Adversarial Attacks and Defenses in Intrusion Detection Systems: A Survey

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

Abstract

The world is becoming more digitized and inter-connected by the day and securing our digital infrastructure is not a topic we can take lightly anymore. Intrusion detection systems (IDSs) have been an integral part of the cybersecurity stack ever since their introduction in the 1980s. Traditionally such systems have relied on signatures and heuristics, however, recently growing demand for scalability, advances in computational power, and increasing dataset availability, have paved the way for machine learning approaches.

The challenge is that even though machine learning can do a better job at detecting intrusions in normal conditions – it itself is left vulnerable to adaptive adversaries who understand how these systems work and "think". In this survey we review the different kinds of attacks such an adversary can mount on IDSs, and perhaps more importantly, the various defenses available for making IDSs more robust. We start by proving some historic context on the matter and introducing the basic taxonomy of adversarial machine learning, before diving into the methods, attacks and defenses in the second part of the write-up.

Keywords: Machine Learning, Intrusion Detection Systems, Adversarial Attacks, Evasion Attacks, Poisoning Attacks.

1. INTRODUCTION

Machine learning in security is not new and has been used for a while. Examples include detecting malware on endpoints [1], [2], [3], cloud instance monitoring [4], [5], user behavior analysis [6], [7], [8], secure orchestration [9] and even automated data triaging in SIEM [10], [11], [12].

The same is true for intrusion detection systems (IDSs). IDSs are responsible for monitoring traffic either on the host device (host IDS or HIDS) or on the network itself (network IDS or NIDS) and recently there has been an explosion in research attempting to apply machine learning in IDS settings (Hindy et al. [13] identified over 80 separate research papers on the topic, written between 2008 and 2018).

It is a well-known fact, however, that machine learning (including deep learning) systems are not perfect and can in fact be exploited by adversaries. Previous research has shown that machine learning systems can be taken advantage of both during training [14], [15] and during test [16], [17] phases. We refer to the former as *poisoning* attacks – and the latter as *evasion* attacks (more on this in Section 3). Although most of the research, especially on the latter, has been conducted in the image realm – it's actually the security realm that faces the biggest challenge from such vulnerabilities.

To prevent such attacks from happening a parallel stream of research in the machine learning community has busied itself with defenses against adversarial attacks. For poisoning these primarily take the shape of detection of outliers [18] and robust learning [19]; for evasion the spectrum is much wider ranging from adversarial re-training [20] to gradient masking [21], input modification [22], detection [23], extra NULL class addition [24], and actual formal methods [25]. Not all of the above are applicable to IDSs as we will see in Section 5, but some indeed are.

The motivation for this paper to produce the first-ever survey discussing adversarial attacks and defenses on IDSs. There have exist surveys that review the IDS classifiers themselves [13], [26], [27], and even ones that review features [28] and branch out into SDN [29] – but none, as far as the authors are concerned, that review the adversarial angle of IDSs. It is worth nothing that because adversarial attacks and defenses on IDSs are so new, we are sometimes forced to go outside pure machine learning-based IDSs and review attacks and defenses conducted on traditional rule-base and statistical systems.

This paper is organized as follows: in Section 2 we introduce IDSs, their history, and the role of ML in modern IDS systems; in Section 3 we introduce the taxonomy around adversarial machine learning to be able to distinguish between different kinds of attacks; in Section 4 we dive deep into the various kinds of attacks on IDSs; and finally in Section 5 we touch on defenses. Section 5 in particular is a brief section due to fundamental lack of research in the area of adversarially robust IDSs, something we hope will change in the near future given the importance of these systems in enabling organizations to stay secure.

2. INTRUSION DETECTION SYSTEMS AND MACHINE LEARNING

2.1 Brief History of IDSs

The concept of IDSs has been around for almost 40 years now, beginning in 1980 with James Anderson's seminal paper, *Computer Security Threat Monitoring and Surveillance* [30]. Written for a government organization, Anderson’s work introduced the idea of audit trails and how they could be used for tracking misuse and malicious user behavior. It led to tremendous improvements in the auditing subsystems of virtually every operating system and seeded the idea of what an IDS might be.

The next big milestone in IDSs came from Dr. Dorothy Denning. Also working on a government project, Denning tried analyzing audit trails from government mainframe computers to create profiles of users based on their activities. In 1984 she helped develop the first model for intrusion detection, the Intrusion Detection Expert System (IDES), and later that same year published *An Intrusion Detection Model* [31], forming the basis for most work on IDS that followed.

Commercial development of IDSs began in early 1990s. The first company to market with a commercial IDS (HIDS, to be more specific) was Haystack Labs with their “Stalker” [32]. Others were SAIC with their “Computer Misuse Detection System (CMDS)” [33] and the Air Force's Cryptologic Support Center with their “Automated Security Measurement System (ASIM)”. In 1994 the ASIM project turned itself into a commercial company (the Wheel Group) and shipped the first-ever commercial NIDS, NetRanger [34].

2.2 Taxonomy of IDSs

There are two important axes on which IDSs differ – where they are deployed and what they look for. Starting with the former, IDSs can be deployed on either a singular host machine (in which case they are termed *host-based IDSs* or “*HIDS*”) – or they can be deployed on the network itself, overseeing packet flow between all the different endpoints (termed *network-based IDSs* or “*NIDS*”) (Figure 1). As we’ve seen above HIDS have been the first to be invented and deployed, while NIDS were a more recent development.

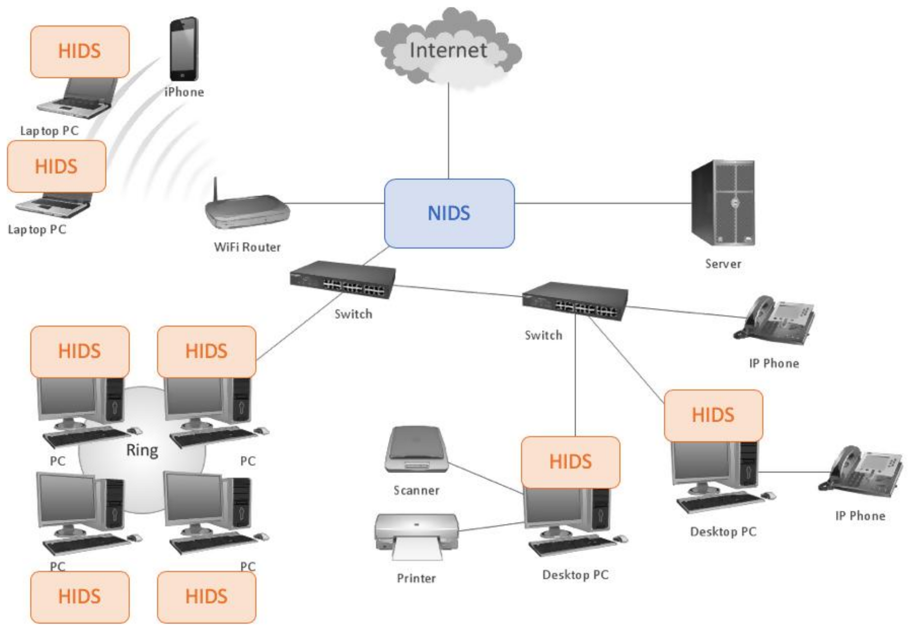


FIGURE 1: A graphical representation of a typical network and the role of HIDS and NIDS in it.

Each of course has its advantages and disadvantages. HIDS, for example, can be installed in isolation, is good at preventing inside intrusions, and is excellent at determining the extent of damage. NIDS, on the other hand, is useful for being a holistic solution saving time and cost of installation and is generally considered the go-to defense for outside intrusions [27]. Today many organizations will use both in combination.

The second axis on which IDSs differ is the way they operate. Broadly speaking IDSs break down into *signature-based* and *anomaly-based*. Similar to the anti-virus world, signature-based IDSs focus on finding known threats. Signatures are effectively fingerprints that allows IDSs to quickly scan a packet and determine if it is carrying anything harmful. Anomaly IDSs, as the name suggests focus on detecting unknown threats [27]. They generally are more sophisticated than signature-based systems, have recently evolved to use machine learning, and are hence the main focus of this survey.

2.3 Machine Learning in IDSs

One of the first works that tried to integrate machine learning into IDSs was this 2002 work by Mahoney and Chan [35]. As the authors themselves claim, their system “learns protocol vocabularies (at the data link through application layers) in order to detect unknown attacks that attempt to exploit implementation errors in poorly tested features of the target software”.

Machine-learning based IDSs can use a variety of features. One of the most popular datasets in the IDS community, KDD-99 [36] and its offspring NSL-KDD [37], for example have as many as 41 features available for modelling, each (Figure 2). Different works have leveraged different number of features, pre-processed in different ways in the past [38], [39], [40]. Other approaches have relied on end-to-end machine learning (typically deep learning) and avoided manual feature pre-processing altogether [41], [42].

No	Feature Name	No	Feature Name	No	Feature Name
1	duration	15	su_attemted	29	same_srv_rate
2	protocol_type	16	num_root	30	diff_srv_rate
3	Service	17	nu_file_creations	31	srv diff host rate
4	flag	18	num_shells	32	dst_host_count
5	src_bytes	19	num_access_file	33	dst_host_srv_count
6	dst_bytes	20	num outbound cmds	34	dst_host same_srv_rate
7	land	21	is_host_login	35	dst_host_diff_srv_rate
8	wrong_fragment	22	is_guest_login	36	dst_host_same_src_port_rate
9	urgent	23	count	37	dst_host_srv_diff_host_rate
10	hot	24	srv_count	38	dst_host_same_rerror_rate
11	num_failde_logins	25	serror_rate	39	dst_host_srv_serror_rate
12	logged_in	26	srv_serror_rate	40	dst_host_rerror_rate
13	num_compromised	27	rerror_rate	41	dst_host_srv_rerror_rate
14	root_shell	28	srv_rerror_rate	42	attack_type

FIGURE 2: KDD-99 and NSL-KDD feature list. Copied from [93].
Note that 42th attribute (attack_type) is not actually a feature, but rather the prediction vector.

Regarding machine learning algorithms applied in IDS, previous work leveraging virtually every type of algorithm exists. One of the most complete surveys in the space done by Hindy et al. [13] compared over 80 works on IDS using machine learning, and concluded that the most popular algorithms were artificial neural networks (19%), SVMs (14.4%), k-means (11.7%), decision trees (9.0%), Bayesian (9.0%), fuzzy logic (7.2%), and kNNs (5.4%). Full details can be seen in Figure 3.

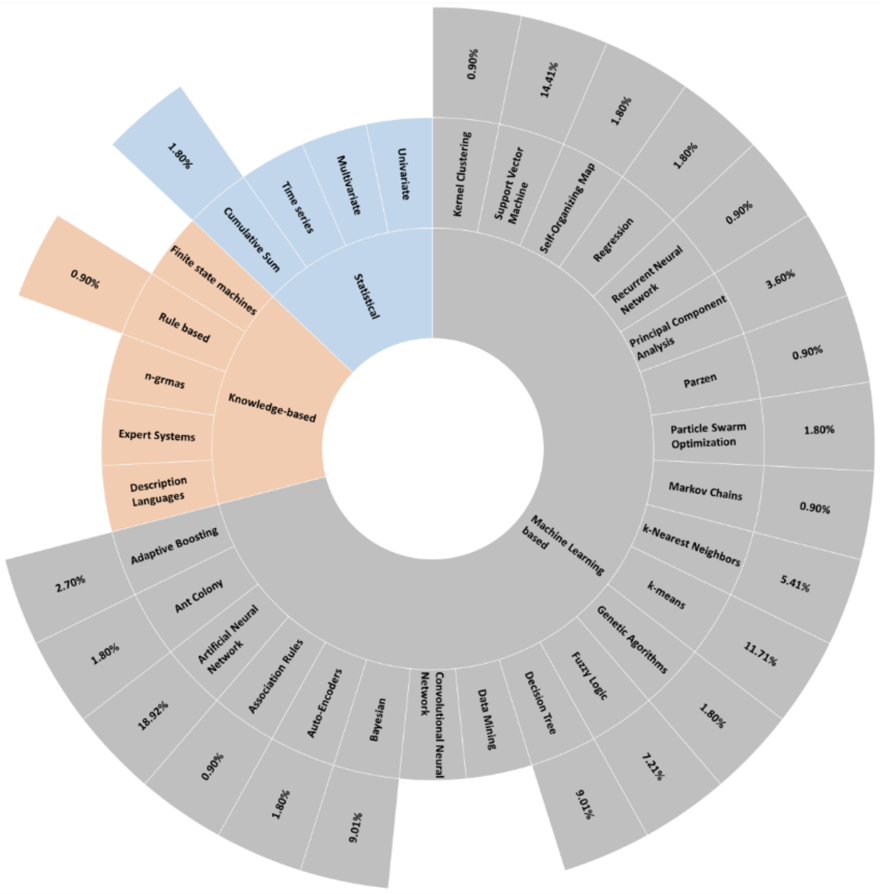


FIGURE 3: Algorithms examined in [13].
Note how the overwhelming majority of papers on IDSs published between 2008 and 2018 are machine-learning based (97.3%).

Despite the many research studies on machine learning-based IDSs, there nevertheless seems to be a glaring gap in terms of actual deployments. Sommer and Paxson [43] conducted an insightful review into the issue and concluded that this is due to five key factors:

1. *Machine learning is generally better at finding similarities than anomalies*. Even thought anomaly detection can be seen as a classification problem, if you're looking for *new* threats you by definition have a class that is under-represented
2. *IDSs have an outsized cost of errors*. False positives become very expensive very quickly, while false negatives can cause serious damage.
3. *There exists a semantic gap between results and their operational interpretations*. An "anomaly" doesn't necessary imply an attack – and it is difficult to separate malicious anomalies from benign.
4. *There exists enormous variability in input data*. Even within a single network, characteristics like bandwidth duration of connections, and application mix can exhibit immense variability – more so when viewed over short periods of time.
5. *Practical limitations preventing sound evaluation*. It is most difficult to obtain a functional network dataset due to sensitivity and so publicly available datasets tend to be either synthesized or anonymized – both not representative of production environments.

3. TAXONOMY OF ADVERSARIAL MACHINE LEARNING

Like in other areas of security, attacks on machine learning can be grouped based on a variety of properties. These include adversary's *goal*, *timing*, and *capability*.

When thinking about adversary's goals we borrow the popular security paradigm of *Confidentiality, Integrity, and Availability (CIA)* [44]. According to CIA, the attacker can have one of three goals:

1. *Confidentiality* – get access to sensitive, private information, such as the underlying dataset the machine learning system was trained on, or the model itself.
2. *Integrity* – violate the authenticity of the machine learning system, such as for example get it to misclassify an input without the host noticing.
3. *Availability* – take the machine learning system down entirely, which, for example, could be useful with IDSs, thus leaving the network undefended.

Timing refers to where in the machine learning pipeline [45] the attacker performs the attack. Machine learning is broadly speaking a two-step process (Figure 4). At first the system "trains" on a known body of training data, trying to learn the patterns hidden in that data (supervised learning) or understand relationships between various parts of the dataset (unsupervised learning). Later the system is put to a test and it issues predictions on new, previously unknown inputs. Depending on the task, those predictions can take the form of class labels (classification), continuous variables (regression), or cluster assignments (clustering).

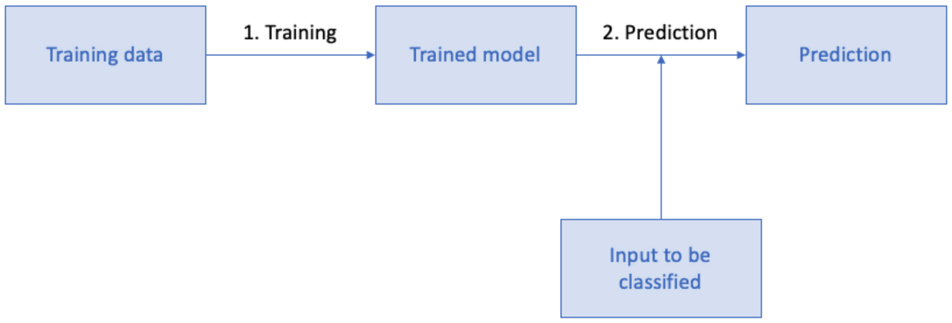


FIGURE 4: Simplified diagram depicting two steps of the machine learning process.

It turns out that both of these steps – training and prediction – are vulnerable to adversarial attacks. Attacks performed at training time aim to inject bad, malicious data into the training pool to teach the model something it shouldn't be learning [14], [15]. We refer to such attacks as poisoning attacks.

Attacks performed at test time make use of an intriguing property [16] machine learning systems seem to universally possess – they are easily fooled by inputs carefully perturbed in a particular direction [16], [17]. We refer to them as *evasion* attacks. The vulnerability has been discovered in 2013 by Szegedy et al. [16] when trying to understand how a neural network "thinks". Ever since there has been a cambrian explosion in "*adversarial example*" research, signifying community's interest in the security and robustness of machine learning systems (Figure 5).

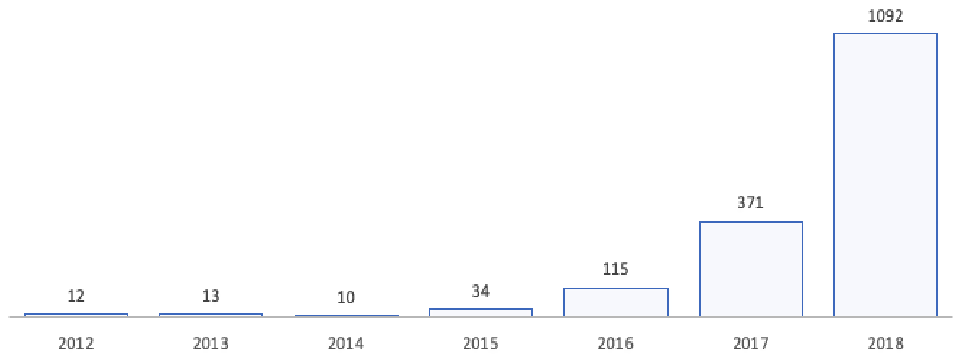
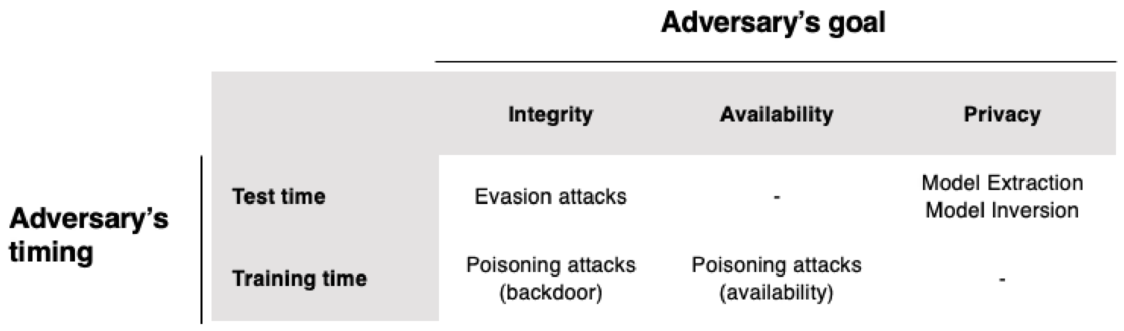


FIGURE 5: Number of "adversarial machine learning" papers released on arxiv [94].

One other type of attacks that can be performed at test time are *privacy* attacks [48], [49], [50] – those designed to elicit information from the machine learning system rather than directly influencing its predictions.

Taking attacker's goal and timing together we can arrive at the following output (Figure 6):



		Adversary's goal		
		Integrity	Availability	Privacy
Adversary's timing	Test time	Evasion attacks	-	Model Extraction Model Inversion
	Training time	Poisoning attacks (backdoor)	Poisoning attacks (availability)	-

FIGURE6: Summary of the types of attacks on machine learning classifiers.

Finally, capability refers to adversary's level of knowledge about the target machine learning system. Although scientific literature tends to vary in how they use the below terms [45], [46], [47], there seems to be broad agreement that there are three possible levels of knowledge:

- *WhiteBox* refers to the type of adversary who has detailed understanding of the internals of the targeted machine learning model. This typically involves all of the following: model architecture, model weights, and training data sample or at least training data distribution.
- *BlackBox* refers to the type of adversary who doesn't possess the above knowledge. BlackBox adversaries have to mount their attacks having no more access than the classifier's output. If they receive only the class label from the classifier the attack is known as *hard-label*, otherwise, if they also get access to a confidence score, it is known as *confidence* attack.

- *GrayBox*, perhaps the most ambiguous out the three, refers to adversaries who sit somewhere in the middle. Such an attacker might be able to query the model and hence get its output, but also have knowledge of the model's feature space and distribution of the underlying dataset it was trained on.

In this survey we focus on both evasion and poisoning attacks, however, we leave out privacy attacks on IDSs. Where possible, we try to note down whether the attack is done in a WhiteBox or BlackBox fashion, as this greatly influences the power of the attack.

4. ATTACKS ON INTRUSION DETECTION SYSTEMS

Like we mentioned in the introduction, because of the relative scarcity of adversarial attacks works on IDSs we start by reviewing attacks on pre-machine learning IDSs, ie ones based on signatures / statistical properties. Such attacks also provide useful historical context. Later, we move to discuss evasion attacks on machine learning-based classifiers and finish the section with a review of poisoning research.

4.1. Evasion Attacks on Non-Machine Learning-Based IDSs

TCP/IP transformation attacks are ones that leverage the intricacies of TCP/IP protocol design to prevent IDS from doing its job correctly. It is not hard to imagine that IDSs have gained most popularity with the development of the internet [51] – and hence the first serious attacks focused strictly on properties related to its protocol design [52], [53], [54], [55].

Chung et al. [52] mounted one of the first TCP/IP transformation attacks in 1995 (just 1 year after the first commercial IDS was released). In this pioneering work, authors showed how to evade detection by distribution malicious activity over several sessions.

Paxson [53] was the first to introduce a formal classification of evasion attacks on IDSs. Specifically, they introduced 3 attacks:

1. *Overload attack*, which overburdens the monitor to the point where it fails to keep up with the data and start to drop packets.
2. *Crash attack*, which knocks the monitor out of action by causing it to fault or run out of resources. This attack type is much more subtle than the overload type, but also requires more work by the attacker to figure out a flaw in the monitor's design.
3. *Subterfuge attack*, which perturbs the input in a way that prevents the monitor from seeing it. For example, in the paper the authors inserted 'NUL' into the text in the hope that the monitor would miss everything after it.

The first two can also be seen as first variations of the modern DoS attack (also below), while the third is a classical evasion TCP/IP transformation.

Ptacek and Newsham [54] have introduced their own classification of TCP/IP transformation attacks:

1. *Insertion attack*: because an IDS will accept every packet but the end system will not, an attacker can "insert" packets that only the IDS sees. eg they can convert "\GET/cgi-bin/phf?" into "\GET/cgi-bin/pleasedontdetecttthisforme?" by sending "leasedontdetectt" and "is" and "orme?" to the IDS.
2. *Evasion attack*: opposite of insertion, here the attacker makes sure some packets don't reach the IDS. Taking the above example, the attacker might split "\GET/cgi-bin/phf?" into "\GET/cgi-bin/p" and "hf?", and as long as IDS receives only one of the two, it won't pick up on the intrusion.There are many different ways (1) and (2) can be done both at IP and TCP layers, mainly due to the reassembly property of TCP.
3. *DOS attack* can be used to take the IDS down and leave the network "fail-open" - eg through resource exhaustion.

The authors tested the above three attacks against 4 commercial IDSs (WheelGroup'sNetRanger, InternetSecuritySystems'RealSecure, AbirNet'sWall-3, NetworkFlightRecorder'sNFR) – and found them all to be vulnerable. They concluded by saying it’s unclear how to fix some of the vulnerabilities explored without building a second layer of protection for the IDS itself.

Handley et al. [55], similarly to Ptacek and Newsham [54] explore the idea that the IDS can see the network very different to how the end system does – and that can be leverage by adversaries to their advantage. They discuss the differences at the network topology and exact protocol implementation levels and mention how attackers can leverage such network properties as IP fragmentation, TTL, and overlapping IP fragments.

Next class of attacks are *mimicry* attacks. Such attacks involve the attacker taking a malicious packet and modifying parts of it to “mimic” a benign version. For example, an attacker might want to include specific strings or functions, or API calls known to be associated with benign version in an attempt to hide the true malicious nature of its payload.

Wagner and Soto [56] successfully mounted one of the first mimicry attacks on a HIDS. They used WhiteBox access to a HIDS to understand what it was looking for, and then manually perturbed the file in order to evade detection. Their method included replacing system calls with their own and inserting plenty of no-op calls to obfuscate real, malicious activity.

Tan et al. [57] examine two different HIDSs based on system calls and given a fairly restrictive "normality" model successfully devised versions of several exploits that mimicked benign files and hence escaped detection. In addition, they demonstrated a way of masquerading the attack to look like adifferent (less dangerous) one. The latter is particularly interesting as it opens up the possibility of not just evasion, but using adversarial attacks as a red herring. Similar work has been done in [58].

Next we examine *polymorphic blending* attacks. *Polymorphism* describes malware that changes its shape from instance to instance; and is typically designed in such a way to evade detection by signatures. *Blending*, on the other hand, is effectively the same as mimicry: malware that attempts to look benign by copying attributes of benign versions. Put together, thus, polymorphic blending refers to malware that both differs from sample to sample, and blends in with “normal” traffic as not to be picked up by the anomaly IDS.

The idea of polymorphic blending was pioneered by Fogla et al. [59]. The authors launch a polymorphic blending attack on a byte distribution classifier [60] that, as the name suggests, looks at frequency of occurrence of various n-grams and decides if that frequency is "normal". Their attack involves three steps:

1. Decide on the right packet size that is not abnormal given the target network’s packet size distribution. Then break their payload into the respective number of packets.
2. Develop a substitution algorithm that learns to "encrypt" characters in the payload to make their distribution look benign. For example, normal network traffic mainly has mainly printable characters, while malicious payload doesn't. The right substitution table (found using the algorithm) fixes this.
3. Pad the packets with some terminal bytes to make them blend with normal traffic still more.

The authors show their attack is successful at evading the target IDS (they used a pure polymorphic attack as baseline, which as expected fails, as it doesn’t “blend” and hence is an anomaly).

In an extension of [59]’s work, Kolesnikov and Lee [61] built their own "blender" that takes in normal traffic profile and spits out an evasive variant. Similar to [59], they use a substitution table to achieve this. The authors successfully tested their attack on 4 different IDSs.

The idea of polymorphic blending is further formalized in [62], which also presents a framework for automatic generation of such attacks.

4.2 Evasion Attacks on Machine Learning-Based IDSs

We will start with the discussion of *WhiteBox* attacks. Remember from the Section 3 that these attacks are the ones where the adversary has the highest level of knowledge possible – typically including model's gradients. This allows them to mount attacks that directly optimize the model's loss function, except the other way around (gradient ascent). A lot of these attacks have been invented in the computer vision realm and hence we'll see a pattern where the algorithms used to execute them are often directly transcribed from there.

Huang et al. [63] launched 3 separate WhiteBox attacks on 3 types of deep learning based IDSs (MLP, CNN, LSTM). The authors used the infamous FGSM [17] and JSMA [47] algorithms from vision. The most serious damage was done by FGSM, which managed to reduce LSTM's accuracy from 98% to 42% (ie over half).

Rigaki and Elragal [64] attacked an SVM, decision tree and random forest using adversarial examples generated on a neural network, and then transferred [75]. They too attempted FGSM [17] and JSMA [64], but concluded that the formal was unusable (modified too many features). JSMA, on the other hand, altered on average just 6% of the features of the packets to successfully evade detection. As a result the authors managed to reduce the accuracy of the classifiers from 5 to 28%, with random forest coming out as the most robust classifier.

Clements et al. [65] attempted to attack a deep learning-based NIDS system with 4 different attack algorithms, borrowed from vision: FGSM [17], JSMA [64], C&W [67], and ENM [68]. They performed two sets of attacks – first set targeting integrity, second availability (as discussed in Section 3) – and managed to find adversarial examples using all 4 methods in 100% of cases. Moreover, the best method, ENM, managed to generate evasive variants with as few as 1.38 features perturbed.

Wang [66] also explored attacks from the vision realm on IDSs. Specifically, they mounted JSMA [64], FGSM [17], LL-FGSM [69], Random FGSM [70], DeepFool [71], and C&W [67] attacks on the infamous NSL-KDD dataset [37]. They found that JSMA required the least features to be perturbed to generate an effective attack, however, all algorithms proved successful in empirical evaluation.

We now move to *BlackBox* attacks. Like we discussed in Section 3, BlackBox attacks are much more restrictive than WhiteBox. The adversary typically has far less information to work with and hence can optimize their attacks a lot less. This is not to say that BlackBox attacks are not powerful, in fact, some of the simplest yet most powerful attacks in computer vision fall into this bucket [68], [72], [73], [74].

The definitive study of BlackBox attacks in IDSs was done by Rigaki and Garcia [76]. The authors explored the idea of modifying network traffic post-exploitation (during C&C) to avoid detection by the IDS. They built a generative adversarial network (GAN) using benign Facebook chat traffic, placed it on a separate server, and modified the malware to periodically request new network behavior prototype it should follow. After just 400 training epochs they managed to successfully guide the malware to evade detection 100% of the time. The study is still the more significant because of how little data it took to train the GAN – just 200 network flows.

4.3 Poisoning Attacks on IDSs

One of the first attempts at poisoning IDSs was performed by Chung and Mok [77]. The authors developed an "allergy" attack against the Autograph worm signature generation system [82]. Autograph operates in two phases: 1)it finds suspicious nodes in the network based on their behavior, and 2)it then observes those nodes and their traffic, and turns those into blocking rules.

The allergy attack 1)convinces the Autograph that the node is indeed infected, and 2)after the Autograph learnt to recognize a certain kind of traffic as malicious based on that node, sends exactly that kind of traffic. This is an example of a poisoning attack targeting availability.

While Chung and Mok [77] tried poisoning a signature-based IDS, Nelson and Joseph [78] were the first to look at poisoning a machine learning-based IDS. Their work, although lacking empirical evaluation, came to a surprisingly positive conclusion: the attacker needed exponential amounts of data to subvert a learning algorithm. It can be argued, however, that the authors used an unrealistic assumption of the classifier retraining with an infinite training window (instead of sliding). A more realistic assumption would be to adjust to non-stationarity of data.

In [19] Rubinstein et al. wanted to poison a classifier to make it miss an upcoming DoS attack. They did this by injecting useless data (“chaff”) into its training pool beforehand. In general, the authors were able to achieve high level of evasion for the DoS attack, eg 28% with as little as 10% chaff inserted. They also looked at the impact of the level of adversary’s knowledge and their timing strategy on the success of the attack.

Kloft and Laskov [80] attempted a poisoning attack on a centroid-based clustering algorithm used by an anomaly IDS. They randomly chose an exploit and a normal training set out of the prepared data and ranpoisoning attacks until success, ie until its vector was accepted by the anomaly detector. They found that in order to stage a successful poisoning attack the attacker needs to control 35% of the training pool (when they're able to ingest an arbitrary large number of points) or 5-15% of all traffic (when they're restricted to only a certain number of samples).

Finally, Biggio et al. [81] performed 3 different attacks – one of them backdoor poisoning – on an IDS system. Each network packet was considered an individual sample to be labeled as normal (legitimate) or anomalous (malicious) and represented as a 256-dimensional feature vector. They assumed the attacker had knowledge of the features used by the system, but no access to the detector itself (neither any knowledge of how it works – ie BlackBox scenario).Samples injected were on purpose made to emulate other existing network traffic (same histogram of payload’s bytes), so that they are not excluded as anomalous during detector re-training. As a result, the authors managed to more than half the accuracy of the system.

The overall picture that we hope this section has manage to portray is that of a “well and alive” world of adversarial research in the field of IDSs. This is not unexpected; IDSs are one of the most important components of modern security systems and it is paramount that we think about how an adaptive adversary may attempt to make use of them.

5. DEFENSES FOR INTRUSION DETECTION SYSTEMS

We now switch to our final topic, discussing what can be done to make machine learning-based IDSs more secure. Unlike in other areas of machine learning - such as computer vision, NLP, speech and even malware classification - the adversarial defense landscape for IDSs is rather scarce. This is only fair because there are far fewer attack papers as well. Nevertheless, below is our best attempt at summarizing existing work on protecting IDSs against adversarial attacks - split the same way as the attacks section - into evasion and poisoning.

5.1 Defenses Against Evasion Attacks on IDSs

The first defense type we will discuss is *detection*. Most of the worms today are polymorphic - they use an encryptor / decryptor to hide the actual shellcode behind a randomly generated key. There are, however, certain components, that are left unencrypted - such as the decryptor itself. Those can be used to detect even the most evasive types of worms.

This idea formed the basis for the work done by Akritidis et al. [83]. The authors designed a heuristic defense for IDSs that relies on detecting “sleds” - a common feature of buffer overflow attacks. Very simply a “sled” is a sequence of NOP commands that the attacker needs to have as part of their malicious payload in order to successfully execute a buffer overflow attack. Although their work was not the first to introduce the idea of looking for sleds [84], [85] – their defense was a major improvement over previous attempts. Specifically, their system, called STRIDE, detected several types of sleds that other techniques were blind to, demonstrated a lower false positive rate, and proved to be more computationally efficient.

Another defense type, *normalization*, was introduced and attempted in [55]. The authors explained that the biggest challenge faced by IDSs is that they sometimes see the network very differently to how the end system does. Indeed, we discussed a number of TCP/IP transformation attacks in Section 4 that made use of this property [54], [55]. The proposed defense, thus, attempts to level those differences. The authors propose a normalization component that prefaces the IDS and tries to resolve any ambiguities and differences between how the IDS and the end system might view the network packets. The biggest challenges with this approach are the “cold start” problem (when the normalizer goes online it has no "state" to compare to) and the fact that the normalizer itself can fall victim to an adaptive adversary.

In [60] Wang introduce *randomized modeling / testing* as a defense against evasion (mimicry) attacks [56], [57], [58]. Under randomized modeling / testing, each monitor randomly partitions the payload before running any analysis on it. The intuition is that because the mimicry attacker doesn’t know how their payload will be partitioned and analyzed, it’s much harder for them to simply “pad” the payload to make it blend with normal traffic. Thus even if the attacker has global knowledge of the target network and IDS - the randomization component makes naïve mimicry virtually impossible. Similar randomization-based defense against polymorphic blending was discussed in [59]. This type of defense of course only works until the attacker discovers the randomization seed – yet another reminder of the “*cat and mouse*” nature of the security game.

Finally, we discovered one work that attempted a version of adversarial training against attacks on IDSs [86]. In general, we found it surprising that adversarial training received so little attention in the IDS community, given that it is hands down the most popular defense in the rest of the adversarial examples literature [20], [87], [88], [89].

The authors in [86] attribute the problem of poor IDS performance to limited and imbalanced datasets. Their solution is a form of data augmentation (can be thought of as adversarial training).They first generate synthesized intrusion data using Monte Carlo method and then use that data to augment the learning of the IDS. Eventually the authors prove that their framework outperforms existing learning based IDSs in terms of improved accuracy, precision, recall, and F1-score.

5.2 Defenses Against Poisoning Attacks on IDSs

Before we discuss defenses against poisoning, let us briefly touch on why machine learning models are even susceptible to it. The much too simplified answer is that in order to function and be able to “learn” anything at all, machine learning models have to make assumptions. These assumptions exist at two levels: the *model* level and the *data* level.

Modelling assumptions are those that involve the classifier model and its representation of the data. One example of this is data linearity, where the model assumes that there indeed exists a linear function that can correctly represent the learning task. Other examples include data separability (there indeed exists a boundary that can correctly separate the two or more classes) and feature independence (each feature is an independent indicator of a latent variable behind the true class of an object).

Data assumptions are assumptions about data’s distribution. Probably the most common assumption made is that of data stationarity - ie that train and test datasets follow the same data distribution. Another popular one is that each data point is independent and identically distributed (i.i.d).

Poisoning attacks intentionally violate the above assumptions. For example in [79] the authors craft spam emails that come from a different data distribution than the ones used for training (data stationarity), while in [19] the authors attack an anomaly IDS by breaking the i.i.d assumption (only a subset of the data is perturbed, making it non-identical).Defenses against poisoning attacks try to work around the above assumptions, to mitigate their impact. Broadly speaking poisoning defenses fall into one of two categories - data sanitization and robust learning.

Data sanitization involves scrutinizing incoming data before adding it to the training pool for the model. The goal is to understand whether newly added data is likely to be adversarial, and hence exclude it before it can poison the model. The term “data sanitization” in the context of machine learning was first used by Cretu et al. in their 2008 seminal paper [90]. The authors used the information about when each datapoint was added to the dataset as an indication of how likely it is to be poisonous. The intuition, as the authors themselves describe it, is such that “in a training set spanning a sufficiently large time interval, an attack or an abnormality will appear only in small and relatively confined time intervals”.

Another popular defense that falls under the sanitization umbrella is the *Reject On Negative Impact (RONI)* technique [18]. It measures the empirical effect of adding each training instance and discards instances that have a substantial negative impact on classification accuracy. The defender first trains a model with the new inputs excluded, then a second copy with the new in-question inputs included. They then compare the accuracy on the two and if adding the candidate instance damages the accuracy, it is rejected. The authors applied this defense to a spam classifier and found it to be very successful. Specifically, they managed to improve classifier’s spam detection from 80% to 87%, while keeping non-spam detection rate consistently high above 95%.

In the field *of robust learning* it is assumed that the bulk of the data originates from a known, trustworthy source, but that a part of the data comes from an unknown and assumed to be adversarial source. The goal then is to limit the impact of this unknown source. As such, the authors in [19] looked at improving the robustness of a principal component analysis (PCA)-based IDS. They postulated that because PCA is known to be strongly affected by outliers [91] – a better algorithm is needed to protect against poisoning. They hence examined the PCA-Grid algorithm proposed by Croux et al. [92] and found that it substantially reduced the effect of outliers and was able to reject poisonous training data successfully.

6. CONCLUSION

Digitization is the undisputed trend of our generation. As more devices come online and the world turns into a one big computer, cybersecurity will only become more important. Traditional approaches to security, such as signatures and heuristics, don’t scale well and will soon likely be replaced by more automated techniques, such as machine learning.

In this survey we looked at machine learning and its role in intrusion detection. We gave a brief overview of IDSs history and how machine learning came to be in the space. Most importantly, however, we looked at how machine learning IDSs themselves can be taken advantage of by adaptive adversaries – and how they can be made robust to such attacks.

We are *cautiously optimistic* about the future – we believe machine learning holds great promise for improved security including that of networks, however, we also believe more work is needed in the area of adversarial attacks – and especially defenses – for IDSs.

7. REFERENCES

[1] “CrowdStrike Introduces Enhanced Endpoint Machine Learning Capabilities and Advanced Endpoint Protection Modules.” Internet: https://www.crowdstrike.com/resources/news/crowdstrike-introduces-enhanced-endpoint-

machine-learning-capabilities-and-advanced-endpoint-protection-modules/, Feb. 13, 2017 [Jul. 24, 2019].

[2] M. Berninger and A. Sopan. “Reverse Engineering the Analyst: Building Machine Learning Models for the SOC.” Internet: https://www.fireeye.com/blog/threat-research/2018/06/build-machine-learning-models-for-the-soc.html, Jun. 05, 2018 [Jul. 24, 2019].

[3] “How does Symantec Endpoint Protection use advanced machine learning?” Internet: https://support.symantec.com/us/en/article.howto125816.html, Apr. 24, 2019 [Jul. 24, 2019].

[4] “GuardDuty Intelligent Threat Detection AWS.” Internet: https://aws.amazon.com/guardduty/, 2018 [Jul. 24, 2019].

[5] R. Ronen. “Machine Learning in Azure Security Center.” Internet: https://azure.microsoft.com/en-us/blog/machine-learning-in-azure-security-center/, Jan. 28, 2016 [Jul. 24, 2019].

[6] “Machine Learning Analytics app.” Internet: https://www.ibm.com/support/knowledgecenter/en/SS42VS_7.2.8/com.ibm.UBAapp.doc/c_Qapps_UBA_ML_intro.html, nd [Jul. 24, 2019].

[7] “RSA NetWitness UEBA.” Internet: https://www.rsa.com/content/dam/en/data-sheet/rsa-netwitness-ueba.pdf, 2018 [Jul. 24, 2019].

[8] “SIEM - Security Information and Event Management.” Internet: https://www.splunk.com/en_us/cyber-security/siem-security-information-and-event-management.html, nd [Jul. 24, 2019].

[9] “Use Cases: Demisto’s Top Machine Learning Use Cases – Part 1.” Internet: https://blog.demisto.com/demistos-top-machine-learning-use-cases-part-1, Feb. 20, 2018 [Jul. 24, 2019].

[10] “Artificial Intelligence for Smarter Cybersecurity.” Internet: https://www.ibm.com/security/artificial-intelligence, nd [Jul. 24, 2019].

[11] “Big Data Analytics for Advanced Security.” Internet: https://logrhythm.com/solutions/security/security-analytics/, nd [Jul. 24, 2019].

[12] “Cognito Detect is the most powerful way to find and stop cyberattackers in real time.” Internet: https://content.vectra.ai/rs/748-MCE-447/images/ProductCompanyOverview_2019_Cognito_Detect_AIpowered_attacker_detection_English.pdf, nd [Jul. 24, 2019].

[13] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson and X. Bellekens. (2018, Jun.). “A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets.” *arXiv Preprint - arXiv:1806.03517*. [On-line]. Available: https://arxiv.org/abs/1806.03517 [Jul. 24, 2019].

[14] B. Biggio, B. Nelson and P. Laskov. (2013, Mar.). “Poisoning Attacks against Support Vector Machines.” *arXiv Preprint - arXiv:1206.6389*. [On-line]. Available: https://arxiv.org/abs/1206.6389 [Jul. 24, 2019].

[15] B. Biggio, K. Rieck, D. Ariu, C. Wressnegger, I. Corona, G. Giacinto and F. Roli. “Poisoning Behavioral Malware Clustering,” in *Proc. AISec’14 - Workshop on Artificial Intelligent and Security Workshop*, 2014, pp. 27-36.

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

[17] I.J. Goodfellow, J. Shlens and C. Szegedy. (2014, Dec.). “Explaining and Harnessing Adversarial Examples.” *arXiv Preprint - arXiv:1412.6572*. [On-line]. Available: https://arxiv.org/abs/1412.6572 [Jul. 24, 2019].

[18] B. Nelson, M. Barreno, F.J. Chi, A.D. Joseph, B.I.P. Rubinstein, U. Saini, C. Sutton, J.D. Tygar and K. Xia. “Misleading Learners: Co-opting Your Spam Filter,” in *Machine Learning in Cyber Trust*. Boston: Springer, 2009, pp. 17-51.

[19] B.I.P. Rubinstein, B. Nelson, L. Huang, A.D. Joseph, S. Lau, S. Rao, N. Taft and J.D. Tygar. “ANTIDOTE: understanding and defending against poisoning of anomaly detectors,” in *Proc. IMC’09 - 9th ACM SIGCOMM conference on Internet Measurement*, 2009, pp. 1-14.

[20] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh and P. McDaniel. (2017, May.). “Ensemble Adversarial Training: Attacks and Defenses.” *arXiv Preprint - arXiv:1705.07204.* [On-line]. Available: https://arxiv.org/abs/1705.07204 [Jul. 24, 2019].

[21] N. Papernot, P. McDaniel, X. Wu, S. Jha and A. Swami. (2015, Nov.). “Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks.” *arXiv Preprint - arXiv:1511.04508.* [On-line]. Available: https://arxiv.org/abs/1511.04508 [Jul. 24, 2019].

[22] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu and J. Zhu. (2017, Dec.). “Defense against Adversarial Attacks Using High-Level Representation Guided Denoiser.” *arXiv Preprint - arXiv:1712.02976*. [On-line]. Available: https://arxiv.org/abs/1712.0276 [Jul. 24, 2019].

[23] N. Akhtar, J. Liu and A. Mian. “Defense against Universal Adversarial Perturbations,” in *Proc. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3389-98.

[24] H. Hosseini, Y. Chen, S. Kannan, B. Zhang and R. Poovendran. (2017, Mar.). “Blocking Transferability of Adversarial Examples in Black-Box Learning Systems.” *arXiv Preprint - arXiv:1703.04318*. [On-line]. Available: https://arxiv.org/abs/1703.04318 [Jul. 24, 2019].

[25] G. Katz, C. Barrett, D.L. Dill, K. Julian and M.J. Kochenderfer. “Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks,” in *Proc. CAV 2017 International Conference on Computer Aided Verification*, 2017, pp. 97-117.

[26] T. Hamed, J.B. Ernst and S.C. Kremer. “A Survey and Taxonomy of Classifiers of Intrusion Detection Systems,” in *Computer and Network Security Essentials.* K. Daimi, Ed. Cham: Springer, 2018, pp. 21-39.

[27] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis and R. Atkinson. (2017, Jan.). “Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey.” *arXiv Preprint - arXiv:1701.02145*. [On-line]. Available: https://arxiv.org/abs/1701.02145 [Jul. 24, 2019].

[28] P.R.K. Varma, V.V. Kumari and S.S. Kumar. “A Survey of Feature Selection Techniques in Intrusion Detection System: A Soft Computing Perspective,” in *Progress in Computing, Analytics and Networking*. P.K. Pattnaik, S.S. Rautaray, H. Das and J. Nayak, Eds. Singapore: Springer, 2018, pp. 785-93.

[29] T.N. Nguyen. (2018, Apr.). “The Challenges in SDN/ML Based Network Security : A Survey.” *arXiv Preprint - arXiv:1804.03539*. [On-line]. Available: https://arxiv.org/abs/1804.03539 [Jul. 24, 2019].

[30] J.P. Anderson. “Computer Security Threat Monitoring and Surveillance,” in *Technical Report: James P. Anderson Company*. Fort Washington: James P. Anderson Company, 1980.

[31] D.E. Denning. “An Intrusion Detection Model.” *IEEE transactions on Software Engineering,* vol. SE-13(No 2), pp. 222-32, Feb. 1987.

[32] S. Smaha. “Computer Misuse and Anomaly Detection.” Internet: http://seclab.cs.ucdavis.edu/projects/cmad/4-1996/pdfs/Smaha.pdf, Nov. 1996 [Jul. 24, 2019].

[33] H. Kvarnström. “A survey of commercial tools for intrusion detection,” in *Technical Report 99-8.* Göteborg: Chalmers University of Technology, Oct. 1999.

[34] P. Innella. “The Evolution of Intrusion Detection Systems.” Internet: https://www.symantec.com/connect/articles/evolution-intrusion-detection-systems, Nov. 16, 2001 [Jul. 24, 2019].

[35] M.V. Mahoney and P.K. Chan. “Learning nonstationary models of normal network traffic for detecting novel attacks,” in *Proc. KDD’02 – 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 376-85.

[36] “KDD Cup 1999 Data.” Internet: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, Oct. 28, 1999 [Jul. 29, 2019].

[37] L. Dhanabal and S.P. Shantharajah. “A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms.” *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 4(6), pp. 442-56, Jun. 2015.

[38] C. Kruegel, D. Mutz, F. Valeur and G. Vigna. “On the Detection of Anomalous System Call Arguments,” in *Proc. Computer Security – ESORICS 2003*, 2003, pp. 326-43.

[39] M.V. Mahoney and P.K. Chan. “An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection,” in *Proc. RAID 2003: Recent Advances in Intrusion Detection,* 2003, pp. 220-37.

[40] W. Lee and S.J. Stolfo. “A framework for constructing features and models for intrusion detection systems.” *ACM Transactions on Information and System Security (TISSEC),* vol. 3(4), pp. 227-61, Nov. 2000.

[41] L.O. Anyanwu, J. Keengwe and G.A. Arome. “Scalable Intrusion Detection with Recurrent Neural Networks,” in *Proc. 7th International Conference on Information Technology: New Generations,* 2010, pp. 919-23.

[42] J. Kim, J. Kim, H.L.T. Thu and H. Kim. “Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection,” in *Proc. 2016 International Conference on Platform Technology and Service (PlatCon)*, 2016, pp. 1-5.

[43] R. Sommer and V. Paxson. “Outside the Closed World: On Using Machine Learning For Network Intrusion Detection,” in *Proc. 2010 IEEE Symposium on Security and Privacy*, 2010, pp. 305-16.

[44] A. Summers and C. Tickner. “Introduction to security analysis.” Internet: https://www.doc.ic.ac.uk/~ajs300/security/index.html, nd. [Jul. 24, 2019].

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

[46] N. Akhtar and A. Mian. (2018, Feb.). “Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey.” *IEEE Access.* [On-line] 6, pp. 14410-30. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8294186 [Jul. 24, 2019].

[47] N. Papernot, P.D. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik and A. Swami. “The Limitations of Deep Learning in Adversarial Settings,” in *Proc. 2016 IEEE European Symposium on Security and Privacy (EuroS&P),* 2016, pp. 372-87.

[48] C. Song and V. Shmatikov. “Auditing Data Provenance in Text-Generation Models,” in *Proc. KDD’19 - 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining,* 2019, pp. 196-206.

[49] S. Truex, L. Liu, M.E. Gursoy, L. Yu and W. Wei. (2019, Feb.). “Demystifying Membership Inference Attacks in Machine Learning as a Service.” *IEEE Transactions on Services Computing*. [On-line] Available: https://ieeexplore.ieee.org/document/8634878 [Jul. 24, 2019].

[50] A. Pyrgelis, C. Troncoso and E. De Cristofaro. “Knock Knock, Who's There? Membership Inference on Aggregate Location Data,” in *Proc. NDSS 2018 - 25th Network and Distributed System Security Symposium,* 2018.

[51] L. Kleinrock. “Information Flow in Large Communication Nets.” Proposal for Ph.D. Thesis. Cambridge: Massachusetts Institute of Technology, May. 1961.

[52] M. Chung, N.J. Puketza, R.A. Olsson and B. Mukherjee. “Simulating Concurrent Intrusions for Testing Intrusion Detection Systems: Parallelizing Intrusions,” in *Proc. 18th National Information Systems Security Conference*, 1995, pp. 173-83.

[53] V. Paxson. “Bro: A System for Detecting Network Intruders in Real-Time.” *Computer Networks*, vol. 31(23-4), pp. 2435-63, Dec. 1999.

[54] T.H. Ptacek and T.N. Newsham. “Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection.” Internet: https://pdfs.semanticscholar.org/fa48/7f1a3e173a8178502eae14a9f9b431eaf62a.pdf?_ga=2.252024270.655035103.1568508389-933039622.1562710349, Jan. 1998 [Jul. 24, 2019].

[55] M. Handley, V. Paxson and C. Kreibich. “Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics,” in *Proc. 10th USENIX Security Symposium*, 2001.

[56] D.A. Wagner and P. Soto. “Mimicry attacks on host-based intrusion detection systems,” in *Proc. CCS’02 – 9th ACM Conference on Computer and Communications Security*, 2002, pp. 255-64.

[57] K. Tan, J. McHugh and K. Killourhy. “Hiding Intrusions: From the Abnormal to the Normal and Beyond,” in *Proc. IH 2002: Information Hiding*, 2002, pp. 1-17.

[58] K.M.C. Tan, K.S. Killourhy and R.A. Maxion. “Undermining an Anomaly-Based Intrusion Detection System Using Common Exploits,” in *Proc. RAID’02 - 5th international conference on Recent Advances in Intrusion Detection,* 2002, pp. 54-73.

[59] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov and W. Lee. “Polymorphic Blending Attacks,” in *Proc. USENIX-SS'06 - 15th conference on USENIX Security Symposium*, 2006.

[60] S.J. Stolfo and K. Wang. “Network Payload-based Anomaly Detection and Content-based Alert Correlation.” Doctoral Dissertation. New York: Columbia University, 2007.

[61] O.M. Kolesnikov and W. Lee. “Advanced Polymorphic Worms: Evading IDS by Blending in with Normal Traffic.” Internet: https://pdfs.semanticscholar.org/97d3/5f789529081a40131b1171a0bb6d6d069b9a.pdf?_ga=2.256374220.655035103.1568508389-933039622.1562710349, 2005.

[62] P. Fogla and W. Lee. “Evading network anomaly detection systems: formal reasoning and practical techniques,” in *Proc. CCS’06 – 13th ACM Conference on Computer and Communications Security*, 2006, pp. 59-68.

[63] C. Huang, T. Lee, L. Chang, J. Lin and G. Horng. “Adversarial Attacks on SDN-Based Deep Learning IDS System,” in *Proc. ICMWT* 2018*: Mobile and Wireless Technology 2018*, 2018, pp. 181-91.

[64] M. Rigaki and A. Elragal. “Adversarial Deep Learning Against Intrusion Detection Classifiers,” in *Proc. ST-152 Workshop on Intelligent Autonomous Agents for Cyber Defence and Resilience,* 2017.

[65] J. Clements, Y. Yang, A. Sharma, H. Hu and Y. Lao. (2019, Mar.). “Rallying Adversarial Techniques against Deep Learning for Network Security.” *arXiv Preprint - arXiv:1903.11688*. [On-line]. Available: https://arxiv.org/abs/1903.11688 [Jul. 24, 2019].

[66] Z. Wang. (2018, Jul.). “Deep Learning-Based Intrusion Detection With Adversaries.” *IEEE Access.* [On-line] 6, pp. 38367-84. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8408779&isnumber=8274985 [Jul. 24, 2019].

[67] N. Carlini and D. Wagner. “Towards Evaluating the Robustness of Neural Networks,” in *Proc. 2017 IEEE Symposium on Security and Privacy*, 2017, pp. 39-57.

[68] P. Chen, Y. Sharma, H. Zhang, J. Yi and C. Hsieh. “EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples,” in *Proc. AAAI-18 – 32nd AAAI Conference on Artificial Intelligence*, 2017, pp. 10-7.

[69] A. Kurakin, I.J. Goodfellow and S. Bengio. “Adversarial Machine Learning at Scale,” in *Proc. ICLR 2017 - 5th International Conference on Learning Representations*, 2017.

[70] P. Panda and K. Roy. (2019, May.). “Implicit Generative Modeling of Random Noise during Training improves Adversarial Robustness,” *arXiv Preprint - arXiv:1807.02188*. [On-line]. Available: https://arxiv.org/abs/1807.02188 [Jul. 24, 2019].

[71] S. Moosavi-Dezfooli, A. Fawzi and P. Frossard. “DeepFool: a simple and accurate method to fool deep neural networks,” in *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2574-82.

[72] W. Brendel, J. Rauber and M. Bethge. “Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models,” in *Proc. 6th International Conference on Learning Representations (ICLR 2018)*, 2018.

[73] J. Uesato, B. O'Donoghue, P. Kohli and A. van den Oord. (2018, Jun.). “Adversarial Risk and the Dangers of Evaluating Against Weak Attacks.” *arXiv Preprint - arXiv:1802.05666*. [On-line]. Available: https://arxiv.org/abs/1802.05666 [Jul. 24, 2019].

[74] P. Chen, H. Zhang, Y. Sharma, J. Yi and C. Hsieh. “ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models,” in *Proc. AISec'17 - 10th ACM Workshop on Artificial Intelligence and Security,* 2017, pp. 15-26.

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

[76] M. Rigaki and S. Garcia. “Bringing a GAN to a Knife-Fight: Adapting Malware Communication to Avoid Detection,” in *Proc. 2018 IEEE Security and Privacy Workshops (SPW),* 2018, pp. 70-5.

[77] S.P. Chung and A.K. Mok. “Allergy Attack Against Automatic Signature Generation,” in *Proc. RAID 2006: Recent Advances in Intrusion Detection*, 2006, pp. 61-80.

[78] B. Nelson and A.D. Joseph. “Bounding an Attack’s Complexity for a Simple Learning Model.” Internet: https://pdfs.semanticscholar.org/71d8/678edf41803a6c0827dc05f9906afb61e454.pdf?_ga=2.256875340.655035103.1568508389-933039622.1562710349, 2006 [Jul. 24, 2019].

[79] B. Nelson, M. Barreno, F.J. Chi, A.D. Joseph, B.I.P. Rubinstein, U. Saini, C. Sutton, J.D. Tygar and K. Xia. “Exploiting Machine Learning to Subvert Your Spam Filter,” in *Proc. LEET'08 - 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[80] M. Kloft and P. Laskov. “Online Anomaly Detection under Adversarial Impact,” in *Proc. 13th International Conference on Artificial Intelligence and Statistics (AISTATS): PMLR 9,* 2010, 405-12.

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

[82] H. Kim and B. Karp. “Autograph: Toward Automated, Distributed Worm Signature Detection,” in *Proc. 13th USENIX Security Symposium*, 2004, pp. 271-86.

[83] P. Akritidis, E.P. Markatos, M. Polychronakis and K. Anagnostakis. “STRIDE: Polymorphic Sled Detection through Instruction Sequence Analysis,” in *Proc. SEC 2005: Security and Privacy in the Age of Ubiquitous Computing,* 2005, pp. 375-91.

[84] T. Toth and C. Krüegel. “Accurate Buffer Overflow Detection via Abstract Payload Execution,” in *Proc. RAID'02 - 5th international conference on Recent advances in intrusion detection*, 2002, pp. 274-91.

[85] M. Roesch. “Snort: Lightweight Intrusion Detection for Networks,” in *Proc. LISA'99 - 13th USENIX conference on System administration,* 2019, pp.229-38.

[86] H. Zhang, X. Yu, P. Ren, C. Luo and G. Min. (2019, Jan.). “Deep Adversarial Learning in Intrusion Detection: A Data Augmentation Enhanced Framework.” *arXiv Preprint - ArXiv, abs/1901.07949*. [On-line]. Available: https://arxiv.org/abs/1901.07949 [Jul. 24, 2019].

[87] T. Na, J.H. Ko and S. Mukhopadhyay. “Cascade Adversarial Machine Learning Regularized with a Unified Embedding,” in *Proc. International Conference on Learning Representations (ICLR) 2018*, 2018.

[88] A. Madry, A. Makelov, L. Schmidt, D. Tsipras and A. Vladu. “Towards Deep Learning Models Resistant to Adversarial Attacks,” in *Proc. International Conference on Learning Representations (ICLR) 2018*, 2018.

[89] F. Farnia, J.M. Zhang and D. Tse. “Generalizable Adversarial Training via Spectral Normalization,” in *Proc. International Conference on Learning Representations (ICLR) 2019*, 2019.

[90] G.F. Cretu, A. Stavrou, M.E. Locasto, S.J. Stolfo and A.D. Keromytis. “Casting out Demons: Sanitizing Training Data for Anomaly Sensors,” in *Proc. SP '08 - 2008 IEEE Symposium on Security and Privacy*, 2008, pp. 81-95.

[91] S. Serneels and T. Verdonck. “Principal component analysis for data containing outliers and missing elements.” *Computational Statistics & Data Analysis*, vol. 52(3), pp. 1712-27*,* 2008.

[92] C. Croux, P. Filzmoser and M.R. Oliveira. “Algorithms for Projection-Pursuit Robust Principal Component Analysis.” *Chemometrics and Intelligent Laboratory Systems*, vol. 87(2), pp. 218-25, 2007.

[93] Z. Rustam and N. Olivera. “Comparison of fuzzy robust Kernel C-Means and support vector machines for intrusion detection systems using modified kernel nearest neighbor feature selection,” in *Proc. 3rd International Symposium on Current Progress in Mathematics and Sciences (ISCPMS 2017),* 2017.