UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Computer and network security

Adversarial attacks in computer and network security domain

Student:
Gabriel Taormina

Contents

1 Introduction

Artificial intelligence, especially as a result of the development of neural networks, deep learning and machine learning is increasingly taking hold in all areas of computing and everyday life, including computer and network security. In the latter, the use of this technology is still relatively in its infancy as the techniques, activities and processes used for security cannot be totally delegated to some model, indeed, computer security is very different from other areas of AI based on machine learning. The machine learning performance in other areas is often determined by a single element and metric, such as classification accuracy, security involves several factors that require attention. Attacks are becoming increasingly automated, making manual, active inspection very time consuming and sometimes ineffective. The choice of systems based on machine learning seems a good choice against this kind of problem, being able to autonomously analyze data and decisions to be made in a short time
However, machine learning systems are vulnerable to adversarial attacks, which limits their application especially in non-stationary and adversive environments such as the security domain.
The increasing use of machine learning has in turn been accompanied by a growing interest in adversarial machine learning, as know as adversarial learning, which corresponds to the attack and defense algorithms of machine learning models [33]. Of particular interest are adversarial samples, datasets modified for the purpose of being incorrectly modified by the attacked classifier. This type of attack is usually used in the fields of computer vision and natural language processing; however, in recent years, methods of generating adversarial examples have been increasingly used in other domains, including the field of computer security [51], domain characterized by many adversaries. The purpose is to go and define the strengths and weaknesses of using such technology in the security field, to try to determine whether it can actually be used and where it can improve in order to achieve a robust system.

2 Key factors in a security system

Before addressing the possible approaches, solutions and problems related to adversary attacks, it is good to define what elements a ML-based security system should consist of, as these will be the target of all the various adversarial attacks. It should be emphasized that all of these key elements are fundamental to any security system, whether or not ML techniques are used.
The effectiveness of a machine learning-based security system is defined by seven key factors or elements:

- **Efficiency**: learning and inference needs to be fast, especially compared to traditional methods to lead to a real advantage. The system should operate in real time, ensuring minimal latency in threat detection without overburdening the infrastructure.
- **Effectivity**: it needs to be effective either in detecting, analyzing or preventing threats. It includes the system's ability to identify both known and unknown threats, while maintaining a high level of reliability in the obtained predictions.
- **Accuracy**: it needs to correctly identify threats or anomalies, minimizing false positives and false negatives.
- **Controllability**: learning-based systems must retain control of the human operator, such that false decisions and errors can be immediately corrected and the system adapted to dynamics of the environments. The users can manage, adjust, and fine-tune the system's behavior. Unlike other fields that rely on learning, we need to move from totally autonomous to a supervised one by a human
- **Transparency**: the system must provide clear explanations for its decisions and actions. The user must be able to understand why a threat was reported and why the system reported it as such
- **Adaptability**: system's ability to understand and adapt itself to new threats or changes in behavior without a human constant intervention.The ML algorithms should be able to update their models in order to recognize new attacks or strange behaviors

- **Robustness**: a system must resist attacks aimed at manipulating the machine learning model or resist attacks that attempt to evade detection

Considering the adversarial attacks, the key points on which we need to focus to defend a system from them are **accuracy** and **robustness**

3 Adversarial Machine Learning

Adversarial machine learning is a field that focuses on studying and defending against attacks, called adversarial attack, that aim to deceive machine learning models by intentionally modifying data to influence their results, with the aim of manipulating the output of the final decision. Adversarial attacks have been studied for several years now in multiple fields, one of the first discoveries inherent to the domain of computer vision was that: a small perturbation in the form of a carefully processed and modified input could confuse a deep neural network and cause it to misclassify an image object [26]. Later other researchers demonstrated the use of adversarial attacks beyond image classification [17] [19] [18] [36].

A major component of an adversarial attack is the adversarial sample. It consists of an input to a machine learning model which has been perturbed. Considering a dataset with features x and label y, the corresponding adversarial sample is a specific data point x' which causes a classifier to predict a different label on x' other than y (x' is almost indistinguishable from x). The creation of adversarial samples involves solving an optimization problem to determine the minimum perturbation δ which maximizes the loss for the neural network such that

$$f(x+\delta) \neq f(x) \tag{1}$$

Figure 1: Adversarial Machine Learning process

In the field of computer vision, this type of attack is very difficult to identify by the human eye, misclassification can lead to serious consequences, just think of autonomous driving systems that rely on the recognition of elements in the surrounding environment. Here is an example

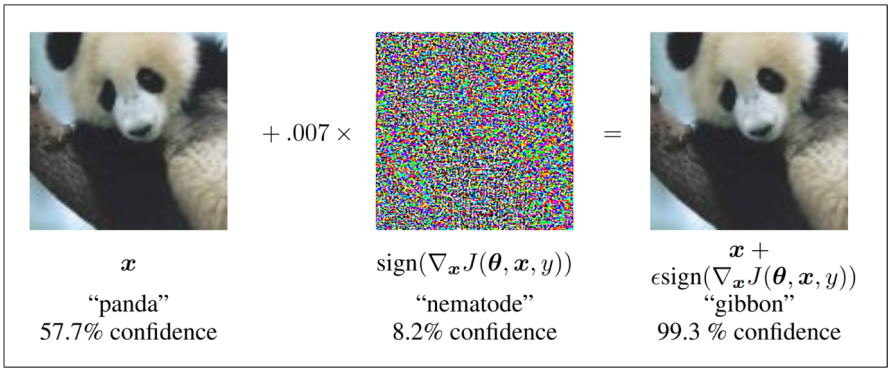

Figure 2: Example applied to images, the model considers the original image as a "panda" with 57.7%. Indeed, the image added a tiny perturbation is classified as a "gibbon" by the same model with 99.3%

3.1 Taxonomy of adversarial attack

We consider the following aspects to define adversarial attacks, based on the model described in [8][53], adapted to the security domain

	Types
Knowledge	White-Box, Black-box, Gray Box
Strategy	Causative, Oracle, Evasion
Space	Feature Space, Problem Space
Target	Label-indiscriminate, Label-targeted, Feature-targeted
Goal	Confidentiality, Integrity, Availability

Table 1: Taxonomy

3.1.1 Knowledge

The adversarial attacks are based on the amount of knowledge that an adversary has.
They can be divided into three types:

- **White-box attack**: the adversary has complete knowledge of the training data, the learning algorithm, the learned model and the parameters used during model training. The adversary has the exact information held by the developers of the machine learning system. In most cases this attack is not really possible
- **Black-box attack**: the adversary is totally unaware of the machine learning system has no knowledge of either the learning algorithm, the learned model, parameters. However, it is possible to train a local replacement model by querying the target model, exploiting the transferability of adversary samples, or using a model inversion method to obtain some information
- **Gray-box attack**: the adversary may have partial information or limited access to training data, one does not have the exact knowledge that the model creator possesses but the information is sufficient to attack the machine learning system and cause it to fail.

3.1.2 Strategy

The strategies correspond to the sequence of operation in which the adversary attack.
They can be divided in three types:

- **Causative**: it occurs during the training phase, corresponds to the corruption of the training data or model logic. They can be carried out throw to data injection where the adversary inserts input to alter the distribution, data manipulation where the input feature or labels are modified by the adversary, logic corruption where modify the model structure. This strategy is also known as "poisoning"
  The goal is to obtain an incorrect prediction from the model.
- **Oracle**: it occurs when the attacker has access to the interface related to model programming. We can subdivide this type of attack into three subtypes: extraction where from the original model output tries to recover the weights and parameters of the model, inversion where tries to recover the training dataset, inference where tries to identify some data points given the distribution of the dataset.
  The goal is to create a model to replace the original, the new model will be very similar to the original.
- **Evasion**: it occurs during testing or inference, it performs an optimization problem with the objective of making the loss function increase.
  The goal is to confuse the decision of the model

3.1.3 Space

The input space can be defined as the dimensional representation of all the possible configurations of the objects in the given context.
It can be divided in two types:

- **Feature space**: it is defined as the n-dimensional space in which all variables in the input dataset are represented. The attacker's goal is to remain benign without generating a new instance.
- **Problem space**: it refers to an input space in which the objects resides. The attacker's goal is to modify the actual instance to produce a new instance.

3.1.4 Target

Each attack has a different targeting, they can be divided in three types:

- **Label-indiscriminate attack**: it minimizes the probability of correctly classifying a perturbed sample.
- **Label-targeted attack**: it maximizes the probability of a specific class to be predicted for the adversarial example
- **Feature-targeted attack**: the harmful behavior of these attacks is triggered only by inputs marked with a specific attack signal

3.1.5 Goal

We have three possible goals that can be achieved by an attacker (often more than one is desired), they are known as the CIA triad:

- **Confidentiality**: the goal is to acquire private information from the system. The attacker try to intercept some communications between two parties to obtain those information. An example can be the stealing of the classifier's model [60]
- **Integrity**: the goal is to cause a wrong classification, perform incorrectly for some or all inputs. the output then will be different from that expected after training. An integrity attack can lead to either a targeted misclassification or a reliability attack. In the first case the model is manipulated to produce a specific wrong prediction, in the second case we obtain a misclassification into an arbitrary class other than the correct or reduce the confidence of the model An example can be misclassify a malware sample as benign [57]
- **Availability**: the goal is to cause a block of the system, in this case the model become unavailable. Another possibility is the get so excessive degradation that the system becomes unusable by the user. An example can be a denial-of-service attack, where excessive requests or inputs overwhelm the system, causing significant slowdowns or crashes.

Other target classification paradigms exist but are not optimal for network security, they are more inherent to the field of computer vision.
For completeness, a quick summary follows
We identify targets in terms of targeted or reliability attacks [2].

- In targeted attacks, the attacker has a specific target in terms of the pattern decision
- In reliability attacks, the attacker only tries to maximize the prediction error of the model without inducing a specific outcome.

3.2 Adversarial attack algorithm

As see in section 3.1.2 there are three types of strategy attack, that work at training time or inference time. In all cases one of the main purposes is to find an adversary example that is as close as possible to the original.
Let's see in detail some algorithms used in the various types of strategies

3.2.1 Causative strategy algorithms

These methods are used to alter the data or the model logic.
We can consider three main categories:

- **Data injection**: It is based on the input of data that alter the data distribution.
  Some relevant example are:
  - Gradient-ascent based attack: it is based on Support Vector Machine (SVM), it uses model gradients to alter inputs and maximize classifier error [11].
  - Backdoor attack (BadNets): the attacker introduces a backdoor in the model that allows to activate a specific behavior when the model meets a specific trigger. The attack is based on a full or partial outsourced training process. This type of attack is particularly insidious when using pre-trained models, since the end user is not aware of the manipulation [30].
- **Data manipulation**: it is based on the input feature or labels that are modified by the adversary.
  Some example are
  - Label flipping attack: the attacker manipulates the training data labels, assigning wrong labels to specific examples. Note that the corruption of the labels confuses the model even if the characteristics of the examples remain unchanged. This leads to learning incorrect associations between inputs and their labels [35].
  - Clean-label poisoning attack: the attacker enters into the training dataset examples that are apparently "clean", with correct labels, but modified to have representations similar to those of a specific example. The model will thus confuse the target example with the poisoned ones, leading to an incorrect classification. The poisoned examples seem to be properly labelled, making it more difficult for security checks to identify as harmful [54].
- **Logic corruption**: it is based on modification of the model structure and logic.
  Some example are:
  - Weight poisoning: the internal weights of the model are altered during training, causing the model to malfunction in specific situations or certain inputs. The attack is hidden in the weights of the model, making it difficult to detect [38].
  - Model tampering: the attacker directly modifies the architecture or model parameters, thus corrupting its logic. This may include the insertion of abnormal behaviour only in certain circumstances, making it difficult to detect [7].

3.2.2 Oracle strategy algorithms

This methods are used when the attacker has access to the interface related to model programming.
We consider three main categories:

- **Extraction Attacks**: type of attack in which an adversary attempts to recreate a machine learning model, often black-box, without having direct access to its parameters or internal structure. The process is based on querying the target model with input data with the aim of training a replacement model. Some example are:
  - Knockoff Nets: the attack occurs in two stages. In the first, the attacker queries the target model with a data set, collecting the results. In the second it uses the obtained input-output pairs to train a replacement model called a knock-off. This technique also involves the use of reinforcement learning [45].
  - Jacobian-based Dataset Augmentation: the attack creates a replacement model using the predictions of a black-box model as labels. Relies on a small set of initial inputs and on the model's Jacobian to find examples that maximize the output of the original model. By continuously querying the model and adding new elements to the dataset is possible to get a model similar to the original one [47].

- **Inversion Attacks**: type of attack in which an adversary attempts to recover the training dataset on which a target model was trained. This type of attack raises privacy issues as it extracts sensitive data from the model.
  Some example are:
  - MI-Face attack: the attack allows to reconstruct the appearance of an individual used in the training dataset based on the model classification probabilities (taking into account a facial recognition model) [24].
  - Gradient inversion attack: the attack uses gradients calculated during training to reconstruct original inputs. This type of attack is common in contexts where model updates are distributed [49].
- **Inference Attacks**: type of attack in which an adversary tries to identify some data points given the distribution of the dataset. The goal is to obtain information about specific input data or reconstruct sensitive properties of that data.
  Some example are:
  - Membership Inference Attack: the attack is intended to determine whether specific data was used to train the model. The attacker uses the model's responses to make inferences about whether this data is present in the training-set [55].
  - Property Inference Attack: type of attack that identifies global properties of training data as distributions, aggregate characteristics or particular subgroups [25].

This type of attacks has become increasingly common due to the increase in machine learning cloud services by private companies such as Amazon, Google, etc.

3.2.3 Evasion strategy algorithms

This methods are used to fool the machine learning system during testing or inference. Evasion attacks rely on generating adversarial samples that can evade a classifier with small modifications. Evasion attacks are the most prevalent type of attacks in the adversarial scenario [23].
We can consider two main categories:

- **Gradient-based**: type of attack typical of white-box knowledge. This type exploits the model's gradients with respect to inputs to generate perturbations that maximize classification error, the goal is to modify the input to force the model to misclassify.
  The attacks focus on the optimization processes that these models use to learn and make predictions. They are called "gradient-based" because they exploit gradients, which represent the rate of change of the model's output with respect to its parameters, calculated during the training phase. The gradients act as a guide for the direction in which the parameters should be adjusted to minimize error. By manipulating the gradient, attackers can then alter the behavior of the model [34].

  Some example are:
  - Fast Gradient Sign Method FGSM: the attack is designed to perturb input data in a way that leads machine learning models to make incorrect predictions. The idea is to utilize the gradient of the loss with respect to the input data to create adversarial samples (created by finding the maximal direction of positive change in the loss). It works by calculating the gradient of the loss function with respect to the input data and then adjusting the input data in the direction of the sign of this gradient. The disturbance is controlled by a parameter ϵ, the disturbed input is then given to the model, which results in an incorrect classification. This method is fast since only a one-step gradient update is performed along the direction of the sign gradient at each level [26].

The attack can be described as:

$$x' = x + \epsilon \cdot \text{sign}\left(\nabla_x J(\theta, x, y)\right)$$

where:

* x' adversarial sample,
* x original input,
* ϵ parameter,
* $J(\theta, x, y)$ loss function
* ∇_x gradient with respect to x,
* sign() sign function, small perturbation in gradient direction.

There are evolutions or extensions which are: Fast Feature Gradient Sign Method FFGSM, Iterative Fast Gradient Sign Method I-FGSM, Momentum Iterative Fast Gradient Sign Method MI-FGSM.
As simple as it is able to hit state-of-the-art models.

– Carlini and Wagner Attack: the attack is designed to create antagonistic examples that can actually deceive deep neural networks, ensuring that disturbances are minimal and undetectable. It is a target attack specifically for existing adversarial defense methods, is one of the most advanced adversary attacks against deep neural networks. The attack is based on the minimization of the cost function:

$$\min_{\delta} ||\delta||_p + c \cdot f(x + \delta)$$

Where:

* δ perturbation added to the input x,
* $||\delta||_p$ norm,
* c trade-off parameter,
* $f(x + \delta)$ loss function.

There are multiple variants based on different types of norms, very common is the L2-based [16].

- **Gradient-free**: type of attack typical of black-box knowledge, is a class of adversarial attacks that does not rely on model gradients to generate perturbations. It operate by evaluating the objective function at different points in the feature space, iteratively searching for the optimal solution using various strategies [53] [65].
Some example are:

  – Zeroth Order Optimization ZOO: the attack attempts to emulate gradient-based methods, but without having access to gradients in any way, the idea is to approximate gradients through the "Finite Difference Method" [12] . This method consists of making continuous queries to the model with slightly perturbed inputs and observing how the responses vary.
  The difference between the responses given by the model on similar inputs is used to estimate the gradient [20].

  – Boundary Attack: the attack is iterative and does not require access to model gradients. The idea is to start with an input that the model already misclassifies and then progressively reduce the magnitude of the perturbation while maintaining the misclassification. The input is modified through small steps so as to act along the model's decision boundary [15].

Of interest is the concept of transferability related to this type of attack (and black-box attacks in general), the concept of transferability of adversary examples between different types of machine learning models has been demonstrated: adversary examples generated for a given model can also be effective in fooling other models such as logistic regression, decision trees, support vector machines, nearest neighbors [46].
Black box attacks are thus feasible even on non-deep learning models.

This type of attacks (and its expansions) are widely used, for this reason it has been described more than other attacks treated before.

Note: Both types can be used for white and black box, but some attacks are more effective and therefore more used with certain knowledge

Follows a table containing all the attacks mentioned and also others not covered but used

Causative Attacks	**Oracle Attacks**	**Evasion Attacks**
Gradient-ascent based	Knockoff Nets	FGSM
BadNets	Jacobian Augmentation	Carlini and Wagner
Label Flipping	MI-Face attack	Zeroth Order Optimization (ZOO)
Clean Label Poisoning	Gradient Inversion Attack	Boundary Attack
Weight Poisoning	Membership Inference Attack	Projected Gradient Descent (PGD)
Model Tempering	Property Inference Attack	L-BFGS
SVM Poisoning	Copycat CNN	Feature Adversaries
Feature Collision	Attribute Inference	Basic Iterative Method (BIM)
	KnockOff Nets	Jacobian Saliency Map (JSM)
	Decision Tree Attack	DeepFool
		NewtonFool
		Elastic-Net Attack (EAD)
		AutoAttack
		Shadow Attack
		Adversarial Patch
		Square Attack
		Threshold Attack
		Hop Skip Jump Attack
		SimBA
		DPatch
		Wasserstein Attack
		Spatial Transformation
		Elastic Net
		IFS
		FFGSM
		I-FGSM
		MI-FGSM

Table 2: Adversarial Attack Algorithms

4 Adversarial attack in security domain

Most of the algorithms presented in the previous sections can be applied to different domains, however, most of the studies have focused in the field of computer vision. The cybersecurity domain seems to be even more relevant as adversaries really exist and have specific goals and objectives. The main reason for focusing on other domains is the fact that an attack in cybersecurity is more difficult than executing such an attack in the computer vision domain (or others)

4.1 Examples of the difference between Computer Vision (CV) and Computer Security (CS)

- Any adversarial executable file must preserve its malicious functionality after the sample modification
  - CV: the attacker can change an image (individual pixel color, intensity,...) always getting a valid image in the attack.
  - CS: the attacker can change the value of a byte in an executable but leading to the execution of a different and detectable (as well as unwanted by the attacker himself) function ex: readFile() to writeFile().

  In this example, the security domain induces a higher level of difficulty.

- Executables are more complex than images
  - CV: an image has fixed dimensions and is represented by a matrix. If the actual image has different dimensions, it is possible to resize, crop the input image to fit the size limits.
  - CS: an executable has a variable dimension. The execution path of an executable can be input-dependent, so the adversary attack must take into account and support every possible input that the malware may encounter when it is executed

  In this example, the security domain induces a higher level of difficulty.

With these two simple comparisons alone, it is evident how the domain of security is positioned at a higher level of difficulty than others, such as computer vision.

4.2 Adversarial attack classification

This subsection will show numerous examples, categorized according to the goal/target of the attack type, of adversary attacks in the field of security.
For each attack there will be a table containing information on the attack (goal, target,...) based on the taxonomy presented in section 3.1 plus some other consideration.
In detail:

- **attacker's goals**: objective of the attack
- **attack's targeting**: target on which the attack goes to act
- **attacker's knowledge**: knowledge possessed to carry out the attack

Name	
Attacker's goals	
Attack's targeting	
Attacker's knowledge	

Table 3: Attack overview

The following subsections will be divided into 4 main categories:

- URL and phishing detection
- Network intrusion detection
- Spam detection
- Malware detection

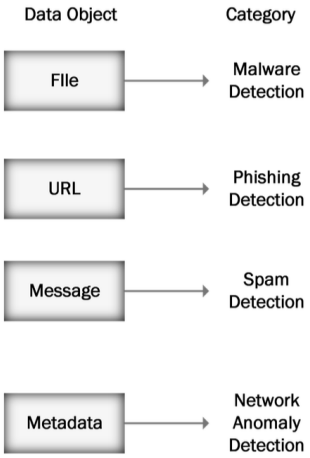

Figure 3: Categories

4.2.1 URL and phishing detection

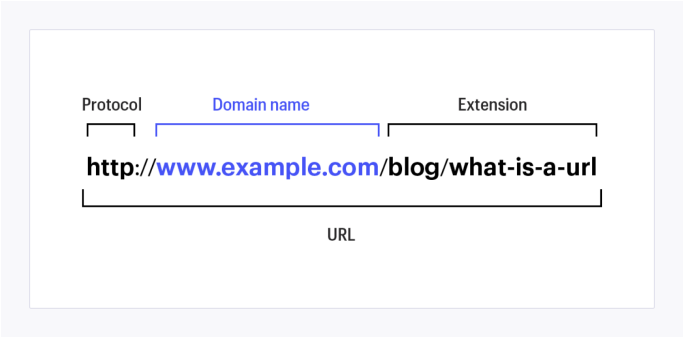

Figure 4: URL

- A URL begins with the protocol used to access the page, the fully qualified domain identifies the server hosting the page.
  It consists of a registered domain name RDN and prefix referred to as sub-domains, it may also have a path and query components.
- Phishing refers to the class of attacks in which a victim is lured to a fake web page masquerading as an authentic target website with the purpose of obtaining personal data or user credentials

A phisher can have full control over the subdomains by setting them to any value, and also can change the queries (or path) at will.

Examples now follow:

- **Bypassing detection URL with GAN**: evasion technique that attacks URL phishing detection systems throw Generative adversarial networks GAN. The generated samples fooled black-box phishing detectors: the experiment revealed that many classifiers failed to identify any of the opposing examples and at the same time false-positive rates increased.
  This attack involves the use of two patterns: the Generator that generates adversary URLs that look legitimate and the Discriminator that tries to distinguish between legitimate phishing URLs and URLs created by the generator. The purpose is to gradually improve the ability of the generator to create URLs that are increasingly difficult to distinguish [4].

Bypassing Detection of URL using GAN	
Attacker's goals	Evading phishing detection systems by generating adversarial URLs
Attack's targeting	Phishing detection models based on ML
Attacker's knowledge	Black-box

Table 4: Attack overview

- **Feature Importance Guided Attack**: it is able to falsify phishing detection models by altering the input data so that the relevant model classifies them as safe or legitimate. This is done by manipulating the features most relevant to the model, those related to classification decisions, when the phishing detection model assigns high weight to a particular URL structure that is usually associated with phishing, the attack will aim to modify the structure to make the valid URL appear. The attack is model-agnostic: it does not depend on the specific architecture or internal parameters of the model. Is able on average to bring F1- score of the model around 40% [27].

Feature Importance Guided Attack	
Attacker's goals	Classify phishing URLs as legitimate manipulating features
Attack's targeting	Phishing detection models based on ML
Attacker's knowledge	Gray-box

Table 5: Attack overview

- **Inference integrity attack**: the attack attempts to circumvent a phishing URL classifier based on a Long Short-Term Memory (LSTM) model that works at the character level to detect malicious URLs. LSTM is a type of recurrent neural network (RNN) suitable for temporal sequences and texts. The attack starts from the concatenation of benign and known URLs, from the concatenation of URLs, a set of sentences of predefined length is then created and used by the LSTM model. After the model has completed generating the text, it is segmented using HTTP structure delimiters to produce a list of pseudo-URLs. Each pseudo-URL is then associated with a compromised domain [10].

Inference integrity attack on LSTM	
Attacker's goals	Elusion of URL phishing detectors based on LSTM
Attack's targeting	Phishing URL classifiers
Attacker's knowledge	Gray-box

Table 6: Attack overview

- **Inference integrity attack "updated"**: A GAN is used to produce domain names that are structured to look legitimate or difficult to classify as malicious. The domain names generated by the GAN, that is trained throw an encoding system based on an LSTM, are used to create synthetic data. That data are then used to train new detection models. The idea is that the generated adversarial domain names are difficult to identify from existing detection models. This attack/method is an evolution and combination of the first and third elements of the list in this subsection [6].

Inference integrity GAN & LSTM	
Attacker's goals	Creating adversarial domains that elude DGA classifiers
Attack's targeting	DGA classifier
Attacker's knowledge	Black-box

Table 7: Attack overview

- **Generating Optimal Attack Paths**: method that creates adversarial phishing attacks by finding optimal subsets of features that lead to a higher evasion rate. To achieve this, techniques such as Recursive Feature Elimination, Lasso and Cancel Out are also used. To select the optimized paths, adversarial samples are generated with different feature combinations. The feature selection methods are categorized based on their relationship with learning models into three categories (using RFE, Lasso, infoGain), then the feature selection techniques are combined together using the union operator and/or intersection operator. To select the optimal number of features instead cross-validation is used.
  This type of attack has been found to have more evasion capabilities than GANs, which do not make a selection of features, randomly disturbing [3].

Optimal Attack Paths	
Attacker's goals	Elude detection through an optimized selection of features
Attack's targeting	Phishing detection systems
Attacker's knowledge	Black-box, Gray-box

Table 8: Attack overview

4.2.2 Network intrusion and anomaly detection

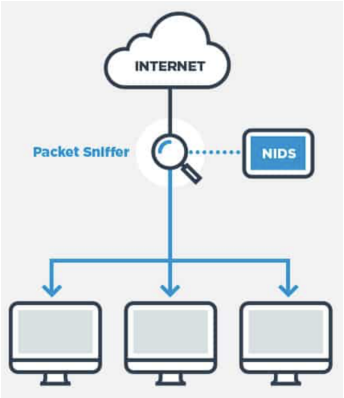

Figure 5: NIDS

Network anomaly detection systems learn the network activity and detect irregularities. They continuously scan the network, analyzing encrypted data and spot anomalies in real-time. The use of machine learning techniques over time applied to this field has become more and more frequent due to its "natural" predisposition (learning, analysis,...)
A security system commonly used to secure networks is the NIDS (network intrusion detection system), which is a device or software that monitors all traffic passing throw a strategic point for malicious activities, when an activity is detected, an alert is generated.

Examples now follow:

- **IDSGAN**: framework of the generative adversarial network, is based on the Wasserstein GAN [8]. It generates the adversarial malicious traffic records, attacking intrusion detection systems by deceiving and evading the detection. IDSGAN is composed a Generator that transform original malicious traffic records into adversarial malicious ones and a Discriminator classifies traffic examples and dynamically learns the real-time black-box detection

system. The restricted modification mechanism is designed to preserve original attack functionalities of adversarial traffic records. IDSGAN can lower the detection rates of some NIDS models to zero percent [40].

IDSGAN	
Attacker's goals	Evade intrusion detection systems by generating adversarial malicious traffic
Attack's targeting	Network-based intrusion detection systems
Attacker's knowledge	Black-box

Table 9: Attack overview

- **Kitsune inference integrity attack**: the attack is directed against a specific model: Kitsune. it is an ensemble of autoencoders used for online network intrusion detection that uses packet statistics which are fed into a feature mapper, it divides the features between the autoencoders to ensure fast prediction. The RMSE output of each autoencoder is then used as input in another autoencoder from which the final RMSE score used for anomaly detection will be obtained.
  In the attack are used four adversarial strategy: FGSM, JSMA, C&W and ENM.
  The attacker as distance metric uses the L_p distance on the feature space between the original input and the perturbed input. The minimization of the L_0 norm is related to the modification of a small number of extracted characteristics [22].

Kitsune inference integrity attack	
Attacker's goals	Evading the Kitsune detection system by altering input data
Attack's targeting	Kitsune
Attacker's knowledge	White-box

Table 10: Attack overview

- **Neptune evasion attack** the attack is designed to fool Neptune, NIDS that uses several classifiers to detect network attacks (such as a SYN Flood Distributed Denial of Service) . The attacker goes to alter only certain characteristics of the network traffic, thus making it more difficult for the system to identify the traffic as malicious, thus reducing its accuracy. The attack uses a series of evasion attacks, which go to alter input data so that malicious traffic is not detected by NIDS.
  To do this, a small number of traffic characteristics are modified in order to maintain the attack but hide its nature from the system's detection techniques.
  The accuracy of Neptune has been successfully brought down from 100% to about 0%. It was also tried on some classifiers based on Random Forest, logistic regression, Support Vector Machines always getting the same results (close to 0%), on the contrary k-NN got a decrease but staying on 50% [1].

Neptune evasion attack	
Attacker's goals	Evading the Neptune detection system by altering input data
Attack's targeting	Neptune
Attacker's knowledge	White-box

Table 11: Attack overview

- **Transmission Control Protocol Obfuscation Techniques**: it is possible by manipulating the properties of TCP connections to make the detection of intrusions in systems more complex. The attack hides a malicious TCP communication by modifying the characteristics of the network traffic so that it does not result as dangerous traffic. Multiple strategies are used in the attack to trick the system: data fragmentation where data packets are fragmented into several parts of different sizes making it more difficult to trace back to the original version, Payload obfuscation which makes it difficult for systems to analyse the contents of packets and change the sequence of TCP packets.
  The experiment done by researcher and the attack confirm that it is possible to circumvent the intrusion detection capability of all trained classifiers without prior knowledge of obfuscated attacks, causing an exacerbation of true positive rate (TPR) ranging from 7.8% to 66.8% [32].

TCP Obfuscation Techniques	
Attacker's goals	Camouflage of malicious network traffic to escape the detection
Attack's targeting	IDS monitoring TCP network traffic
Attacker's knowledge	Gray-box

Table 12: Attack overview

- **NIDS Evasion Attack**: this attack is based on some studies covered by the author of IDSGAN. The attack was carried out on a DNN model trained to classify malicious/harmful behavior in a network using NSL-KDD database, this model is comparable from the performance point of view to more sophisticated NIDS classifiers. It is assumed that there is no information on the model being attacked. Three separate attacks were executed in total: a zero-order optimization ZOO attack, one based on GAN, a C&W attack but on a replacement model (properly trained).
  The last attack (C&W) works because the disturbances created against a replacement model can be successfully transferred to the original model, where a mount designed for one model can trick other models if they have similar characteristics (transferability property).
  The attack, in this way, creates small disruptions on network packet input data to trick the system into labeling malicious traffic as legitimate while maintaining the malicious functionality of undetected traffic [63].

NIDS Evasion Attack	
Attacker's goals	Elusion of a NIDS based on a DNN model
Attack's targeting	NIDS based on DNN model
Attacker's knowledge	Black-box

Table 13: Attack overview

4.2.3 SPAM detection and filtering

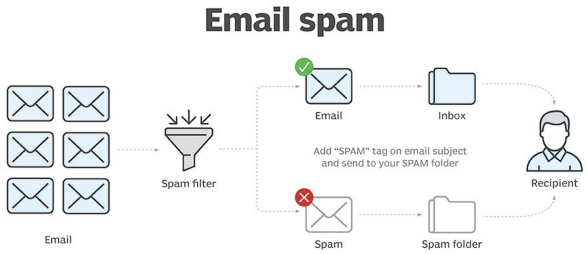

Figure 6: Caption

A spam filter is used to determine whether an incoming message is legitimate or dangerous, preventing (or at least warning) the user from running into malicious content that is harmful to their

device or personal data.
Spam can also contain harmless content but that clutter the space inside the box, the identification of these leads to not occupy resources and avoid "hide" important emails.
Spam detectors were among the first applications in the security field to use machine learning, this at the same time meant that they were among the first applications subject to attack.
Most of the models were trained using supervised training methods.

Examples now follow:

- **Subverting of Spam Filter**: the attack infer with the training process by modifying the filter itself, not only allows SPAM emails to be placed inside the box without making it appear as such but is able to remove legitimate emails from the inbox. The study of the attack focused on multiple spam filters, specifically SpamBayes.
  The attack can be divided in two parts:
  1)Dictionary attack: indiscriminate attack, it is based on sending attack emails that contain many words that are used/present in legitimate emails. In the training process of SpamBayes, due to the use of these emails, the words in them will have a higher spam score. By doing so future legitimate emails will have a higher chance of being classified as spam, making malicious ones legitimate instead.
  2)Focused attack: a non-indiscriminate attack that involves knowledge of a specific legitimate email or type of email that the attacker wants blocked. Attack emails containing words that are likely to be found in the target email are sent to the victim. When SpamBayes trains on these attack emails, the spam scores of the targeted tokens increase, thus increasing the likelihood that the target message will be filtered as spam.
  with only 1% control over training data, the attack managed to get 36% of legitimate emails classified as spam [44].

Subverting of Spam Filter	
Attacker's goals	Elusion of spam filtering systems
Attack's targeting	SpamBayes
Attacker's knowledge	Gray-box

Table 14: Attack overview

- **Generalized Greedy Attack on NLP Classifiers**: the attack has been gradually improved over time by multiple researchers. It is a generalized attack against classifiers based on NLP natural language processing, such classifiers are nowadays also used as spam filters. A greedy approach (greedy attack) is used that goes to find semantically similar words and then replace them in the text, going through all possible word substitutions and then selecting those that cause the least impact on the classifier.
  The goal is to maintain a high score in the language model while minimizing the change in original meaning. The classifiers that have been escaped are those based on Naive Bayes, Long Short-Term Memory (LSTM), word-level 1D CNN.
  The proposed improvement over time using a combined technique of paraphrasing at the sentence and word level, going not only to change individual words but also the syntactic structure of the sentence while maintaining semantic and syntactic coherence. What makes these attacks noteworthy is at their applicability to various NLP domains, not limited to spam classification [37] [39].

Generalized Greedy Attack on NLP Classifiers	
Attacker's goals	Evading NLP classifiers by word substitution
Attack's targeting	NLP classifiers:NB, LSTM, word-level 1D CNN
Attacker's knowledge	Black-box

Table 15: Attack overview

- **Image Spam Classifiers**: an image spam is spam email that has been embedded in an image, it was developed in an effort to evade text-based filters.
  The attack exploits universal adversarial perturbations by relying on transformations to create tailored "natural perturbations" in images.
  Universal perturbations are perturbations designed to confuse the image classifier over a wide range of inputs, thereby decreasing the accuracy of the model.
  Natural perturbations are obtained thanks to the DeepDream algorithm, which goes to enhance the "natural" or legitimate features maximizing them, with the aim of confusing the classifiers of image-spam.
  In addition, Gradient-weighted Class Activation Mapping (Grad-CAM) is used to identify areas of an image that have the greatest impact on the classifier. Once the key areas for classification as spam have been identified, the inverse mapping obtained by Grad-CAM is applied to the natural perturbations.
  After creating the natural perturbations and integrating them into the spam image, the universal perturbations is applied.
  The final image is modified to such an extent that it is not classified as spam but at the same time readable and understandable by the user (to ensure the opening of the malicious file).
  The model's accuracy can be reduced by up to 23.7% with this attack [48] [42].

Image Spam Classifiers	
Attacker's goals	Avoid detection by image-spam detector
Attack's targeting	Deep learning-based image spam classifiers
Attacker's knowledge	Gray-box

Table 16: Attack overview

- **Marginal attack on Bayesian filter**: marginal attacks cause classification errors by inserting sensitive words with attributes contrary to the original message into the original text.
  Sensitive words entered must meet both requirements: a sensitive word must occur frequently in legitimate messages, a sensitive words must have distinctive features to serve as a basis for the division of valid and spam.
  Consider the fact that messages are often composed of short texts and the words to be inserted will be as few as possible.
  Marginal attacks are launched in two ways:
  1) Changing the structure of keywords in the text. This method may circumvent classifier detection but undermines the effect expression of the original message if tampering is beyond the limit and obtains an opposite effect.
  2) Inserting irrelevant words into a spam. To improve the effectiveness of the attack, the words inserted are usually sensitive words of attribute against the original message.
  Naive Bayes algorithms are used in spam-filtering tasks, it is a family of algorithms based on applying Bayes theorem with strong independence assumptions among the features.
  When a naive Bayes algorithm is applied to spam-filtering tasks, a minor tweak in the eigenvalue will influence the result.
  Whit this attack the accuracy could be reduced from 93% to 56%. This type of attack is highly transferable and excellent results (reaching an accuracy of 1.5%) have been obtained on models such as Logistic regression , decision trees, SVM [67].

Marginal attack on Bayesian filter	
Attacker's goals	Keyword manipulation to bypass Bayesian filters
Attack's targeting	Bayesian filter
Attacker's knowledge	White-box

Table 17: Attack overview

- **Adversarial email crafting**: the attack is based on two methods to evade filters:
  1) the method modifies the content of the email so that the TF-IDF in the modified email are similar to those of a legitimate email thus confusing the detector.
  The TF-IDF (term frequency–inverse document frequency [62]) is a function used in information retrieval to measure the importance of a term in relation to a document, more emphasis is placed on terms appearing in the document but which are generally infrequent.
  2) the method identifies a group of significant words used in legitimate emails and then adds them to the text of spam emails. By doing so these words will confuse the models making the email classify as legitimate.
  The attack has been tested on various models such as KNN, SVM, decision trees and logistic regression in both black-box and white-box scenarios, with variable results but always highlighting the fact that the second method was found to be the most effective and impactful, leading to a significant reduction in accuracy by the model.
  This attack calls up the first presented (Subverting of Spam Filter) [61]

Adversarial email crafting	
Attacker's goals	Defeat spam filters by altering the content of emails
Attack's targeting	ML models for spam detection
Attacker's knowledge	Black-box, White-box

Table 18: Attack overview

4.2.4 Malware detection

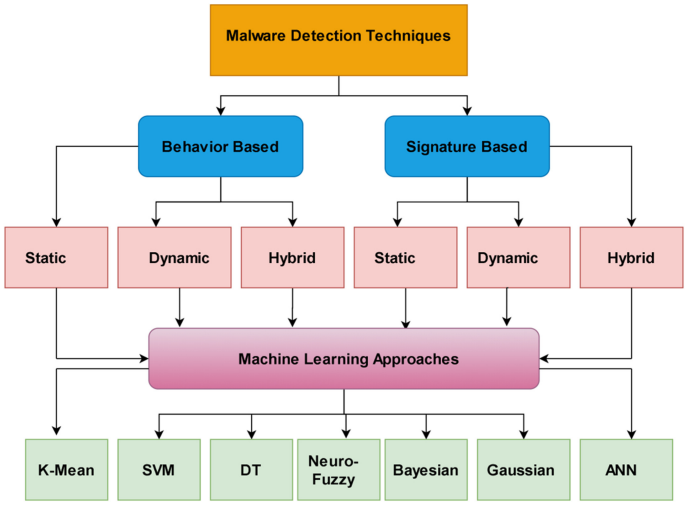

Figure 7: Malware detection techniques: "classic" and ML-based

Malware detection is a crucial security issue. Over the years, both offensive and defensive parts have grown exponentially, just think from 1984 to today more than 1,416,204,248 malware [9] has been identified and this number is increasing every day. For this reason, traditional malware detection systems that rely mainly on digital signatures have become less effective. The use of machine learning has become more and more popular to protect against malware attacks because they allow to detect malware not seen and not signed (instead of the eurisitic methods and signatures), thus better able to deal with the large and continuous development of malware. The problem is that these defense methods are as seen vulnerable to adversarial attack, thus introducing an additional danger beyond the malware itself. An example of products based on machine learning and deep models are Microsoft ATP, CrowdStrike, SentinelOne, Cylance.
Malware detection can be based on the analysis of:

- Static features: the malware is detected without executing the code.
- Dynamic features: the malware is detected executing the code, typically a suspicious malware

sample is executed in a sandbox in an attempt to discover dynamic behavioural patterns (ex:API call sequences).

Examples now follow:

- **EvadeDroid**: the attack exploits a method based on n-grams, sequences of n elements representing the behavior of an application (in this context they can be APIs or instructions), to select candidate donors. They are benign examples from which is possible to extract elements that will be injected into malware samples.
  The goal is to select benign samples similar so as to add them to the malware sample, without altering or disrupting the original malicious behavior of the malware; thus mimicking benign behavior.
  The attack is based on an iterative and incremental process to achieve the progressive alteration of the malware. In order for the malware to be identified as a benign element, its characteristics are changed in small steps. With each iteration, gadgets (code snippets or API calls from licit origin) are added into the malware samples to confuse detection by the system, the addition of the gadgets makes the malware look more like a harmless application being the similar characteristics in terms of n-grams.
  The attack proved to be very effective against several malware detection models, for example against DREBIN, a well-known Android malware detection system achieved an of evasion rate of 81% [13].

EvadeDroid	
Attacker's goals	Evasion of Android malware detectors
Attack's targeting	Android malware detectors
Attacker's knowledge	Black-box

Table 19: Attack overview

- **Discrete Feature Gradient Attack**: the attack targets a malware classifier based on a fully connected neural network used to analyze Android applications, leading to incorrect (benign) classification by the system.
  The attack is a variant of the Fast Gradient Sign Method (FGSM) that considers a discrete domain. In the security domain, in fact, features are often discrete, represented by 0-1.
  In the attack one does not go to remove features but to add new ones to change their classification, thus doing so the "malicious" behavior remains unchanged and the classifier deceived. The attack develops in an iterative process until the malware is identified as benign, this process is divided into two steps:
  1) Gradient calculation, the gradient of the pattern is calculated with respect to the vector of (binary) features of the malware to obtain the information about which features most influence the classification of the pattern. The vector goes to represent the presence or absence of features used by the model
  2) Modification of the vector, we search within the vector for the element that if modified (from 0 to 1 or vice versa) brings the greatest increase in the benign score, thus maximizing the positive impact on the classification result.
  The attack has been tested on several DNN models for Android malware detection with good results [28] [29].

Discrete Feature Gradient Attack	
Attacker's goals	Evasion of Android malware detectors by adding features
Attack's targeting	Android malware detectors
Attacker's knowledge	White-box

Table 20: Attack overview

- **API Call Injection Attack**: the attack is based on adding new API calls to the original malware traces without removing the existing ones with the aim of fooling the classifier by passing the malware as benign, making the API call trace look like a benign one.
  The API call traces correspond to the sequence of APIs used by the malware during execution. Unlike other attacks presented this was initially used on a replacement model trained to approximate the behavior of the original model, the purpose of this step is to use the replacement model to optimize API call changes.
  The replacement model is based on a variant of RNN, the Gated Recurrent Unit GRU. Attacks that are successful against GRU are then transferred to the original model by exploiting the already encountered property of transferability of adversary attacks. It is also possible to extend the attack to hybrid classifier using both static and dynamic features by acting on each feature separately, thus making the attack more versatile.
  The attack has been tested on different models such as LSTM , RNN, DNN, CNN 1D, SVM.
  The authors of this attack have also developed a comprehensive and automated framework that can create a new malware executable from an existing sample [51].

API Call Injection Attack	
Attacker's goals	Fooling malware classifiers by adding API calls to execution traces
Attack's targeting	Malware detector based on ML & NN
Attacker's knowledge	Gray-box

Table 21: Attack overview

- **GAPGAN**: the attack is based on the use of Generative Adversarial Networks GAN for generating adversarial examples against binaries-based malware detection.
  A GAN is a type of neural network already used in other previously described attacks, consisting of a generator and a discriminator. The system has two phases: training, in which the generator and discriminator are trained, and attack in which only the generator trained in the previous step is used.
  In the first step, the generator creates adversary perturbations that are added to the malware binaries while simultaneously the discriminator tries to identify them (when the discriminator can no longer identify them, the generator is ready).
  The perturbations in this case are small changes applied to the binaries that are intended to change the classification of the malware but without altering its malicious functionality. In fact, the attack operates at the byte level, directly modifying the binaries without going through the source.
  GAPGAN's success rate reached 100% attacks with the addition of payloads equal to 2.5% of the total length of the original data, such a small modification makes the attack very efficient [64].

GAPGAN	
Attacker's goals	Bypassing malware classifiers by modifying binaries
Attack's targeting	Malware DNN detection models
Attacker's knowledge	Black-box

Table 22: Attack overview

- **Slack attacks**: the attack focuses on binaries such as GAPGAN where the input malware binaries are altered so that they are classified as benign while retaining their malicious function, however the approach used is very different. The difficulty in this type of attack lies precisely in the alteration of the bytes involved, because altering the bytes of the malware arbitrarily could affect the malicious effect of the malware by thwarting the attack.
  To avoid this problem, two methods are used that add adversarial noise to the end of the malware binaries:
  1) Gradient append, the added bytes are modified according to the pattern error gradient (and thus not randomly), with the purpose of increasing the probability of misclassification.
  2) Random append, bytes are added to the end of the file randomly from a uniform distribution.

Both methods rely on the fact that adding bytes to the end of files does not alter the functionality of the program.
These two methods while effective may encounter convergence problems, variants can be used to overcome this problem:
1) Bening append, bytes added are benign
2) FGM append, similar to gradient but exploits the fast gradient method.
The peculiarity of this attack lies in its adaptability: when bytes cannot be added to the end of the file (due perhaps to too large a size) the attack performs a Slack attack by exploiting slack bytes. This area can be modified freely without altering the behavior of the malware. The most common approach is Slack FGM (merging of the methods presented).
Note: in the Slack version bytes are modified while in the append version they are added.
The attack managed to evade MalCon, a convolutional neural network designed to classify binary files [36] [58].

Slack attacks	
Attacker's goals	Evasion of detection models by the addition or modification of bytes
Attack's targeting	Malware classification systems based on binary files, ex: MalCon
Attacker's knowledge	Gray-box

Table 23: Attack overview

5 Defense methods against adversarial attacks

In an ideal scenario every attack should be matched by a defense, unfortunately this is not always the case. Strategies for defense against adversary attacks do exist, but as already mentioned with attacks, their development has taken place mostly in the field of computer vision. Many methods that are effective in that domain turn out to be invalid for the cyber security domain.
Let us now follow the same approach covered in the attacks section: first a general analysis of defense methods using a generic machine learning approach and then an analysis to methods in the security domain.
Note: the paper is structured primarily on the explanation of adversary attacks, consequently this section will have a smaller scope, the purpose is to provide a general smattering on possible methods of defense against adversary attacks.

The Adversarial defense methods can be divided into two categories:

- **Detection methods**: methods used to detect adversarial samples.
- **Robustness methods**: methods used to make a classifier more robust to adversarial samples but not intended to detect them.

Each defense method can be attack-specific or attack-agnostic. Attack agnostic defense methods are more generic and therefore preferable from the standpoint of versatility but at the same time more difficult to create.
Adversarial attack methods are mostly optimization algorithms, which search for a perturbation at the lower bound that corresponds to an adversarial sample.
The underlying optimization algorithms often produce high-frequency results, so the same thing follows for defense methods, leading to "specialization" against high frequencies. However, this leads to a breach that can in turn be used in an attack, making vulnerable to adversarial samples that are generated within a low frequency subspace. Considering high and low frequencies makes defense more complicated than offense [14] [31].

5.1 Common defensive methods

There are many approaches used for defense. One of the most common and used is based on the gradient, as seen in attacks (e.g. FGSM) the use of techniques based on the gradient are very powerful and used.
Let us now look at two methods that aim to block these kinds of attacks:

- **Gradient masking**: method that modifies the machine learning model in an attempt to obscure its gradient to an attacker. This can be achieved by saturating the sigmoid network, leading to a gradient effect in gradient-based attacks. Doing so forces the neural network to work in a nonlinear saturation system.
  Furthermore, it is possible to make the model insensitive to perturbations generated by the FGSM method and iterative adversarial attacks by using Jacobian regularization for each layer of the network, including the output layer.
  Jacobian regularization corresponds to the addition of a penalty term during training with the purpose of reducing the sensitivity of the model to changes in input [43].

- **Gradient regularization**: method that penalizes large variations in the output of some layer of the neural network, minimizing the loss function, increasing the robustness of the model and preventing overfitting (or underfitting).
  DNN training with gradient adjustment improves robustness to adversarial perturbations as much as adversary training. Combining gradient regularization and adversary training together can achieve greater robustness.
  One disadvantage, however, is the time increase of the training phase [52].

In the gradient adjustment method, another method used for defense was mentioned: **adversarial training**. The idea is quite intuitive (especially since we are always talking about systems based on machine learning), which is to train the model through adversarial examples as well.

- **Adversarial training**: method that aims to increase the robustness of a machine learning model against adversarial samples by generating adversarial examples that maximize the loss function of the model and minimizing the loss function not only on the original inputs, but also on the generated adversarial inputs; this process is based on the min-max formulation.
  The process consists of three steps:
  1) train the class on the original dataset
  2) generate adversarial samples
  3) iterate additional training epochs using the adversary samples
  Adversary training is based on the min-max formulation.
  Adversary training however can reduce the performance of deep learning models on clean input also there is always the possibility that the adversary may implement a different attack method than the one used for training the network [59] [41].

Other approaches (not described) are given in the following table:

Other methods
Defensive Distillation
Detecting Adversarial Samples
Feature Reduction
Input Randomization
Ensemble Defenses

Table 24: Defense methods

5.2 Examples of defensive methods in security domain

Now follow some examples of defense applied to the security domain, specifically following the four categories seen in attacks section in that domain (URL and phishing detection, network intrusion detection, spam detection, malware detection).

For URL and phishing detection:

- Increased defenses of DGA systems: to improve the robustness of DGA detection models against adversarial attacks, two defense techniques can be used (or combined): adversarial training and distillation so as to make the models more robust to adversarial perturbations. The first method was analyzed in the previous section, in brief we describe Distillation: agnostic method, which involves training the model in two stages, in the first stage a model is trained which is in turn used in the second stage to train a second model, with the aim of reducing the "temperature" of the softmax to better distribute the probabilities over the classes. These methods have proven their effectiveness by demonstrating improved robustness against adversary attacks [56].

For Network and intrusion detection:

- Increased defense of network and intrusion detection system: to improve the robustness of network intrusion detection models against adversarial attacks a framework has been proposed that functions is similar to the defense adversarial training method.
  The framework integrates deep adversarial learning with statistical learning, using data augmentation techniques. Doing so improves intrusion detection through synthetic data used in training that simulates real attacks.
  The statistical model used is the Poisson-Gamma model, a probabilistic model that combines two statistical distributions (Poisson and Gamma) to generate data [66].

For SPAM detection:

- Increased defense of SPAM detection systems: to improve the robustness of SPAM detection models against adversarial attacks one of the techniques used is adversarial training, which as explained in the previous section consists of retraining the model by including adversarial samples. This approach is particularly effective when the detection model used is an RNN [5].

For Malware detection:

- Increased defense of malware detection: in this case the detection system is an Android malware classifier. The proposed method is agnostic and is divided into two stages:
  1) offline training phase: the classifier is trained on data that does not change dynamically (offline), selecting relevant features to then build an initial version of the model
  2) online training phase: the model is used in real time for malware detection.
  The data used contains adversarial samples [21].

6 Conclusion

From the many examples, it is clear that the use of adversarial attacks is a real and growing problem, especially at this time when we are increasingly approaching artificial intelligence.
It should be noted, however, that in the attacks presented, misclassification (and consequently failure to identify) of the defensive-detection system was always mentioned and never other types of actions, thus limiting the types of attacks.
In general, although possible, these attacks nowadays are still too "challenging" in the field of cybersecurity, this is due to the unique characteristics of this particular domain. In addition, most of the attacks have been conducted mostly on data at rest with few successful attempts at attacks on data in transit or streaming data, a situation less common in a "modern" attack.
Thinking about the future, it is undeniable that these attacks will have to be taken into account, especially trying to develop new methods to defend data and model logic. One idea that researchers are working on is the use of encryption on the data used for training (such a system is already used to preserve some sensitive data)[50], in fact, it has been seen that encrypted data is not altered by

adversary attacks.
What they want to determine is whether performing encryption before applying learning on the data is a reliable defense against adversary learning in network security.

References

[1] James Aiken and Sandra Scott-Hayward. "Investigating adversarial attacks against network intrusion detection systems in sdns". In: *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE. 2019, pp. 1–7.

[2] N. Akhtar and A. Mian. "Threat of adversarial attacks on deep learning in computer vision: A survey". In: *arXiv preprint arXiv:1801.00553* (2018).

[3] R. Al-Qurashi et al. "Generating optimal attack paths in generative adversarial phishing". In: *2021 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2021, pp. 1–6.

[4] A. AlEroud and G. Karabatis. "Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks". In: *Proceedings of the Sixth International Workshop on Security and Privacy Analytics*. 2020, pp. 53–60.

[5] Moustafa Alzantot et al. "Generating natural language adversarial examples". In: *Proceedings of EMNLP*. 2018, pp. 2890–2896.

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

[7] Arikkl. *Model Tampering*. `https://www.cyberin.ai/post/model-tampering`.

[8] M. Arjovsky, S. Chintala, and L. Bottou. "Wasserstein GAN". In: *arXiv preprint arXiv:1701.07875* (2017).

[9] AV-ATLAS. *AV-ATLAS Portal*. `https://portal.av-atlas.org/malware`. 2024.

[10] Alejandro Correa Bahnsen et al. *DeepPhish: Simulating Malicious AI*. `https://albahnsen.com/2018/06/03/deepphish-simulating-malicious-ai/`. Retrieved March 29, 2021. 2018.

[11] B. Biggio, B. Nelson, and P. Laskov. "Poisoning attacks against support vector machines". In: *arXiv preprint arXiv:1206.6389* (2012).

[12] George Boole. *Calculus of finite differences*. BoD–Books on Demand, 2022.

[13] H. Bostani and V. Moonsamy. "EvadeDroid: A practical evasion attack on machine learning for black-box Android malware detection". In: *arXiv preprint arXiv:2110.03301* (2021).

[14] W. Brendel, J. Rauber, and M. Bethge. "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models". In: *arXiv preprint arXiv:1712.04248* (2017).

[15] Wieland Brendel, Jonas Rauber, and Matthias Bethge. "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models". In: *arXiv preprint arXiv:1712.04248* (2017).

[16] N. Carlini and D. Wagner. "Towards evaluating the robustness of neural networks". In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017.

[17] L. Chen and Y. Ye. "SecMD: Make machine learning more secure against adversarial malware attacks". In: *Australasian Joint Conference on Artificial Intelligence*. Springer, 2017, pp. 76–89.

[18] L. Chen, Y. Ye, and T. Bourlai. "Adversarial machine learning in malware detection: Arms race between evasion attack and defense". In: *European Intelligence and Security Informatics Conference (EISIC)*. IEEE, 2017, pp. 99–106.

[19] L. Chen et al. "DroidEye: Fortifying security of learning-based classifier against adversarial Android malware attacks". In: *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2018, pp. 782–789.

[20] Pin-Yu Chen et al. "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models". In: *Proceedings of the 10th ACM workshop on artificial intelligence and security*. 2017, pp. 15–26.

[21] Sen Chen et al. “Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach”. In: *Computers & Security* 73 (2018), pp. 326–344.

[22] Joseph Clements et al. “Rallying adversarial techniques against deep learning for network security”. In: *arXiv preprint arXiv:1903.11688* (2019).

[23] Marco Farinetti. “Evasion attacks against machine-learning based behavioral authentication”. PhD thesis. Politecnico di Torino, 2018.

[24] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. “Model inversion attacks that exploit confidence information and basic countermeasures”. In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 2015, pp. 1322–1333.

[25] Karan Ganju et al. “Property inference attacks on fully connected neural networks using permutation invariant representations”. In: *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 2018, pp. 619–633.

[26] I. J. Goodfellow, J. Shlens, and C. Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv:1412.6572* (2014).

[27] G. Gressel et al. “Feature importance guided attack: A model agnostic adversarial attack”. In: *arXiv preprint arXiv:2106.14815* (2021).

[28] K. Grosse et al. “Adversarial perturbations against deep neural networks for malware classification”. In: *arXiv preprint arXiv:1606.04435* (2016).

[29] Kathrin Grosse et al. “Adversarial examples for malware detection”. In: *Proceedings of ESORICS*. 2017, pp. 62–79.

[30] T. Gu, B. Dolan-Gavitt, and S. Garg. “Badnets: Identifying vulnerabilities in the machine learning model supply chain”. In: *arXiv preprint arXiv:1708.06733* (2017).

[31] C. Guo, J. S. Frank, and K. Q. Weinberger. “Low frequency adversarial perturbation”. In: *arXiv preprint arXiv:1809.08758* (2018).

[32] I. Homoliak et al. “Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: An adversarial approach”. In: *arXiv preprint arXiv:1805.02684* (2018).

[33] Ling Huang et al. “Adversarial Machine Learning”. In: *Proceedings of the ACM Workshop on Artificial Intelligence and Security (AISec)*. 2011, pp. 43–58.

[34] Marin Ivezic. *Gradient-Based Attacks: A Dive into Optimization Exploits*. `https://securing.ai/ai-security/gradient-based-attacks/`. 2023.

[35] Marin Ivezic and Luka Ivezic. *Label Flipping in AI: A Threat to AI Systems*. `https://securing.ai/ai-security/label-flipping-ai/`. 2022.

[36] B. Kolosnjaji et al. “Adversarial malware binaries: Evading deep learning for malware detection in executables”. In: *arXiv preprint arXiv:1803.04173* (2018).

[37] Volodymyr Kuleshov et al. *Adversarial examples for natural language classification problems*. Unpublished Manuscript. 2018.

[38] Keita Kurita, Paul Michel, and Graham Neubig. *Weight Poisoning Attacks on Pre-trained Models*. 2020. arXiv: `2004.06660 [cs.LG]`. URL: `https://arxiv.org/abs/2004.06660`.

[39] Qi Lei et al. “Discrete attacks and submodular optimization with applications to text classification”. In: *arXiv preprint arXiv:1812.00151* (2018).

[40] Z. Lin, Y. Shi, and Z. Xue. “IDSGAN: Generative adversarial networks for attack generation against intrusion detection”. In: *arXiv preprint arXiv:1809.02077* (2018).

[41] A. Madry et al. “Towards deep learning models resistant to adversarial attacks”. In: *arXiv preprint arXiv:1706.06083* (2017).

[42] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. *Inceptionism: Going deeper into neural networks*. `https://research.googleblog.com/2015/06/inceptionism-going-deeperinto-neural.html`. 2015.

[43] A. Nayebi and S. Ganguli. “Biologically inspired protection of deep networks from adversarial attacks”. In: *arXiv preprint arXiv:1703.09202* (2017).

[44] B. Nelson et al. “Exploiting machine learning to subvert your spam filter”. In: *Proceedings of LEET*. Vol. 8. 1. 2008, p. 9.

[45] T. Orekondy, B. Schiele, and M. Fritz. “Knockoff nets: Stealing functionality of black-box models”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4954–4963.

[46] N. Papernot, P. McDaniel, and I. Goodfellow. “Transferability in machine learning: from phenomena to black-box attacks using adversarial samples”. In: *arXiv preprint arXiv:1605.07277* (2016).

[47] Nicolas Papernot et al. “Practical black-box attacks against machine learning”. In: *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 2017, pp. 506–519.

[48] A. Phung and M. Stamp. “Universal adversarial perturbations and image spam classifiers”. In: *Malware Analysis Using Artificial Intelligence and Deep Learning*. Springer, 2021, pp. 633–651.

[49] Jin Qian et al. *GI-SMN: Gradient Inversion Attack against Federated Learning without Prior Knowledge*. 2024. arXiv: `2405.03516 [cs.LG]`. URL: `https://arxiv.org/abs/2405.03516`.

[50] Christian Rechberger and Roman Walch. “Privacy-Preserving Machine Learning Using Cryptography”. In: *Security and Artificial Intelligence: A Crossdisciplinary Approach*. Springer International Publishing, 2022. URL: `https://doi.org/10.1007/978-3-030-98795-4_6`.

[51] Ishai Rosenberg et al. “Generic black-box end-to-end attack against state of the art API call based malware classifiers”. In: *Proceedings of RAID*. 2018, pp. 490–510.

[52] A. Ross and F. Doshi-Velez. “Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 2018.

[53] Saiyam Sakhuja. *Gradient-Based vs. Gradient-Free Optimization*. `https://sakhujasaiyam.medium.com/gradient-based-vs-gradient-free-optimization-82fa7e90b3e9`. 2024.

[54] A. Shafahi et al. “Poison frogs! Targeted clean-label poisoning attacks on neural networks”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 6103–6113.

[55] Reza Shokri et al. “Membership inference attacks against machine learning models”. In: *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, pp. 3–18.

[56] Lior Sidi, Asaf Nadler, and Asaf Shabtai. “MaskDGA: A black-box evasion technique against DGA classifiers and adversarial defenses”. In: *arXiv preprint arXiv:1902.08909* (2019).

[57] Nedim Srndic and Pavel Laskov. “Practical evasion of a learning-based classifier: A case study”. In: *Proceedings of the IEEE Symposium on Security and Privacy (SP)*. 2014, pp. 197–211.

[58] J. W. Stokes et al. “Attack and defense of dynamic analysis-based, adversarial neural malware detection models”. In: *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 2018, pp. 1–8.

[59] C. Szegedy et al. “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199* (2013).

[60] Florian Tramèr et al. “Stealing machine learning models via prediction APIs”. In: *Proceedings of USENIX Security Symposium*. 2016, pp. 601–618.

[61] C. Wang et al. “Crafting adversarial email content against machine learning based spam email detection”. In: *Proceedings of the 2021 International Symposium on Advanced Security on Software and Systems*. 2021, pp. 23–28.

[62] Wikipedia contributors. *TF-IDF*. `https://it.wikipedia.org/wiki/Tf-idf`. 2024.

[63] K. Yang et al. “Adversarial examples against the deep learning based network intrusion detection systems”. In: *Proceedings of MILCOM*. 2018, pp. 559–564.

[64] J. Yuan et al. “Black-box adversarial attacks against deep learning based malware binaries detection with GAN”. In: *ECAI 2020*. IOS Press, 2020, pp. 2536–2542.

[65] Matthew Yuan, Matthew Wicker, and Luca Laurenti. “Gradient-free adversarial attacks for bayesian neural networks”. In: *arXiv preprint arXiv:2012.12640* (2020).

[66] H. Zhang et al. “Deep adversarial learning in intrusion detection: A data augmentation enhanced framework”. In: *arXiv preprint arXiv:1901.07949* (2019).

[67] G. Zhaoquan et al. “Marginal attacks of generating adversarial examples for spam filtering”. In: *Chinese Journal of Electronics* 30.4 (2021), pp. 595–602.