed model. A low entropy in predicted classes implies the presence of a Trojaned input. Once feature-targeted attacks are published in the cyber security domain, defense methods to mitigate them will follow.

6.2 Attacker's Goals Gap: Resource Exhaustion-Based Availability Attacks

There are two availability attack variants. One is the subversion attack, in which the system is fooled so it does not serve legitimate users. Such adversarial attacks have been implemented in the cyber security domain by fooling the classifier into considering legitimate traffic as malicious, thus blocking it (e.g., Huang et al. [50], Nelson et al. [74]). The other variant is the resource exhaustion attack, in which all of a system' s available resources are used to prevent legitimate users from using them. The second variant is very common in the cyber security domain (e.g., distributed denial-of-service attacks on websites, zip bombs) but not in adversarial attacks in this domain.

Given recent advancements in the computer vision domain, such as sponge examples (Shumailov et al. [96]), which are adversarial examples whose classification requires 10 to 200 times the resources of classifying a regular sample, we believe that it is only a matter of time before such examples will reach the cyber security domain (e.g., PE files that take a very long to classify).

6.3 Attacker's Goals and Knowledge Gap: Confidentiality Attacks via Model Queries and Side-Channels

Reverse engineering (reversing) of traditional (non-machine learning based) anti-malware is a fundamental part of a malware developer' s modus operandi. So far, confidentiality attacks have only been conducted against image recognition models and not against cyber security related models

(e.g., malware classifiers). However, performing such attacks in the cyber security domain might provide the attacker with enough data to perform more effective white-box attacks instead of black-box ones.

In the image recognition domain, confidentiality attacks have been conducted by querying the target model. Tramèr et al. [109] formed a query-efficient gray-box (the classifier type should be known) score-based attack. The attack used equation solving to recover the model's weights from sets of observed sample-confidence score pairs $(x, h(x))$ retrieved by querying the target model. For instance, a set of n such pairs is enough to retrieve the n weights of a logistic regression classifier using n-dimensional input vectors. Wang and Gong [112] used a similar approach to retrieve the model's hyperparameters (e.g., the factor of the regularization term in the loss function equation).

In non-cyber domains, confidentiality attacks have also been conducted via side-channel information. Duddu et al. [30] used timing attack side-channels to obtain neural network architecture information. Batina et al. [16] used electromagnetic radiation to get neural network architecture information from embedded devices. Hua et al. [48] used both a timing side-channel and off-chip memory access attempts during inference to discover on-chip CNN model architecture.

6.4 Perturbed Features Gap: Exploiting Vulnerabilities in Machine and Deep Learning Frameworks

In non-machine learning based cyber security solutions, vulnerabilities are commonly used to help attackers reach their goals (e.g., use buffer overflows to run adversary-crafted code in a vulnerable process). A vulnerability in the underlying application or operating system is a common attack vector for disabling or bypassing a security product. This trend is starting to be seen in the adversarial learning domain against image classifiers; however, such attacks have not been demonstrated against malware classifiers. Such vulnerabilities are specialized, and should be researched explicitly for the proper cyber use cases to be effective.

In the image recognition domain, Xiao et al. [118] discovered security vulnerabilities in popular deep learning frameworks (Caffe, TensorFlow, and Torch). Stevens et al. [102] did the same for OpenCV (a computer vision framework), scikit-learn (a machine learning framework), and Malheur (a dynamic analysis framework used to classify unknown malware using a clustering algorithm (Rieck et al. [81])). By exploiting these frameworks' implementations, attackers can launch denial-of-service attacks that crash or hang a deep learning application, or control-flow hijacking attacks that lead to either system compromise or recognition evasions.

We believe that future adversarial attacks can view the deep learning framework used to train and run the model as a new part of the attack surface, leveraging vulnerabilities in the framework, which can even be detected automatically by a machine learning model (as reviewed in Ghaffarian and Shahriari [36]). Some of these vulnerabilities can be used to add data to input in a way that would cause the input to be misclassified, just as adversarial perturbation would, but by subverting the framework instead of subverting the algorithm. This can be viewed as an extension of the perturbed features attack characteristics in our taxonomy.

6.5 Attack's Output Gap: End-to-End Attacks in Complex Format Sub-Domains

As discussed in Section 3.7, only end-to-end attacks can be used to attack systems in the cyber security domain. Some sub-domains, such as email, have a simple format, and therefore it is easier to map from features (words) back to a sample (email) and create an end-to-end attack. However, in more complex sub-domains, such as NIDS and CPSs, the mapping from features to a full sample (e.g., a network stream) is complex. As can be seen in Sections 4.3 and 4.5, only a small number of attacks in the NIDS and CPS sub-domains (less than 10%) are end-to-end attacks.

We predict that the trend seen in the computer vision domain, where after several years of feature vector attacks (e.g., Goodfellow et al. [39]), end-to-end attacks followed Eykholt et al. [32], will also be seen in the cyber security domain. There will likely be three directions for such end-to-end attacks: (1) adding new features to an existing sample (e.g., Li et al. [66], Rosenberg et al. [89], Srndic and Laskov [101]), (2) modifying only a subset of features that can be modified without harming the functionality of an existing sample (e.g., Kuppa et al. [62], Rosenberg et al. [86]), and (3) using cross-sample transformations (e.g., packing) that would change many features simultaneously (Anderson et al. [7], Rosenberg et al. [86]).

6.6 Adversarial Defense Method Gaps

The gaps in the domain of defense methods against adversarial attacks in the cyber security domain is acute because this domain involves actual adversaries: malware developers who want to evade next generation machine and deep learning based classifiers. Such attacks have already been executed in the wild against static analysis DNN Cyl [1]. We have mapped two gaps specific to the cyber security domain, which are shown in the following.

6.6.1 Metrics to Measure the Robustness of Classifiers to Adversarial Examples. Several papers (Katz et al. [53], Peck et al. [77], Weng et al. [117]) suggested measuring the robustness of machine learning systems to adversarial attacks by approximating the lower bound on the perturbation needed for any adversarial attack to succeed; the larger the perturbation, the more robust the classifier. However, these papers assume that the robustness to adversarial attacks can be evaluated by the minimal perturbation required to modify the classifier's decision. This raises the question of whether this metric is valid in the cyber security domain.

Section 2.2 leads us to the conclusion that minimal perturbation is not necessarily the right approach for adversarial learning in the cyber security domain. As already mentioned in Biggio and Roli [19], maximum confidence attacks, such as the C&W attack (Appendix B in the supplemental materials), are more effective. However, this is not the complete picture.

As mentioned in Section 2.3, in the cyber security domain, classifiers usually use more than a single feature type as input (e.g., both PE header metadata and byte entropy in Saxe and Berlin [91]). Certain feature types are easier to modify without harming the executable functionality than others. However, an attacker can add as many strings as needed; in contrast to images, adding more strings (i.e., a larger perturbation) is not more visible to the user than adding less strings, since the executable file is still a binary file.

This means that we should not only take into account the impact of a feature on the prediction but also the difficulty of modifying this feature type. Unfortunately, there is currently no numeric metric to assess the difficulty of modifying features. Currently, we must rely on the subjective opinion of experts who assess the difficulty of modifying each feature type, as shown in Katzir and Elovici [54]. When such a metric becomes available, combining it with the maximum impact metric would be a better optimization constraint than minimal perturbation.

In conclusion, both from an adversary's perspective (when trying to decide which attack to use) and from the defender's perspective (when trying to decide which classifier would be the most robust to adversarial attack), the metric of evaluation currently remains an open question in the cyber security domain.

6.6.2 Defense Methods Combining Domain-Specific and Domain-Agnostic Techniques. Domain-specific constraints and properties have been leveraged in the computer vision domain. For instance, the *feature squeezing* method mentioned in Xu et al. [120] applied image-specific dimensionality reduction transformations to the input features, such as changing the image color depth (e.g., from 24 bit to 8 bit), to detect adversarial examples in images.

We argue that the same approach can be applied in the cyber security domain as well. In the cyber security domain, there are very complex constraints on the raw input. For instance, a PE file has a very strict format (to be loadable by the Windows OS), with severe limitations on the PE header metadata values, and so forth. Such constraints can be leveraged to detect "domain-specific anomalies" more accurately. For example, the attacks used to evade malware classifiers in Rosenberg et al. [185, 96] involved concatenation of strings from the IAT to the EOF/PE overlay; although such concatenation generates a valid PE, one might ask whether it makes sense for such strings to be appended to the EOF instead of, for example, being in the data section or a resource section.

6.6.3 Defense Methods Robust to Unknown and Transparent-Box Adversarial Attacks. There are two main challenges when developing a defense method.

The first challenge is creating a defense method which is also robust against transparent-box attacks, i.e., attackers who know what defense methods are being used and select their attack methods accordingly.

In the computer vision domain, Carlini and Wagner [22] and Tramer et al. [108] showed that many different types of commonly used defense methods (e.g., detection of adversarial examples using statistical irregularities) are rendered useless by a specific type of adversarial attack. He et al. [45] showed the same for feature squeezing, and Athalye et al. [12], Hashemi et al. [44], and Ilyas et al. [52] presented similar results against Defense-GAN.

Similar research should be conducted in the cyber security domain. For instance, attackers can make their attack more robust against RNN subsequence model ensembles presented in Rosenberg et al. [88] by adding perturbations across the entire API call sequence and not just until the classification changes.

The second challenge is creating defense methods that are effective against all attacks and not just specific ones, termed *attack-agnostic* defense methods in Rosenberg et al. [88]. However, the challenge of finding a metric to evaluate the robustness of classifiers to adversarial attacks in the cyber security domain, already discussed in Section 6.6.1, makes the implementation of attack-agnostic defense methods in the cyber security domain more challenging than in other domains.

7 CONCLUSION

In this article, we reviewed the latest research on a wide range of adversarial learning attacks and defenses in the cyber security domain (malware detection, network intrusion detection, etc.).

One conclusion based on our review is that although feature vector adversarial attacks in the cyber security domain are possible, real-life attacks (e.g., against NGAV software) are challenging. This is due to the unique challenges that attackers and defenders are faced with in the cyber security domain; these challenges include the difficulty of modifying samples end-to-end without damaging the malicious business logic and the need to modify many feature types with various levels of modification difficulty.

From the gaps in our taxonomy discussed earlier and from the recent advancements in other domains of adversarial learning, we identified some directions for future research in adversarial learning in the cyber security domain. One direction focuses on the implementation of feature-triggered attacks that would work only if a certain trigger exists, leaving the system's integrity unharmed in other cases, thus making it harder to detect the attack. Another possible direction is performing confidentiality attacks involving model reversing via queries or side-channels. A third research direction aims at expanding the attack surface of adversarial attacks to include the vulnerabilities in the relevant machine learning framework and designing machine learning models to detect and leverage them. From the defender's point of view, more robust defense methods against adversarial attacks in the cyber security domain should be the focus of future research.

A final conclusion derived from our review is that adversarial learning in the cyber security domain is becoming more and more similar to the cat and mouse game played in the traditional cyber security domain, in which the attackers implement more sophisticated attacks to evade the defenders and vice versa. A key takeaway is that defenders should become more proactive in assessing their system's robustness to adversarial attacks, the same way penetration testing is proactively used in the traditional cyber security domain.

REFERENCES

[1] Skylight. 2019. Cylance, I Kill You! Retrieved August 24, 2019 from https://skylightcyber.com/2019/07/18/cylance-i-kill-you.

[2] Ahmed Abusnaina, Aminollah Khormali, Hisham Alasmary, Jeman Park, Afsah Anwar, and Aziz Mohaisen. 2019. Adversarial learning attacks on graph-based IoT malware detection systems. In *Proc. of CDCS*, Vol. 10.

[3] Naveed Akhtar and Ajmal S. Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* 6 (2018), 14410–14430.

[4] Ahmed AlEroud and George Karabatis. 2020. Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks. In *Proc. of IWSPA*.

[5] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proc. of EMNLP*. 2890–2896.

[6] A. Anand, K. Gorde, J. R. Antony Moniz, N. Park, T. Chakraborty, and B. Chu. 2018. Phishing URL detection with oversampling based on text generative adversarial networks. In *Proc. of Big Data*. 1168–1177.

[7] Hyrum S. Anderson, Anant Kharkar, Bobby Filar, David Evans, and Phil Roth. 2018. Learning to evade static PE machine learning malware models via reinforcement learning. arXiv:1801.08917

[8] Hyrum S. Anderson, Anant Kharkar, Bobby Filar, and Phil Roth. 2017. Evading machine learning malware detection. In *Proc. of Black Hat USA*.

[9] Hyrum S. Anderson, Jonathan Woodbridge, and Bobby Filar. 2016. DeepDGA: Adversarially-tuned domain generation and detection. In *Proc. of AISec*. 13–21.

[10] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. 2012. From throw-away traffic to bots: Detecting the rise of DGA-based malware. In *Proc. of USENIX Security*. 491–506.

[11] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, and Konrad Rieck. 2014. DREBIN: Effective and explainable detection of Android malware in your pocket. In *Proc. of NDSS*.

[12] Anish Athalye, Nicholas Carlini, and David A. Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proc. of ICML*. 274–283.

[13] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. González. 2017. Classifying phishing URLs using recurrent neural networks. In *Proc. of eCrime*. 1–8.

[14] Alejandro Correa Bahnsen, Ivan Torroledo, Luis David Camacho, and Sergio Villegas. 2018. DeepPhish : Simulating Malicious AI. Retrieved March 29, 2021 from https://albahnsen.com/2018/06/03/deepphish-simulating-malicious-ai/.

[15] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. 2010. The security of machine learning. *Machine Learning* 81, 2 (2010), 121–148.

[16] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2019. CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel. In *Proc. of USENIX Security*. 515–532.

[17] Daniel Berman, Anna Buczak, Jeffrey Chavis, and Cherita Corbett. 2019. A survey of deep learning methods for cyber security. *Information* 10 (April 2019), 122.

[18] B. Biggio, G. Fumera, and F. Roli. 2014. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering* 26, 4 (April 2014), 984–996.

[19] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proc. of CS*. 2154–2156.

[20] Michael Brückner, Christian Kanzow, and Tobias Scheffer. 2012. Static prediction games for adversarial learning problems. *Journal of Machine Learning Research* 13, 1 (Sept. 2012), 2617–2654.

[21] Wilson Cai, Anish Doshi, and Rafael Valle. 2018. Attacking speaker recognition with deep generative models. arXiv:1801.02384

[22] Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected. In *Proc. of AISec*.

[23] Sen Chen, Minhui Xue, Lingling Fan, Shuang Hao, Lihua Xu, Haojin Zhu, and Bo Li. 2018. Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. *Computers & Security* 73 (2018), 326–344.

[24] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. arXiv:1712.05526
[25] Simon P. Chung and Aloysius K. Mok. 2006. Allergy attack against automatic signature generation. In *Recent Advances in Intrusion Detection*. Lecture Notes in Computer Science, Vol. 4219. Springer. 61–80.
[26] George W. Clark, Michael V. Doran, and William Glisson. 2018. A malicious attack on the machine learning policy of a robotic system. In Proc. of *TrustCom/BidDataSE*. 516–521.
[27] Joseph Clements, Yuzhe Yang, Ankur A. Sharma, Hongxin Hu, and Yingjie Lao. 2019. Rallying adversarial techniques against deep learning for network security. arXiv:1903.11688
[28] Hung Dang, Yue Huang, and Ee-Chien Chang. 2017. Evading classifiers by morphing in the dark. In *Proc. of CCS*. 119–133.
[29] Tianyu Du, Shouling Ji, Jinfeng Li, Qinchen Gu, Ting Wang, and Raheem A. Beyah. 2019. SirenAttack: Generating adversarial audio for end-to-end acoustic systems. arXiv:1901.07846
[30] Vasisht Duddu, Debasis Samanta, D. Vijay Rao, and Valentina E. Balas. 2018. Stealing neural networks via timing side channels. arXiv:1812.11720
[31] Alessandro Erba, Riccardo Taormina, Stefano Galelli, Marcello Pogliani, Michele Carminati, Stefano Zanero, and Nils Ole Tippenhauer. 2019. Real-time evasion attacks with physical constraints on deep learning-based anomaly detectors in industrial control systems. arXiv:1907.07487
[32] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *Proc. of CVPR*.
[33] Cheng Feng, Tingting Li, Zhanxing Zhu, and Deeph Chana. 2017. A deep learning-based framework for conducting stealthy attacks in industrial control systems. arXiv:1709.06397
[34] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *Proc. of SPW*.
[35] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. 2019. STRIP: A defence against trojan attacks on deep neural networks. In *Proc. ofACSAC*.
[36] Seyed Mohammad Ghaffarian and Hamid Reza Shahriari. 2017. Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey. *ACM Computing Surveys* 50, 4 (2017), Article 56, 36 pages.
[37] Amin Ghafouri, Yevgeniy Vorobeychik, and Xenofon D. Koutsoukos. 2018. Adversarial regression for detecting attacks in cyber-physical systems. In *Proc. of IJCAI*. 3769–3775.
[38] Yuan Gong and Christian Poellabauer. 2017. Crafting adversarial examples for speech paralinguistics applications. arXiv:1711.03280
[39] I. J. Goodfellow, J. Shlens, and C. Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proc. of ICLR*.
[40] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel. 2016. Adversarial perturbations against deep neural networks for malware classification. arXiv:1606.04435
[41] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2017. Adversarial examples for malware detection. In *Proc. of ESORICS*. 62–79.
[42] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* 7 (2019), 47230–47244.
[43] Luiz G. Hafemann, Robert Sabourin, and Luiz Eduardo Soares de Oliveira. 2019. Characterizing and evaluating adversarial examples for offline handwritten signature verification. *IEEE Transactions on Information Forensics and Security* 14 (2019), 2153–2166.
[44] Mohammad Hashemi, Greg Cusack, and Eric Keller. 2018. Stochastic substitute training: A gray-box approach to craft adversarial examples against gradient obfuscation defenses. In *Proc. of CCS*. 25–36.
[45] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. 2017. Adversarial example defense: Ensembles of weak defenses are not strong. In *Proc. of WOOT*.
[46] Weiwei Hu and Ying Tan. 2017. Black-box attacks against RNN based malware detection algorithms. arXiv:1705.08131
[47] Weiwei Hu and Ying Tan. 2017. Generating adversarial malware examples for black-box attacks based on GAN. arXiv:1702.05983
[48] Weizhe Hua, Zhiru Zhang, and G. Edward Suh. 2018. Reverse engineering convolutional neural networks through side-channel information leaks. In *Proc. ofDAC*. Article 4, 6 pages.
[49] Chi-Hsuan Huang, Tsung-Han Lee, Lin-Huang Chang, Jhih-Ren Lin, and Gwoboa Horng. 2019. Adversarial attacks on SDN-based deep learning IDS system. In *Proc. of ICMWT*. 181–191.
[50] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I. P. Rubinstein, and J. D. Tygar. 2011. Adversarial machine learning. In *Proc. of AISec*. 43–58.
[51] Olakunle Ibitoye, M. Omair Shafiq, and Ashraf Matrawy. 2019. Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks. arXiv:1905.05137

[52] Andrew Ilyas, Ajil Jalal, Eirini Asteri, Constantinos Daskalakis, and Alexandros G. Dimakis. 2017. The robust manifold defense: Adversarial training using generative models. arXiv:1712.09196
[53] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proc. of CAV*. 97–117.
[54] Ziv Katzir and Yuval Elovici. 2018. Quantifying the resilience of machine learning classifiers used for cyber security. *Expert Systems with Applications* 92 (2018), 419–429.
[55] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Proc. of NIPS*. 971–980.
[56] Clemens Kolbitsch, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiaoyong Zhou, and XiaoFeng Wang. 2009. Effective and efficient malware detection at the end host. In *Proc. of USENIX Security*. 351–366.
[57] Bojan Kolosnjaji, Ambra Demontis, Battista Biggio, Davide Maiorca, Giorgio Giacinto, Claudia Eckert, and Fabio Roli. 2018. Adversarial malware binaries: Evading deep learning for malware detection in executables. In *Proc. of EUSIPCO*. 533–537.
[58] Moshe Kravchik and Asaf Shabtai. 2019. Efficient cyber attacks detection in industrial control systems using lightweight neural networks. arXiv:1907.01216
[59] Felix Kreuk, Yossi Adi, Moustapha Cissé, and Joseph Keshet. 2018. Fooling end-to-end speaker verification with adversarial examples. In *Proc. of ICASSP*.1962–1966.
[60] Felix Kreuk, Assi Barak, Shir Aviv-Reuven, Moran Baruch, Benny Pinkas, and Joseph Keshet. 2018. Adversarial examples on discrete sequences for beating whole-binary malware detection. arXiv:1802.04528
[61] Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. 2018. Adversarial examples for natural language classification problems. Unpublished Manuscript.
[62] Aditya Kuppa, Slawomir Grzonkowski, Muhammad Rizwan Asghar, and Nhien-An Le-Khac. 2019. Black box attacks on deep anomaly detectors. In *Proc. ofARES*. 21:1–21:10.
[63] A. Kuppa and N. A. Le-Khac. 2020. Black box attacks on explainable artificial intelligence (XAI) methods in cyber security. In *Proc. of IJCNN*. 1–8.
[64] Qi Lei, Lingfei Wu, Pin-Yu Chen, Alexandros G. Dimakis, Inderjit S. Dhillon, and Michael Witbrock. 2018. Discrete attacks and submodular optimization with applications to text classification. arXiv:1812.00151
[65] Jiangnan Li, Jin Young Lee, Yingyuan Yang, Jinyuan Stella Sun, and Kevin Tomsovic. 2020. ConAML: Constrained adversarial machine learning for cyber-physical systems. arXiv:3004.05631
[66] Yuanzhang Li, Yaxiao Wang, Ye Wang, Lishan Ke, and Yu-An Tan. 2020. A feature-vector generative adversarial network for evading PDF malware classifiers. *Information Sciences* 523 (2020), 38–48.
[67] Zilong Lin, Yong Shi, and Zhi Xue. 2018. IDSGAN: Generative adversarial networks for attack generation against intrusion detection. arXiv:1809.02077
[68] Xiaolei Liu, Xiaojiang Du, Xiaosong Zhang, Qingxin Zhu, Hao Wang, and Mohsen Guizani. 2019. Adversarial samples on Android malware detection systems for IoT systems. *Sensors* 19, 4 (2019), 974.
[69] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *Proc. of NDSS*.
[70] Zhengzhe Liu, Xiaojuan Qi, and Philip H. S. Torr. 2020. Global texture enhancement for fake face detection in the wild. In *Proc. of CVPR*.
[71] Jiang Ming, Zhi Xin, Pengwei Lan, Dinghao Wu, Peng Liu, and Bing Mao. 2015. Replacement attacks: Automatically impeding behavior-based malware specifications. In *Proc. of ACNS*. 497–517.
[72] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: An ensemble of autoencoders for online network intrusion detection. In *Proc. of NDSS*.
[73] Luis Muñoz González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proc. ofAISec*. 27–38.
[74] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles A. Sutton, J. Doug Tygar, and Kai Xia. 2008. Exploiting machine learning to subvert your spam filter. In *Proc. of LEET*.
[75] Ozan Özdenizci, Ye Wang, Toshiaki Koike-Akino, and Deniz Erdogmus. 2019. Adversarial deep learning in EEG biometrics. *IEEE Signal Processing Letters* 26 (2019), 710–714.
[76] O. M. Parkhi, A. Vedaldi, and A. Zisserman. 2015. Deep face recognition. In *Proc. of BMVC*.
[77] Jonathan Peck, Joris Roels, Bart Goossens, and Yvan Saeys. 2017. Lower bounds on the robustness to adversarial perturbations. In *Proc. ofNIPS*.
[78] Vinay Uday Prabhu and John Whaley. 2017. Vulnerability of deep learning-based gait biometric recognition to adversarial perturbations. In *Proc. of CV-COPS*.
[79] Shilin Qiu, Qihe Liu, Shijie Zhou, and Chunjiang Wu. 2019. Review of artificial intelligence adversarial attack and defense technologies. *Applied Sciences* 9 (March 2019), 909.

[80] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles K. Nicholas. 2018. Malware detection by eating a whole EXE. In *Proc. of AAAI Workshops*. 268–276.
[81] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. 2011. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security* 19, 4 (2011), 639–668.
[82] Maria Rigaki and Ahmed Elragal. 2017. Adversarial deep learning against intrusion detection classifiers. In *Proc. of NATO IST-152*.
[83] Gary Robinson. 2003. A statistical approach to the spam problem. *Linux Journal* 2003, 107 (Jan. 2003), 3.
[84] Ishai Rosenberg and Ehud Gudes. 2016. Bypassing system calls-based intrusion detection systems. *Concurrency and Computation: Practice and Experience* 29, 16 (Nov. 2016), e4023.
[85] Ishai Rosenberg and Shai Meir. 2020. Bypassing NGAV for fun and profit. In *Proc. of Black Hat Europe*.
[86] I. Rosenberg, S. Meir, J. Berrebi, I. Gordon, G. Sicard, and E. Omid David. 2020. Generating end-to-end adversarial examples for malware classifiers using explainability. In *Proc. ofIJCNN*. 1–10.
[87] Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. 2018. Query-efficient black-box attack against sequence-based malware classifiers. arXiv:1804.08778
[88] Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. 2019. Defense methods against adversarial examples for recurrent neural networks. arXiv:1901.09963
[89] Ishai Rosenberg, Asaf Shabtai, Lior Rokach, and Yuval Elovici. 2018. Generic black-box end-to-end attack against state of the art API call based malware classifiers. In *Proc. of RAID*. 490–510.
[90] Andrew Ross and Finale Doshi-Velez. 2018. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proc. of AAAI*.
[91] Joshua Saxe and Konstantin Berlin. 2015. Deep neural network based malware detection using two dimensional binary program features. In *Proc. of MALWARE*.
[92] Stefano Schiavoni, Federico Maggi, Lorenzo Cavallaro, and Stefano Zanero. 2014. Phoenix: DGA-based botnet tracking and intelligence. In *Proc. of DIMVA*. 192–211.
[93] Tegjyot Singh Sethi and Mehmed Kantardzic. 2018. Data driven exploratory attacks on black box classifiers in adversarial domains. *Neurocomputing* 289 (2018), 129–143.
[94] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proc. ofCCS*. 1528–1540.
[95] Hossein Shirazi, Bruhadeshwar Bezawada, Indrakshi Ray, and Charles Anderson. 2019. *Adversarial sampling attacks against phishing detection*. In *Data and Applications Security and Privacy XXXIII*. Lecture Notes in Computer Science, Vol. 11559. Springer, 83–101.
[96] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. 2020. Sponge examples: Energy-latency attacks on neural networks. arXiv:2006.03463
[97] Lior Sidi, Asaf Nadler, and Asaf Shabtai. 2019. MaskDGA: A black-box evasion technique against DGA classifiers and adversarial defenses. arXiv:1902.08909
[98] Sobhan Soleymani, Ali Dabouei, J. Dawson, and N.M. Nasrabadi. 2019. Defending against adversarial iris examples using wavelet decomposition. arXiv:1908.03176
[99] Sobhan Soleymani, Ali Dabouei, Jeremy Dawson, and Nasser M. Nasrabadi. 2019. Adversarial examples to fool iris recognition systems. arXiv:1906.09300
[100] Felix Specht, Jens Otto, Oliver Niggemann, and Barbara Hammer. 2018. Generation of adversarial examples to prevent misclassification of deep neural network based condition monitoring systems for cyber-physical production systems. In *Proc. of INDIN*. 760–765.
[101] Nedim Srndic and Pavel Laskov. 2014. Practical evasion of a learning-based classifier: A case study. In *Proc. of SP*. 197–211.
[102] Rock Stevens, Octavian Suciu, Andrew Ruef, Sanghyun Hong, Michael W. Hicks, and Tudor Dumitras. 2017. Summoning demons: The pursuit of exploitable bugs in machine learning. arXiv:1701.04739
[103] Jack W. Stokes, De Wang, Mady Marinescu, Marc Marino, and Brian Bussone. 2017. Attack and defense of dynamic analysis-based, adversarial neural malware classification models. arXiv:1712.05919
[104] Octavian Suciu, Scott E. Coull, and Jeffrey Johns. 2018. Exploring adversarial examples in malware detection. In *Proc. of ALEC*. 11–16.
[105] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. 2018. When does machine learning FAIL? Generalized transferability for evasion and poisoning attacks. In *Proc. of USENIX Security*. 1299–1316.
[106] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2014. Deep learning face representation from predicting 10,000 classes. In *Proc. ofCVPR*. 1891–1898.
[107] Shayan Taheri, Milad Salem, and Jiann-Shiun Yuan. 2019. RazorNet: Adversarial training and noise training on a deep neural network fooled by a shallow neural network. *Big Data and Cognitive Computing* 3 (July 2019), 43.

[108] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On adaptive attacks to adversarial example defenses. arxiv:2002.08347
[109] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction APIs. In *Proc. of USENIX Security*. 601–618.
[110] Martino Trevisan and Idilio Drago. 2019. Robust URL classification with generative adversarial networks. *ACM SIGMETRICS Performance Evaluation Review* 46, 3 (Jan. 2019), 143–146.
[111] Rakesh Verma and Keith Dyer. 2015. On the character of phishing URLs: Accurate and robust statistical learning classifiers. In *Proc. ofCODASPY*.
[112] B. Wang and N. Z. Gong. 2018. Stealing hyperparameters in machine learning. In *Proc. of SP*. 36–52.
[113] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proc. ofSP*.
[114] Bolun Wang, Yuanshun Yao, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. 2018. With great training comes great vulnerability: Practical attacks against transfer learning. In *Proc. of USENIX Security*. 1281–1297.
[115] Z. Wang. 2018. Deep learning-based intrusion detection with adversaries. *IEEE Access* 6 (2018), 38367–38384.
[116] Arkadiusz Warzynski and Grzegorz Kolaczek. 2018. Intrusion detection systems vulnerability on adversarial examples. In *Proc. of INISTA*. 1–4.
[117] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. 2018. Evaluating the robustness of neural networks: An extreme value theory approach. In *Proc. of ICLR*.
[118] Q. Xiao, K. Li, D. Zhang, and W. Xu. 2018. Security risks in deep learning implementations. In *Proc. of SPW*. 123–128.
[119] Peng Xu, Bojan Kolosnjaji, Claudia Eckert, and Apostolis Zarras. 2020. MANIS: Evading malware detection system on graph structure. In *Proc. of ACMAC*.
[120] Weilin Xu, David Evans, and Yanjun Qi. 2018. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proc. of NDSS*.
[121] Weilin Xu, Yanjun Qi, and David Evans. 2016. Automatically evading classifiers: A case study on PDF malware classifiers. In *Proc. of NDSS*.
[122] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan. 2012. Detecting algorithmically generated domain-flux attacks with DNS traffic analysis. *IEEE/ACM Transactions on Networking* 20, 5 (Oct. 2012), 1663–1677.
[123] Sandeep Yadav, Ashwath Kumar Krishna Reddy, A. L. Narasimha Reddy, and Supranamaya Ranjan. 2010. Detecting algorithmically generated malicious domain names. In *Proc. ofIMC*. 48–61.
[124] Shakiba Yaghoubi and Georgios Fainekos. 2019. Gray-box adversarial testing for control systems with machine learning components. In *Proc. ofHSCC*. 179–184.
[125] K. Yang, J. Liu, C. Zhang, and Y. Fang. 2018. Adversarial examples against the deep learning based network intrusion detection systems. In *Proc. of MILCOM*. 559–564.

Received July 2020; revised February 2021; accepted February 2021