AdverSPAM: Adversarial SPam Account Manipulation in Online Social Networks

Article

Accepted version

F. Concone, S. Gaglio, A. Giammanco, G. Lo Re, M. Morana

ACM Transactions on Privacy and Security
DOI: 10.1145/3643563

Publisher: ACM

AdverSPAM: Adversarial SPam Account Manipulation in Online Social Networks

FEDERICO CONCONE, University of Palermo, Italy
SALVATORE GAGLIO, University of Palermo, Italy
ANDREA GIAMMANCO, University of Palermo, Italy
GIUSEPPE LO RE, University of Palermo, Italy
MARCO MORANA*, University of Palermo, Italy

In recent years, the widespread adoption of Machine Learning (ML) at the core of complex IT systems has driven researchers to investigate the security and reliability of ML techniques. A very specific kind of threats concerns the *adversary* mechanisms through which an attacker could induce a classification algorithm to provide the desired output. Such strategies, known as Adversarial Machine Learning (AML), have a twofold purpose: to calculate a perturbation to be applied to the classifier's input such that the outcome is subverted, while maintaining the underlying intent of the original data. Although any manipulation that accomplishes these goals is theoretically acceptable, in real scenarios perturbations must correspond to a set of permissible manipulations of the input, which is rarely considered in the literature. In this paper, we present *AdverSPAM,* an AML technique designed to fool the spam account detection system of an Online Social Network (OSN). The proposed black-box evasion attack is formulated as an optimization problem that computes the adversarial sample while maintaining two important properties of the feature space, namely *statistical correlation* and *semantic dependency.* Although being demonstrated in an OSN security scenario, such an approach might be applied in other context where the aim is to perturb data described by mutually related features. Experiments conducted on a public dataset show the effectiveness of *AdverSPAM* compared to five state-of-the-art competitors, even in the presence of adversarial defense mechanisms.

Additional Key Words and Phrases: adversarial machine learning, spammer detection, online social networks, evasion attacks

ACM Reference Format:
Federico Concone, Salvatore Gaglio, Andrea Giammanco, Giuseppe Lo Re, and Marco Morana. . AdverSPAM: Adversarial SPam Account Manipulation in Online Social Networks. 1, 1 (July), 31 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The need to provide people with more and more sophisticated smart services has recently fostered a renewed interest in the topics of Artificial Intelligence (AI) and Machine Learning (ML). At the same time, it has now been widely accepted that the fallibility of intelligent systems can lead to

*Corresponding author.

Authors' addresses: Federico Concone, University of Palermo, Viale delle Scienze, ed. 6 - 90128, Palermo, Italy, federico.concone@unipa.it; Salvatore Gaglio, University of Palermo, Viale delle Scienze, ed. 6 - 90128, Palermo, Italy, salvatore.gaglio@unipa.it; Andrea Giammanco, University of Palermo, Viale delle Scienze, ed. 6 - 90128, Palermo, Italy, andrea.giammanco@unipa.it; Giuseppe Lo Re, University of Palermo, Viale delle Scienze, ed. 6 - 90128, Palermo, Italy, giuseppe.lore@unipa.it; Marco Morana, University of Palermo, Viale delle Scienze, ed. 6 - 90128, Palermo, Italy, marco.morana@unipa.it.

XXXX-XXXX//7-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

diffuse, and potentially serious, errors. This is especially true in certain critical scenarios, where the misbehavior of AI and ML can threaten the security of a cyber system. For instance, intelligent algorithms are commonly adopted to analyze large amounts of data and recognize anomalous behaviors, such as network intrusions, cyber-attacks, slander campaigns, or spam activities. In all these cases, a failure will quickly propagate from the cyber space to the real world. The issue is even more severe when the malfunctioning of ML algorithms is induced by attackers exploiting automated Adversarial Machine Learning (AML) strategies, whose applications are as broad as ML itself.

AML [8] techniques aim to exploit the same optimization mechanism at the core of ML with an opposite intent: to let the model be sure, with high confidence, about an erroneous prediction. *Adversarial samples* are defined as "those that change the verdicts of Machine Learning systems but not those of humans" [10]. To be more specific, adversarial attacks can be characterized on the basis of different traits, the most general of which is whether they aim to directly alter the input data, or the corresponding feature values. In the simplest scenario, we can assume that the classifier operates directly on the input data; thus, the adversary calculates a perturbation to be added to the input in order to obtain an adversarial sample. In the case of image analysis, for instance, this would result in modifying individual pixel values directly. Most works consider AML in this scenario since it is straightforward for a human to verify the appearance of a certain image and assess the correctness of the classifier.

When ML algorithms operate on complex feature sets that indirectly represent the input, and have no meaning to a human, it is burdensome to find out how to manipulate the feature values while also preserving the nature of the input. In these cases, two alternatives exist. In the easiest one, the features are independent; thus the single feature value can be modified without impacting the others. Otherwise, if the features are dependent, i.e., if they capture aspects that influence each other, only a subset of all possible manipulations ensures that such an interconnection is not broken, and of these, the most effective must be identified.

In this paper, we consider this last setting in the context of a ML system aimed to detect and block spam accounts in Online Social Networks (OSNs).

The high *availability* of OSNs, that is, the fact that they can be easily accessed at any time from anywhere, is the key factor in their success and, at the same time, the main reason for the interest of malicious entities. Since spammers may adopt different strategies to achieve their goal, ML algorithms are usually trained on a wide set of features capable of capturing various aspects, such as information concerning the properties of the account, the history of shared content, and the degree to which a user is connected to the rest of the network. While such a comprehensive set of characteristics allows to identify different types of threats, large feature sets may also extend the attack surface of ML systems, making them more easily deceived.

From an AML point of view, features describing the user of an OSN are closely interrelated (e.g., adding or deleting a message containing a URL would impact multiple feature values at the same time) and the steps required to fool a classifier cannot be made on a trial-and-error basis. In this context, accomplishing an attack means finding a way for the adversary to automatically alter the feature vector describing a *spammer* so that it is recognized as *genuine*, without impairing the malicious behavior.

In order to achieve this goal, we propose *AdverSPAM*[1], an evasion adversarial attack against OSN spam account detection systems that allows to find the optimal perturbation to deceive the target model, while preserving the inter-relationships among the features describing the user behavior. The main contributions of this work are summarized as follows.

[1]https://github.com/agiammanco94/AdverSPAM

- We propose a novel AML strategy that explicitly preserves the *statistical correlation* among the features of the input space. This is achieved by formulating the attack as an optimization problem in which the search for the adversarial sample is constrained by the maintenance of the correlation coefficients observed in the original data.
- To the best of our knowledge, this is also the first work in which the adversarial perturbation is chosen while preserving the *semantic dependency* that occurs when multiple features are computed from the same data. Without this constraint, the algorithm would produce perturbations that are numerically admissible, but not obtainable through real (legitimate) account manipulations.
- *AdverSPAM* allows to deceive unknown classifiers by forging an adversarial sample that has minimum distance from the original sample (so as to preserve the input nature), while showing characteristics that are referable to the desired class samples (so as to maximize the probability of deceiving different target models).
- In order to make the results easily reproducible, the experimental analysis was carried out using a public dataset of *spammer* and *genuine* Twitter accounts. Experiments involved four distinct surrogate models, twelve possible target models, and the performance of *AdverSPAM* were compared with five different state-of-the-art attacks. Such a robust evaluation is essential to guarantee the generalisation capacity of the AML technique. Moreover, we present a concrete case aimed at exploring the manipulations to be applied to a real OSN account in order to carry out the attack suggested by *AdverSPAM*.
- The evaluation of *AdverSPAM* also includes the analysis of its robustness to three adversarial defense mechanisms, namely, two state-of-the-art approaches and a confidence-based technique we designed to counter the specificities of our attack.

The remainder of the paper is organized as follows. Section 2 provides the essential background to understand the core aspects of an AML strategy. Relevant related works are discussed in Section 3, where a categorization based on the adversary attributes and the application domain explored is also given. The proposed methodology is thoroughly presented in Section 4 by introducing the attack scenario, the threat model, and the algorithms that constitute *AdverSPAM*. Experimental analysis is illustrated in Section 5, which includes a description of the experimental setup, a preliminary assessment of the attack followed by a comparative analysis with five baselines, a concrete case study with a real example of the modifications made by AdverSPAM, and the evaluation of three mitigation strategies. Discussion on advantages and limitations of the approach is provided in Section 6, while conclusions follow in Section 7.

2 BACKGROUND ON ADVERSARIAL MACHINE LEARNING

Supervised learning algorithms consist of a mapping $\mathcal{L} : \mathcal{L}(x) = \hat{y}$, whose aim is to assign the independent variables x with the dependent variables $\hat{y}$ according to the observed ground truth labels y. In order to learn the best mapping from inputs to predictions, the classifier has to be *trained* on a set of measurements; generally, the aim of such a procedure is to find a model $\mathcal{L}$ which is an acceptable approximation of the training data labels. This is pursued by minimising a *loss function*, $\ell(\cdot)$, between the real y and the predicted $\hat{y}$ label:

$$\min_{\theta} \ell(\theta, x, y) + \lambda(\theta) \,, \tag{1}$$

where θ is the set of $\mathcal{L}$'s adjustable parameters, and λ is a *regularization* term which is typically employed to counter *overfitting*, i.e., the inability of the learnt model to generalize on new instances of the data.

Adversarial Machine Learning stems from the observation that the same optimization procedure can be used by an *adversary* for leading a learning algorithm to a wrong prediction. The strategy is to forge an *adversarial sample* $\tilde{x} = x+\delta$, where the value of the *perturbation* δ is found by subverting Eq. 1:

$$\max_{\tilde{x}} \ \ell(\theta, \tilde{x}, y) \ . \tag{2}$$

Indeed, the goal of the adversary is to find the perturbation that maximises the error between the ground truth and the predicted labels. Since the creation of $\tilde{x}$ starts from following the positive direction of the loss function, depending on the magnitude of δ it may happen that $\tilde{x}$ succeeds in escaping $\mathcal{L}$'s decision boundary, so altering the final predicted label $\hat{y}$. Thus, the value of δ should be large enough to allow the decision boundary to be crossed. However, it also have to be small enough not to completely alter the sense of the input x. Both of these requirements are addressed in the method we propose.

In the most favorable scenario, the attacker possesses perfect knowledge of all the four elements that define the model, namely the training data, $\mathcal{D}$, the corresponding feature representation, $\mathcal{F}$, the target learning model, $\mathcal{T}$, and its parameters, θ. When the 4-uple ($\mathcal{D}$, $\mathcal{F}$, $\mathcal{T}$, θ) is known, the attack is called *white-box*. Otherwise, if knowledge of some components of the domain is missing, the conducted attacks are known as *gray-box*.

A situation in which the adversary does not know any parameters is called *black-box* attack. However, even in this case, the attacker must know something about the aim of the classifier and the features that are commonly used in the domain under analysis. A particular class of black box attacks, named *model based*, exploits the *transferability* property [51] of adversarial samples: a surrogate model mimicking the unknown target algorithm is leveraged to synthesize adversarial samples that will transfer to the target black box [41]. Reasons for such a property include orthogonality of the models' gradient directions, the alignment of their decision boundaries as well as geometric correlations between different regions of such boundaries [48], and magnitude of input gradients [29].

Let $\mathcal{T}$ be the target learning algorithm that the adversary wants to evade, and $\mathcal{S}$ the local surrogate model available to the adversary, the corrupted version of sample x, i.e. $\tilde{x}$, built by adding perturbation δ computed from $\mathcal{S}$, is said to transfer towards model $\mathcal{T}$ if:

$$\begin{cases} \mathcal{S}(x) \neq y^* \neq \mathcal{T}(x) \\ \mathcal{S}(\tilde{x}) = y^* = \mathcal{T}(x) \end{cases} , \tag{3}$$

where y^* is the adversary's desired output label.

In order to measure the *distance* between the original and the corrupted input samples, several metrics based on the Hölder norm L_p are commonly adopted [1]. The L_p norm between two samples x and $\tilde{x}$, is defined as:

$$L_p(\tilde{x}, x) = ||x - \tilde{x}||_p = \sqrt[p]{\sum_{i=1}^{d} (|x_i - \tilde{x}_i|^p)} \ , \tag{4}$$

where $p \in \mathbb{Z}$, x and $\tilde{x}$ are two vectors of d components with $d \in \mathbb{N}$, and subscript i denotes the i-th component [11]. Starting from defining $0^0 = 0$, the L_0 norm is computed as:

$$L_0(\tilde{x}, x) = \sum_{i=1}^{d} (|x_i - \tilde{x}_i|^0), \tag{5}$$

which provides a measure of how many single components are different in the two samples compared. The L_2 norm computes the euclidean distance between samples in order to measure

their offset in the feature space:

$$L_2(\tilde{x}, x) = \sqrt{\sum_{i=1}^{d} (|x_i - \tilde{x}_i|^2)} \ . \tag{6}$$

Another type of widely used L_p norm for adversarial attacks is the L_∞ norm, which computes the maximum drift between the original and perturbed components of the input and is defined as:

$$L_\infty(\tilde{x}, x) = \max_i \ |x_i, \tilde{x}_i|. \tag{7}$$

Using different metrics for determining the imperceptibility of an adversarial perturbation has a strong effect on the attack's outcome. By minimising L_2, for instance, it is possible to forge adversarial samples where multiple features have been slightly perturbed, thus performing a *dense* attack. On the other hand, focusing on L_0 permits to modify only a limited set of features, which results in a *sparse* attack [59].

3 RELATED WORK

Over the past two decades, ML has become one of the core pillars of information technology and has acquired a central role in our daily lives. In this sense, the scientific community has invested a considerable effort in defining learning-based pattern classifiers that, so far, show impressive performance in several application domains. More recently, it has been shown that adversarial perturbations, carefully created in both training and testing phases, can easily subvert predictions made by ML algorithms. The vulnerability of ML to forged *adversarial* patterns, along with the design of appropriate countermeasures, is addressed in a quite novel research area, known as Adversarial Machine Learning.

The effectiveness of AML is frequently demonstrated on computer vision scenarios [71], in which it is easier to visually assess the validity of an adversarial sample. In fact, the *imperceptibility* of the perturbed input is a desired property of any AML attack [33]. Among the most popular approaches in AML, the *Fast Gradient Sign Method* (FGSM) [33] paved the way for many studies exploring evasion strategies against image classifiers. In [51], the FGSM is used for creating adversarial images on the MNIST dataset, and testing their effectiveness against classifiers hosted on Google and Amazon. A thorough analysis of this property, known as *transferability*, is provided in [29], where the root causes of this phenomenon are connected with the magnitude of the loss' gradient, implying that strongly regularized learning algorithms are more robust to attacks. *DeepFool* (DF) attack [47] introduced the idea of reasoning on the particular structure of the decision boundary to evade, testing approaches against well known convolutional neural network architectures. *Carlini and Wagner* (C&W) approach [11] focuses on the formulation of the adversarial evasion as an optimization problem, where the cost function tends to maximize the confidence of the attacked classifier about the misclassification; such approach, originally tested against image classifiers, has become a milestone for testing the efficiency of novel evasion strategies. The attack proposed in [15] is one of the first effective methods for evading unknown classifiers, where the adversary has the capability to query such models for reasoning on the provided label. AML may also regard the perturbation of audio signals in speech recognition systems. The goal of [57] is to let such a system transcribe a prefixed target sentence by perturbing only those frequencies that are un-listenable by humans. A stochastic compression technique is proposed in [5] for creating more robust models for speech recognition in smart home devices. Attacks are simulated through *Fast Gradient Sign Method* (FGSM) and *Projected Gradient Descent* (PGD), aiming at the execution of unwanted commands by the intelligent assistants deployed in the house. When the target model is unknown and its only observable output is composed of classes probabilities, authors of [67] propose a technique based

on the computation of the gradient in a limited set of selected coordinates, with the momentum iterative method for creating the audio adversarial sample.

Other application scenarios include cybersecurity domains in which ML plays a dominant role in threat detection. Malware detection systems, for instance, can be attacked by adversaries through the manipulation of different sections of the source code [27]. In [54], the authors formalize a novel problem-space attack with the aim of automatically generating realistic and inconspicuous evasive adversarial applications for Android devices. Malware can also spread through the infection of PDF files; a comprehensive analysis of PDF-based AML attacks is provided in [43]. On the same topic, a methodology designed to evade structural PDF malware detection systems is presented in [44]. Similarly, Adobe Flash files may be perturbed by acting both on structural features which do not alter the functionalities, and content features through the addition of specific ActionScript commands. In this context, authors of [45] use a bisect line search algorithm for finding the most efficient step along the loss gradient direction for producing the adversarial sample. Windows malware are targeted in [26], where a genetic algorithm is used for generating adversarial executables, in which only a small set of functionality-preserving manipulations have been applied, such as header fields changes, slack space filling, or shifting the content before the start of a program section. Attackers may insert malicious code in users' browser, which will be activated upon visit of webpages and usually imply the gathering of sensible data contained in cookies. By targeting specific aspects of HTML and JavaScript syntaxes, authors of [66] leverage *Soft Q-learning* for creating adversarial samples able to evade cross-site scripting detectors. Considering the robustness of ML models deployed in this scenario, authors of [28] propose a method for learning uniformly distributed feature weights, which have been shown to strengthen the resistance of the model towards adversarial attacks, since adversaries have to perturb a larger number of features in order to succeed in their intent. *Intrusion Detection Systems* (IDSs) are also frequently considered as targets of AML attacks. In [3], the *Jacobian-based Saliency Map Attack* is leveraged in order to raise false alarms for short-circuit faults and other sensible threats. Authors of [2] use several approaches for creating adversarial traffic vectors for camouflaging malicious network flows, such approaches include *Generative Adversarial Networks* and genetic algorithms. It has been observed that features of network traffic present several statistical correlations, and the approach in [53] addresses such aspect by proposing a black-box attack that leverages the Mahalanobis distance between traffic vectors.

Other relevant scenarios include *electronic healthcare* [32], *biometric authentication* [6], *recommender systems* [64], *graph-based ML* [14, 80], *mobile edge computing* [77, 79], *wireless network security* [39, 60], and *spam detection* [23]. With regard to the last topic, it is worth highlighting the difference between *spam* detection, which refers to the identification of *unwanted content*, and *spam account* detection, which instead aims to distinguish *spammers*, be they human or bots, from *genuine users*.

Over the years, spam detection has evolved from naive systems capable of recognizing common spammy words [20], to more reliable algorithms that consider wide sets of interconnected features. This led to the definition of more sophisticated attack strategies. The authors of [59], for instance, propose a *sparse* evasion attack based on L_1 norm aiming at adding or removing specific terms for letting a *Support Vector Machine* recognize spam emails as genuine. Similarly, by performing several other manipulations such as synonym replacement, ham word injection and spam word spacing, adversaries can fool a Bayesian model trained to detect spam emails [37]. In [76] the impact of feature selection on evasion attacks is evaluated; in particular, it has been observed that a drastic decrease in the number of features easily allow adversaries to fool the ML spam detectors by altering only few words in emails.

Table 1. Distance norms and threat models of relevant attacks at the state-of-the-art. Abbreviations: (I)ntegrity/(A)vailability, (T)argeted/(U)ntargeted; (W)hite-/(G)ray-/(B)lack-box; (E)vasion/(P)oisoning.

Domain	*Ref.*	*Norm*	*Threat Model*		
Image Processing	[33]	L_∞	*I, U*	*W*	*E*
	[29]	L_∞	*I, A, U*	*B*	*P*
	[51]	L_1	*I, U*	*W, B*	*E*
	[47]	L_2	*I, U*	*W*	*E*
	[11]	L_0, L_2, L_∞	*I, T*	*W*	*E*
	[15]	L_2	*I, U*	*B*	*E*
	[71]	L_2	*I, U*	*W*	*E*
Malware Detection	[27]	L_0	*I, T*	*W, B*	*E*
	[54]	L_0	*I, T*	*W*	*E*
	[44]	L_0	*I, T*	*B*	*E*
	[45]	L_1	*I, T*	*W*	*E*
	[28]	L_1	*I, T*	*W*	*E*
	[26]	L_1	*I, T*	*B*	*E*
Intrusion Detection	[2]	L_0	*I, T*	*B*	*E*
	[3]	L_0	*I, T*	G	*E*
	[53]	L_2	*I, U*	B	*E*
Speech Recognition	[57]	L_ω	*I, T*	*W*	*E*
	[5]	L_∞	*I, T, U*	*B, W*	*E*
	[67]	L_∞	*I, T*	*G*	*E*
Spam Email Detection	[37]	L_0	*I, U*	*W*	*E*
	[59]	L_1, L_2	*I, T*	*W*	*E*
	[76]	L_1	*I, T*	*W*	*E*
Other	[32]	L_0	*I, U*	*W*	*E*
	[6]	L_1	*I, T*	*B*	*E*
	[64]	L_0	*A, T*	*G*	*P*
	[66]	L_0	*I, T*	*B*	*E*
	[80]	L_0	*A, I, T*	*B*	*E, P*
	[14]	L_0	*I, U*	*W*	*E*
	[79]	L_0	*I, T*	B	*E*
	[77]	L_0	*I, T*	W	*E*
	[60]	$L_\dagger$	*A, U*	B	*P*
	[39]	L_2	*I, U*	B	*E*

Spam account detection exploiting ML algorithms has also been discussed in a number of works [19, 22]. State-of-the art solutions usually exploit feature sets that are aimed to capture different attributes of a spammer, such as its connection with the rest of the social network, or data/metadata associated with the content shared [18]. These characteristics can be analyzed by means of a variety of models that typically include Neural Networks (NNs), Support Vector Machines (SVMs), and Random Forests (RFs), where the last one proved to be the most proper classifier when dealing with large feature sets [70]. Spam account detection in OSNs clearly has unique traits compared to other application domains because of the many ways a user can operate within social networks, hiding its malicious behavior, and thus achieving its disturbance objective. However, the study of the literature has revealed that the only intersections between AML and OSNs analysis regards *fake news* and *social bots* detection systems [21]. Thus, to the best of our knowledge, this could be one of the first papers addressing OSNs' account manipulation through AML.

A summary of the related work is reported in Table 1, which highlights the characteristics of each approach according to the properties discussed in Section 2.

4 METHODOLOGY

4.1 Scenario

OSNs are protected by intelligent systems that are able to identify and block malicious accounts. Usually, the detection algorithms exploit heterogeneous features to describe the different behaviors that spammers may adopt [18]; these can be logically organized into four categories, according to their aims [31]:

- **Metadata-based features** describe the general characteristics of the account, such as its creation date, the geographic location, or the average tweet time. These features can be obtained very easily and can be quite effective in recognizing clear malicious behaviors.
- **Content-based features** are useful to evaluate the quality of the content shared by an account. In order to be effective, spammers need to reach a large number of users; thus, their messages frequently include mentions, hashtags, and URLs. Spam account detection algorithms may look for these elements in order to determine whether an account is genuine or not.
- **Interaction-based features** allow to model the friendship network of the account under analysis. Neighborhood information can be very useful to distinguish influential accounts (characterized by many followers and generally by almost no followings), listener accounts (few followers and many followings), isolated accounts, or even sub-networks that could be used for orchestrated attacks.
- **Community-based features** are able to capture the characteristics of account groups that have similar interests, physical locations, professions or other relevant social aspects. The general idea is that the behavior of an account can be inferred by observing those of the community to which it belongs [31], e.g., an account with a good reputation network is unlikely to be a spammer.

In order to bypass the detection systems, the attacker should make the feature vector describing a spammer resemble that of a genuine user. This goal can be achieved through trial and error strategies, however deceiving the OSN can take a long time, during which actions (e.g., banning or blacklisting) can be taken against the spammer. Altering *metadata*-based features, for instance, would require a shift in the account habits, which is very complicated to achieve. *Content*-based features may be altered by forging ad hoc content in order to re-balance those feature values that might suggest a malicious behavior, e.g., creating new "clean" tweets, or removing those that clearly refer to a spam campaign. These alterations may be implemented manually or automatically, depending on the nature of the user. In either case, a large number of changes are required for the features to undergo a significant change. *Interaction-* and *community*-based features can also be modified by creating new connections within the social network, which is usually done by purchasing followers or followings [72] from third-party providers, by creating fake accounts, or by exchanging followers with other users.

4.2 AML Attack Strategy and Threat Model

The proposed attack strategy (see Figure 1) follows the general structure of black-box adversarial attacks [51] while capturing the peculiarities of the scenario just described.

The attacker (*Darth*) aims to perform an *integrity* violation of the defense mechanism of the OSN (i.e. *Spam Account Detector*) so as to be *mis*-classified as a genuine user, although showing a typical spammer behavior. The attack specificity is *targeted* since the intent is to hide the spammer behavior to the smart detector.

Even though in a *black-box* scenario the target model $\mathcal{T}$ is not known to Darth, we can make a common assumption [29] by supposing the adversary knows the feature representation $\mathcal{X}$ of the

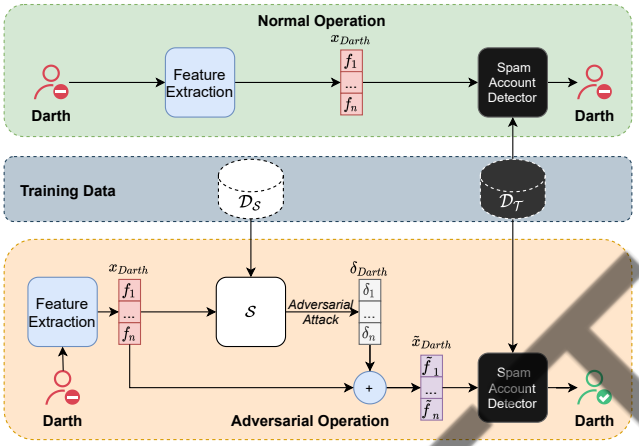


Fig. 1. The proposed AML strategy in OSN security scenario.

data. The model $\mathcal{T}$ can be queried by Darth, while meeting time constraints and consecutive query limits, in order to collect a training dataset $\mathcal{D}_{\mathcal{S}}$ for a local *surrogate* model $\mathcal{S}$, which mimics the functioning of the target (unknown) ML system; this configures our approach as a query-limited setting black-box attack [34]. In particular, $\mathcal{D}_{\mathcal{S}}$ can be obtained by following a strategy known as *mimicry attack* [28, 50], according to which a group of satellite agents is exploited to create a balanced set of spam and genuine accounts. Since Darth knows $\mathcal{S}$, he can build a perturbation vector δ to be added to the original feature vector x so that the perturbed feature vector $\tilde{x}$ is classified by $\mathcal{S}$ as genuine. Finally, by exploiting the *transferability* property, Darth may obtain that $\tilde{x}$ eludes even the unknown model $\mathcal{T}$.

The computation of δ must satisfy some constraints; indeed, in the scenario considered here, some features may be mutually dependent and therefore not all δ values are valid. Some works, addressing other application scenarios [53], have proposed the generation of perturbations capable of preserving the *statistical* correlation between features; this means that a change in a given feature value should propagate proportionally to the others. However, such a *numerical* dependency, although being able to highlight the hidden relationships between feature values, is not really able to describe the *semantic* dependencies of the elements of the feature set. More specifically, we define two features to be *semantically dependent* if their computations require one or more common raw data.

The remainder of the section describes the proposed attack algorithm. For the sake of clarity, the adopted notations and abbreviations are listed in Table 2.

4.3 AdverSPAM Algorithm

The requirements discussed so far translate into ensuring the fulfilment of three constraints, namely (i) allowing the computation of an adversarial sample, $\tilde{x}$, beyond the decision boundary of the surrogate model, $\mathcal{S}$, while maintaining the (ii) the *statistical correlation* and (iii) the *semantic dependency* of the features. In order to better explain the meaning of these constraints, we complement the description with figures that illustrate a simplified classification example in which only two features are employed (see Figure 2).

Given the problem of associating an observation $\{x_1, \ldots, x_\kappa\}$ with a class from the binary set $\Omega = \{\omega^-, \omega^+\}$, the goal of the attacker is to take an original sample, $x \in \mathcal{X}$, that lies in the region ω^+ and project it into ω^- so obtaining a perturbed sample $\tilde{x}$. Since low-complexity surrogate models

Table 2. Notations used in the paper.

Symbol	Description
$\mathcal{L}$	A general learning model.
$\mathcal{S}$	Surrogate model.
$\mathcal{T}$	Target model.
θ	Set of $\mathcal{L}$'s adjustable parameters.
$\ell(\theta, x, y)$	Loss function of $\mathcal{L}$.
$\mathcal{X}$	Set of input samples.
x	Original sample.
$\tilde{\mathcal{X}}$	Set of perturbed samples.
$\tilde{x}$	Adversarial sample.
$\hat{y}$	Predicted label.
δ	Adversarial perturbation.
$L_p(\tilde{x}, x)$	p-norm between original and perturbed samples.
$db_{\mathcal{S}}$	Decision boundary for $\mathcal{S}$.
α_i	i-th coefficient of $db_{\mathcal{S}}$.
β	Intercept of $db_{\mathcal{S}}$.
$\mathcal{R}_{j,i}$	Linear regression between x_j, x_i.
$m_{j,i}$	Slope of $\mathcal{R}_{j,i}$.
$q_{j,i}$	Intercept of $\mathcal{R}_{j,i}$.
$z(\tilde{x}, x)$	Cost function for $\tilde{x}$.
λ	Controlling factor for $z(\tilde{x}, x)$.

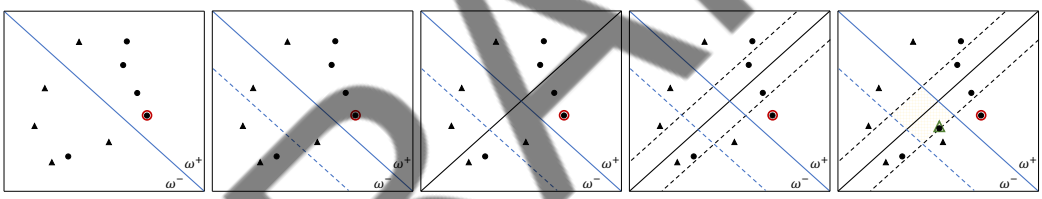


Fig. 2. Simplified example of binary classification with only two features. (a) The decision boundary $db_{\mathcal{S}}$ (blue line) separates spammers (circles) from genuine users (triangles). The goal of the attack is to project one chosen spammer sample (red) into the opposite region by crossing the decision boundary. (b) The adversarial sample must be generated within a certain distance (dashed blue line) from the decision boundary, which depends on a parameter ψ. (c) The regression line (black) provides a good approximation of the correlation between the spammer samples. (d) In order to preserve the nature of the input, the feasible region (yellow area) for the adversarial sample is further constrained by a margin around the regression line. (e) The adversarial sample (green triangle) is finally computed by solving the optimization problem within the feasible region.

have been shown to transfer attacks more effectively [29], we assume $\mathcal{S}$ to be any model based on a *linear* decision function:

$$db_{\mathcal{S}} : \sum_i \alpha_i x_i + \beta = 0, \tag{8}$$

where α_i and β are the learned coefficients and intercept, respectively. Therefore, crossing the decision boundary $db_{\mathcal{S}}$ means performing a search in one of the two regions (see Figure 2a) delimited by Eq. 8. To be more specific, every perturbed sample $\tilde{x}$ must satisfy one of the following conditions:

$$\begin{cases} \sum_i \alpha_i \tilde{x}_i + \beta > 0 & \text{if } \omega^- \text{ is above } db_{\mathcal{S}}, \\ \sum_i \alpha_i \tilde{x}_i + \beta < 0 & \text{otherwise.} \end{cases} \tag{9}$$

Algorithm 1 Crossing the Decision Boundary

Input:
$\mathcal{S}$: Surrogate model to attack;
$\mathcal{X}$: The set of input samples for $\mathcal{S}$.
Output:
db_c: The decision boundary constraint;
db_ψ: The constraint on the parallel to $db_\mathcal{S}$.

$db_\mathcal{S} \leftarrow \mathcal{S}$.getDecisionBoundary()
$[\alpha_i, \beta] \leftarrow db_\mathcal{S}$.getParameters()
$x \leftarrow \mathcal{X}$.getRandomSample()
$\hat{y} \leftarrow \mathcal{S}$.predict$(x)$
$\psi \leftarrow$ getOffset$(db_\mathcal{S})$
if test$(x, [\alpha_i, \beta]) \geq 0$ **then**
  if $\hat{y} \in \omega^-$ **then**
    $db_c \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta > 0$"
    $db_\psi \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta - \psi < 0$"
  else
    $db_c \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta < 0$"
    $db_c \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta + \psi > 0$"
else if test$(x, [\alpha_i, \beta]) < 0$ **then**
  if $\hat{y} \in \omega^-$ **then**
    $db_c \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta + \psi < 0$"
    $db_\psi \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta + \psi > 0$"
  else
    $db_c \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta - \psi > 0$"
    $db_\psi \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta - \psi < 0$"
return db_c, db_ψ

It is also useful to choose $\tilde{x}$ on the basis of its distance from the decision boundary. In fact, the farther $\tilde{x}$ is from $db_\mathcal{S}$, the greater the probability of evasion success on the target model. On the other hand, points that are too far from the decision boundary would exhibit characteristics too dissimilar from x, resulting in the attack being meaningless. Thus, we chose to regulate the maximum allowed distance from the decision boundary (Figure 2b) through a parameter ψ, whose sign depends on the position of ω^- with respect to the $db_\mathcal{S}$:

$$\begin{cases} \sum_i \alpha_i \tilde{x}_i + \beta - \psi < 0 & \text{if } \omega^- \text{ is above } db_\mathcal{S}, \\ \sum_i \alpha_i \tilde{x}_i + \beta + \psi > 0 & \text{otherwise.} \end{cases} \tag{10}$$

The procedure that implements this last constraint is described by Algorithm 1.

The search in the region beyond the decision boundary is further driven by the need to ensure that the *statistical correlation* and the *semantic dependency* of the features originally extracted from x are preserved in the forged adversarial sample $\tilde{x}$. These two properties are defined as follows.

Definition 1. *Statistical Correlation: two features are statistically correlated if having a strong linear relationship with each other.*

Definition 2. *Semantic Dependency: two features are semantically dependent if their computations require one or more common raw data.*

In *AdverSPAM*, the *statistical correlation* among features is leveraged by considering the regression line fitting the data. In particular, since the aim of the adversary is to preserve the core properties of the *spammers*, only the samples of the positive class ω^+ are considered (see Figure 2c). Formally,

Algorithm 2 Maintaining Correlations and Dependencies

Input:
$\mathcal{X}$: The set of input samples for $\mathcal{S}$.
Output:
$\tilde{C}$: List of constraints.

$C \leftarrow [\,]$
for all $x \in \mathcal{X}$ **do**
  for all $i \in \mathcal{F}$ **do**
    for all $j \in \mathcal{F}$ **do**
      $sd \leftarrow$checkSemanticDependence$(i,\ j)$
      if $i \neq j$ **and** $sd == True$ **then**
        $\mathcal{R}^{\omega^+}_{j,i} \leftarrow$ getRegressionLine$(\mathcal{X},\ j,\ i)$
        $[m^{\omega^+}_{j,i}, q^{\omega^+}_{j,i}\,] \leftarrow \mathcal{R}^{\omega^+}_{j,i}$.getParameters()
        $margin_{j,i} = \text{sqrt}(\sum_{x \in \mathcal{X}} (\mathcal{R}^{\omega^+}_{j,i}(x_i) - x_j)^2)$
        C.add("$\tilde{x}_j \leq m^{\omega^+}_{j,i}\ \tilde{x}_i + q^{\omega^+}_{j,i} + (margin_{j,i})$")
        C.add("$\tilde{x}_j \geq m^{\omega^+}_{j,i}\ \tilde{x}_i + q^{\omega^+}_{j,i} - (margin_{j,i})$")
return C

the regression line regarding features j and i for all samples $x \in \omega^+$, is defined by:

$$\mathcal{R}^{\omega^+}_{j,i}(x_i) = m^{\omega^+}_{j,i} x_i + q^{\omega^+}_{j,i}, \tag{11}$$

where $m^{\omega^+}_{j,i}$ and $q^{\omega^+}_{j,i}$ represent its slope and intercept, respectively. In order to maintain the *statistical correlation* between the features, it is therefore necessary that the manipulated sample does not deviate too far from $\mathcal{R}^{\omega^+}_{j,i}$. Hence, $\tilde{x}$ must lie within a certain *margin* from this line (see Figure 2d), that we compute as the squared root of the Residual Sum of Squares (RSS) of the regression model:

$$margin_{j,i} = \sqrt{\sum_{x \in \mathcal{X}} (\mathcal{R}^{\omega^+}_{j,i}(x_i) - x_j)^2}, \tag{12}$$

where the squared root is adopted for dimensional homogeneity. When RSS is approximately 0, the regression line is a good predictor of the data, resulting in a particularly tight margin of allowable displacement; conversely, a high value of RSS means $\mathcal{R}^{\omega^+}_{j,i}$ is an unreliable model of the data, thus leading to a wide margin of possible perturbations. In other words, this margin actually outlines the minimum and maximum perturbations permitted on the j-th feature of $\tilde{x}$:

$$\begin{cases} \tilde{x}_j \leq m^{\omega^+}_{j,i}\ \tilde{x}_i + q^{\omega^+}_{j,i} + (margin_{j,i}) \\ \tilde{x}_j \geq m^{\omega^+}_{j,i}\ \tilde{x}_i + q^{\omega^+}_{j,i} - (margin_{j,i}). \end{cases} \tag{13}$$

Furthermore, we adopt the notation $sd(i, j)$ to indicate whether a *semantic dependency* between the features i and j exists. As stated in *Definition 2*, such a relationship occurs when multiple features capture similar traits of the account, so requiring the same information (e.g., the number of followers, or the amount of posted URLs) to be computed. Therefore, the maintenance of *statistical correlation* is strictly related to the existence of *semantic dependency*:

$$\begin{cases} \tilde{x}_j \leq m^{\omega^+}_{j,i}\ \tilde{x}_i + q^{\omega^+}_{j,i} + (margin_{j,i}) & \forall i, j\ s.t.\ sd(i,j) = true, \\ \tilde{x}_j \geq m^{\omega^+}_{j,i}\ \tilde{x}_i + q^{\omega^+}_{j,i} - (margin_{j,i}) & \forall i, j\ s.t.\ sd(i,j) = true. \end{cases} \tag{14}$$

Conversely, features that are not in a direct cause-effect relationship can be manipulated independently of each other. The steps required to compute these last constraints are described in Algorithm 2.

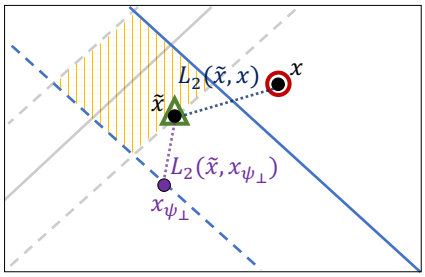


Fig. 3. The objective function conjugating the minimisation of the distance imposed over the adversarial samples and the maximization of the evasion capability against unknown classifiers.

Once the feasible region has been defined (Figure 2e), the attack proceeds by determining the *best* adversarial sample within it. This corresponds to finding a set of new feature values which pursues a twofold purpose. Firstly, the overall distance of the adversarial sample $\tilde{x}$ from the input x has to be limited, in order to ensure that the (malicious) nature of x is not actually altered. Secondly, the adversarial sample has to be transferable, that is, $\tilde{x}$ has to be general enough to evade the unknown target classifiers. We modeled these two aspects as a weighted sum [25] of two objectives controlled by a factor $\lambda \in [0, 1]$:

$$z(\tilde{x}, x) : \lambda L_2(\tilde{x}, x) + (1 - \lambda) L_2(\tilde{x}, x_{\psi_\perp}), \tag{15}$$

where $x_{\psi_\perp}$ is the projection of the input sample x over the parallel to the decision boundary.

In order to understand the meaning of this objective function, let us consider Figure 3. If the need of the adversary is to evade the local surrogate model with the minimum effort possible, then, setting $\lambda = 1$ allows to only perform the minimization of the euclidean distance between the input and the adversarial sample. Conversely, when the adversary wants to deceive unknown black-box models, since their decision boundaries may greatly differ from the one of the local surrogate, a reasonable amount of additional distance has to be imposed over the adversarial sample; this can be achieved by setting $\lambda = 0$. Any other value of $\lambda \in [0, 1]$ will constitute a trade-off between such two opposing situations. Hence, *AdverSPAM* calculates the final adversarial sample by solving the following optimization problem:

$$\begin{aligned}
&\tilde{x} = \min_{\tilde{x}} \ \lambda L_2(\tilde{x}, x) + (1 - \lambda) L_2(\tilde{x}, x_{\psi_\perp}), \\
&s.t. \\
&\text{(as reported in Eq. 9):} \quad \sum_i \alpha_i \tilde{x}_i + \beta > 0 \quad \textbf{or} \quad \sum_i \alpha_i \tilde{x}_i + \beta < 0 \\
&\text{(as reported in Eq. 10):} \quad \sum_i \alpha_i \tilde{x}_i + \beta - \psi < 0 \quad \textbf{or} \quad \sum_i \alpha_i \tilde{x}_i + \beta + \psi > 0 \\
&\text{(as reported in Eq. 14):} \quad \tilde{x}_j \leq m^{\omega^+}_{j,i} \ \tilde{x}_i + q^{\omega^+}_{j,i} + (margin_{j,i}) \quad \forall i, j \ s.t. \ sd(i, j) = true \\
&\qquad\qquad\qquad\qquad\quad \tilde{x}_j \geq m^{\omega^+}_{j,i} \ \tilde{x}_i + q^{\omega^+}_{j,i} - (margin_{j,i}) \quad \forall i, j \ s.t. \ sd(i, j) = true \\
&\qquad\qquad\qquad\qquad\quad \tilde{x} \in [0, 1]^n
\end{aligned} \tag{16}$$

The entire procedure is described by Algorithm 3, which exploits the other two algorithms discussed in this section. In our implementation, the problem was solved using COBYLA [56], which operates iteratively by generating local linear approximations of the objective function and constraints. The solution $\tilde{x}$ is searched in $[0, 1]^n$, which represents the range of admissible values for the feature set of n elements that will be described in the next section.

Algorithm 3 AdverSPAM

Input:
$\mathcal{S}$: Surrogate model to attack;
$\mathcal{X}$: The set of input samples for $\mathcal{S}$;
Output:
$\tilde{\mathcal{X}}$: The set of adversarial samples.

$\tilde{\mathcal{X}} \leftarrow [\,]$
$db_c, db_\psi \leftarrow$ Algorithm1($\mathcal{S}$, $\mathcal{X}$)
for all $x \in \mathcal{X}$ **do**
  $C \leftarrow [\,]$
  C.append(db_c)
  C.append(db_ψ)
  C.append(Algorithm2($\mathcal{X}$))
  $z \leftarrow$ getCostFunction($\tilde{x}$, x)
  $\tilde{x} \leftarrow$ solveoptimizationProblem($'min'$, x, z, C)
  $\tilde{\mathcal{X}}$.append($\tilde{x}$)
return $\tilde{\mathcal{X}}$

5 EXPERIMENTAL ANALYSIS

The effectiveness of the proposed technique[2] has been evaluated in different steps. The first part of this section describes the experimental setup, the metrics adopted for the assessment, as well as the tuning of *AdverSPAM* internal parameters. Then, comparisons with five state-of-the-art attack techniques are presented, discussing both the performance obtained and the quality of the forged adversarial samples. The section continues with the introduction of a concrete case study aimed at exploring the manipulations that should be applied to a real OSN account in order to accomplish the considered attacks. The results of adopting three mitigation strategies follow; finally, advantages and limitations of *AdverSPAM* are outlined.

5.1 Experimental Setup

The choice of the classifiers to adopt as *target* and *surrogate* models was driven by the analysis of the literature in the field of AML. A preliminary consideration concerned whether to select ML or Deep Learning methods. When low-dimensional features spaces are considered, it has been shown that classical ML algorithms can produce superior results, which also tend to be better interpretable, than those based on deep neural networks [35]. Conversely, the latter are recommended especially in domains characterized by large, high-dimensional data, such as image, video, and audio data processing [24]. Hence, we focused on five classifiers that are the most commonly chosen for the final assessment of adversarial attacks [7, 63], namely: Neural Network (NN), Support Vector Machine (SVM), Logistic Regression (LR), Ridge Regression (RR), and Random Forest (RF). All the considered models has been examined as possible targets, while only SVM (with linear kernel) and LR were chosen as surrogates since they satisfy the requirement of linear decision boundary needed in *AdverSPAM*.

Moreover, since the attack transferability strongly depends on the complexity of the target model [29], our assessment considers high-complexity (H) and low-complexity (L) variants. The complexity of a ML model is measured by the number of hyperparameters it has. In general, a model characterized by a large set of hyperparameters (high degree of complexity) may be able to capture more variations in the data, but it will also be more difficult to train and may be more prone to overfitting. Conversely, a low complexity model may be easier to train, but may not be able to capture all the relevant information in the data.

[2]https://github.com/agiammanco94/AdverSPAM

Table 3. The list of features used to model the accounts.

Category	Acronym	Description
Metadata	RR	Retweet Ratio
	AR	Automated Tweet Ratio
	TSD	Tweet Time Standard Deviation
	TISD	Tweet Time Interval Standard Deviation
Content	UUR	Unique URL Ratio
	UMR	Unique Mention Ratio
	CHS	Content and Hashtag Similarity Ratio
	UR	URL Ratio
	MR	Mention Ratio
	HTR	Hashtag Ratio
	AUR	Automated Tweet URL Ratio
	ATS	Automated Tweet Similarity
Interaction	FR	Follower Ratio
	MFFFR	Mean Follower's Followings to Follower Ratio
	FBR	Follower-based Reputation
	R	Reputation
	CC	Clustering Coefficient
Community	CBR	Community-based Reputation
	CBCC	Community-based Cluster Coefficient

In order to properly tune the parameters of the classifiers, we performed a 10-fold Cross-Validation with the objective of letting all the models achieve a f-score of about 90%. To this aim, a public dataset [31] consisting of 10.000 *genuine* users and 1.000 *spammers*, each described by a set of 19 features (see Table 3), was used. Being this dataset unbalanced, we employed *SMOTE* augmentation technique [13] so as to obtain a new dataset of 10.000 users per class. This was split with 80:20 ratio leading to 16.000 accounts for the training and 4.000 for the test set. The former was further split in two parts representing the datasets $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{T}}$ (see Figure 1), each containing 4.000 *genuine* and 4.000 *spammer* accounts. Finally, $\mathcal{D}_{\mathcal{T}}$ was also used to perform the adversarial training described in Section 5.5.1.

A summary of the adopted models and their tuned parameters is provided in Table 4.

5.2 AdverSPAM Assessment

The behavior of *AdverSPAM* is mainly influenced by two parameters, namely ψ (Eq. 10) and λ (Eq. 15). Given the definition of ψ, we calculate it as the value that guarantees a desired percentage (*ratio*) of samples of the opposite class ω^- lies between the decision boundary and its shift by a quantity ψ. The higher the *ratio*, the greater the probability that the adversarial sample will be located in a region with a higher density of ω^- samples. On the other hand, greater distances from the decision boundary would lead to an over-distortion of the original sample. For this reason, the choice of the best ψ value was determined on the basis of a set of evaluation metrics.

In particular, being the main goal of the algorithm to create a perturbed sample capable of deceiving the target model, a good measure of its effectiveness is the percentage of actual *spammers* that are misclassified as *genuines*. This information is provided by the *False Negative Rate* (*FNR*), which is defined as the ratio between false negatives and true positives. Moreover, the L_2 and L_∞ distance norms, defined in Section 2, can be exploited to evaluate the distortion introduced in the adversarial samples.

Tests were run on four surrogate models while varying both the *ratio* and the value of λ, i.e., the weight of the two components of the objective function. Results are shown in Figure 4, organized

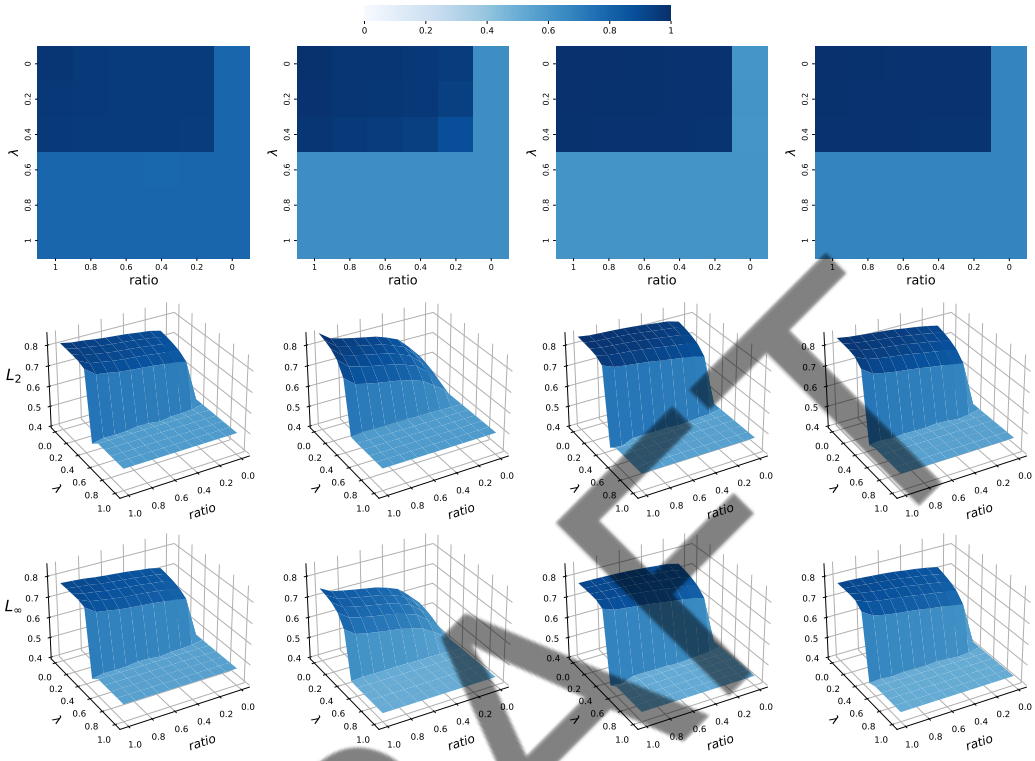
Table 4. The surrogate and target models used in the experiments, and the corresponding tuned parameters. For all the models, low (L) and high (H) complexity variants are evaluated. Furthermore, two different SVM kernels are considered, namely, linear (lnr) and radial basis function (rbf).

Surrogates	SVM^L_{lnr}, SVM^H_{lnr}, LR^L, LR^H	
Targets	RF^L, RF^H, NN^L, NN^H, SVM^L_{lnr}, SVM^H_{lnr}, SVM^L_{rbf}, SVM^H_{rbf}, LR^L, LR^H, RR^L, LR^H	
Parameters	RF^L	trees = 30; max_depth = 8
	RF^H	trees = 30; max_depth = no-limit
	NN^L	learning_rate = 0.01; weight_decay = 0.01; neurons_layers = [50, 50, 2]
	NN^H	learning_rate = 0.01; weight_decay = 0; neurons_layers = [50, 50, 2]
	SVM^L_{lnr}	C = 1
	SVM^H_{lnr}	C = 100
	SVM^L_{rbf}	C = 1
	SVM^H_{rbf}	C = 100
	LR^L	C = 1
	LR^H	C = 10
	RR^L	α = 10
	RR^H	α = 1

as a matrix in which each column refers to a different surrogate. The first row summarizes the mean FNR values obtained on the 12 target models for every pair $(ratio, \lambda)$; highest values are represented with darker colors. The 3-D plots in the second and third rows reveal how varying the parameters $ratio$ and λ impacts on the values of L_2 and L_∞. As a general trend we can notice that both the FNRs and the distance values grow proportionally to $ratio$, but inversely to λ. Indeed, small values of λ (i.e., $\lambda \leq 0.4$) cause the objective function to push the adversarial sample away from the decision boundary, which leads to both a greater transferability and higher peaks in the distance metrics.

As the considered surrogate model varies, the specific optimal values of the parameters change accordingly. In particular, by analyzing the last two columns of Figure 4 (i.e., LR^L and LR^H models), it can be noticed that the best FNRs values are obtained when the $ratio \in [0.2, 1]$ and $\lambda \in [0, 0.4]$. Considering the L_2 and L_∞ values measured in these ranges, a good trade-off between success rate and perturbation degree is reached by choosing $ratio = 0.2$ and $\lambda = 0.4$. The same assessment can be made for the SVM^L_{lnr} and SVM^H_{lnr} models, resulting in $ratio = 1$ and $\lambda = 0.2$. The obtained $ratio$ values correspond to ψ equals to 1.91, 5.44, 3.78, and 5.47 for SVM^L_{lnr}, SVM^H_{lnr}, LR^L, and LR^H, respectively.

Further experiments were carried out in order to evaluate the capability of *AdverSPAM* to preserve the statistical correlations of the features. In literature, statistical correlation is expressed in terms of linear correlation between pairs of features; an indicator typically used in this regard is the Pearson's coefficient [4, 38, 40, 69]. In order to assess what type of correlation exists between the 19 features considered, some preliminary tests were carried out on the original dataset by representing all the pairs of features in a two-dimensional space and fitting them with polynomials



(a) SVM^L_{lnr} (b) SVM^H_{lnr} (c) LR^L (d) LR^H

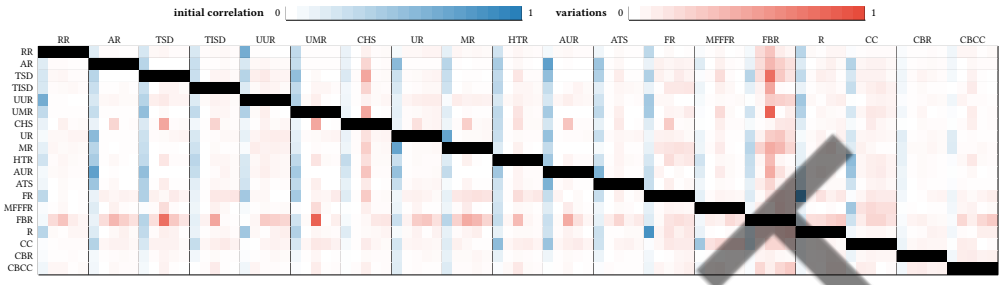
Fig. 4. AdverSPAM tuning. Mean FNRs over the 12 targets (top row), and L_2 (middle) and L_∞ (bottom) distances measured while varying the parameters *ratio* and λ for every surrogate model (columns).

Table 5. Average and variance of the Mean Square Deviation (MSD) computed by different degree polynomials fitted on the features of the *spammer* class.

metric \ degree	I	II	III	IV
avg(MSD)	.032	.029	.028	.027
var(MSD)	.002	.002	.002	.002

of degree from 1 to 4. For each pair, the Mean Square Deviation (MSD) of the points w.r.t. the fitting polynomials was calculated as an indicator of approximation quality. The results shown in Table 5 indicate that polynomials of a higher degree correspond to a smaller fitting error, which is intrinsic in the fact that the higher the degree the greater the freedom in fitting the data. Thus, we also calculated the variance of the deviations, which should decrease if a certain curve was actually able to better follow the distribution of the data. The results, instead, indicate that the variance is stable as the degree of the polynomial changes; thus, we can conclude that linear dependency fairly accurately represents the distribution of the data.

Table 6. Correlation matrix. For each feature, the following 5 values (columns) are shown: Pearson correlation coefficients (blue) measured on the samples of the spammer class before the attack is launched; variations of these values (red) after creating the adversarial samples with SVM^L_{lnr}, SVM^H_{lnr}, LR^L, and LR^H, respectively.



The correlation matrix of the feature set before and after the adversarial attack was performed is shown in Table 6, where, for each pair of features, the following 5 values (columns) are reported: original Pearson correlation coefficients (blue); variations of these coefficients (red) after creating the adversarial samples with SVM^L_{lnr}, SVM^H_{lnr}, LR^L, and LR^H, respectively. As a general rule, the stronger the correlation between the features, the smaller the changes in Pearson's coefficients caused by the attack should be. Thus, if correlations are maintained, light-red colored cells are expected in correspondence with dark-blue ones, and vice versa; this trend is clearly visible for almost any feature in Table 6. Moreover, because of the constraint in Eq. 14, features that are not semantically dependent on each other are more likely to be strongly modified. This is particularly evident when looking at FBR (Follower-based Reputation), which belongs to the *interaction* category (see Table 3). Features ranging from RR to ATS are not semantically dependent with FBR because they belong to different categories; moreover, they are lowly correlated with FBR as indicated by the corresponding Pearson coefficients. Hence, these features are good candidates for manipulation. This is indeed confirmed by the results, which show greater variations (darker red values) in the rows from RR to ATS, regardless of the surrogate model used.

5.3 Comparing AdverSPAM with the baselines

State-of-the-art AML attacks can be classified into two main categories, namely white-box and black-box [30, 58]. In order to cover both classes, we decided to compare AdverSPAM with gradient-based white-box attacks (i.e., *FGSM*, *DF*, and *C&W*), and decision-based black-box attacks (i.e., *Cheng* and *Peng*). While *FGSM*, *DF*, and *C&W* attacks are widely adopted in the literature for the comparative analyzes [5, 17, 27, 36, 49, 52, 73, 75, 78], the other two methods are less known but no less important. In particular, to the best of our knowledge, *Peng* is the only technique that addresses the problem of statistical correlation between features, just like AdverSPAM. Thus, comparative analysis is based on the following baselines:

- *Fast Gradient Sign Method* (**FGSM**) [33], where the adversarial perturbation is computed considering the sign of the targeted classifier loss function's gradient, and projecting the adversarial sample on a sphere of radius ϵ_{FGSM} around the input sample.
- *DeepFool* (**DF**) [47], where the adversarial sample is shifted along the direction of the gradient of the target model loss w.r.t. the input, until the decision boundary is crossed. The obtained perturbation is then scaled by a factor $(1 + \eta_{DF})$, which is useful for getting adversarial samples farther from the decision boundary.
- *Carlini and Wagner* L_2 (**C&W**) [11], where an optimization problem is formulated with a loss function leveraging the representation in the *logits* layer (the layer prior to the final softmax

Table 7. The chosen parameters for the adopted algorithms.

	FGSM	**DF**	**C&W**
Parameter / Model	ϵ_{FGSM}	η_{DF}	$c_{C\&W}$
SVM^L_{lnr}	0.37	0.35	0.5
SVM^H_{lnr}	0.42	0.75	0.5
LR^L	0.45	0.5	0.85
LR^H	0.45	0.2	0.95

layer, containing the probabilities that a sample belongs to a specific class) as a measure of attack success. The properties of such a loss function depend on a regularization parameter $c_{C\&W} > 0$, which controls the confidence that the adversarial sample belongs to the opposite class. When the attacked model is not a Neural Network, the equivalent of the logits layer is represented by the class memberships probability.

- *Cheng et al. method* (**Cheng**) [15], where the model to attack is considered as a black box, whose decision boundary is estimated by considering the vector connecting the input sample to a sample belonging to the desired class. The adversarial perturbation is computed taking incremental steps along such direction until the decision boundary is crossed; then, a gradient descent procedure is followed in order to consider the direction minimizing the euclidean distance between the input and the adversarial sample.
- *Peng et al. method* (**Peng**) [53], which is an extension of the *Cheng* method where the optimal direction on which to project the adversarial sample is searched by minimizing the Mahalanobis distance, thus preserving the statistical correlations between features.

The methods just introduced require a calibration phase to tune the corresponding parameters. Similarly to the analysis described in Section 5.2, we computed them by maximizing the *FNR*s of the target models, while minimizing the distance metrics. The chosen values are listed in Table 7, in which *Cheng* and *Peng* methods are omitted because their functioning parameters are indicated in the respective original works.

Table 8 reports the transfer matrices of the six attacks launched against twelve targets (columns), while using the surrogates previously discussed. For each subtable, the last column and row indicate the average transfer rate of the attacks and the average *FNR* against every target, respectively.

A first consideration can be made about the performance achieved using *SVM*s and *LR*s. Indeed, basing the attacks on logistic regression models, regardless of their complexity, results in a stronger transferability capacity than *SVM*s, as revealed by the values highlighted in green. This corroborates the thesis that simpler sources of adversarial samples are preferable for transferring against unknown models [29], as they are less specialized and thus less prone to drive the perturbations toward regions which will not translate in miss-classification against black boxes.

By analyzing Table 8 by columns, it is also possible to observe some dependency between the measured *FNR*s and the complexity of the target models. In particular, high-complexity targets seem to be less resistant to transferred adversarial samples, as pointed out by the average values highlighted in pink. For instance, when the surrogate is SVM^L_{lnr}, the average *FNR*s against NN^L and NN^H are 0.48 and 0.81, respectively. This could depend on the learned decision curve of higher complexity targets, which is highly fitted (very close) to the input data; thus, a small perturbation on the input sample is often sufficient to cross the decision boundary. The same consideration

Table 8. Transfer matrices (*FNR* values) of five attack strategies compared with AdverSPAM exploiting four different surrogate models, namely low and high complexity SVMs (a) and LRs (b). Each attack is carried out against 12 target models (columns).

SVM^L_{lnr}	NN^L	NN^H	RF^L	RF^H	SVM^L_{lnr}	SVM^H_{lnr}	SVM^L_{rbf}	SVM^H_{rbf}	LR^L	LR^H	RR^L	RR^H	*average*
AdverSPAM	1	1	.81	.81	1	1	1	1	1	1	1	1	.97
FGSM	.56	1	1	1	.99	1	.19	.99	.68	1	.92	1	.86
DF	1	.98	.74	.74	1	.99	1	1	1	1	1	1	.95
C&W	.27	.69	.03	.03	.73	.73	.10	.73	.42	.73	.71	.72	.49
Cheng	.02	.73	.43	.43	1	.98	0	1	0	.99	1	1	.63
Peng	.03	.48	.37	.37	.63	.58	0	.63	0	.63	.25	.57	.38
average	.48	.81	.56	.56	.89	.88	.38	.89	.52	.89	.81	.88	

LR^L	NN^L	NN^H	RF^L	RF^H	SVM^L_{lnr}	SVM^H_{lnr}	SVM^L_{rbf}	SVM^H_{rbf}	LR^L	LR^H	RR^L	RR^H	*average*
AdverSPAM	1	.99	.91	.91	.99	.99	1	.99	1	1	1	1	.98
FGSM	1	1	1	1	.80	1	.95	.72	1	1	1	1	.96
DF	1	.99	.99	.99	1	.99	1	.99	1	1	1	.99	1
C&W	.88	.97	.42	.42	.93	.99	.98	.90	.99	.99	.99	.99	.87
Cheng	.80	.95	.68	.68	.99	.99	.33	.99	.99	.99	.99	.99	.86
Peng	.23	.23	.15	.15	.23	.23	0	.23	.23	.23	.23	.23	.20
average	.82	.86	.69	.69	.82	.87	.71	.80	.87	.87	.87	.87	

SVM^H_{lnr}	NN^L	NN^H	RF^L	RF^H	SVM^L_{lnr}	SVM^H_{lnr}	SVM^L_{rbf}	SVM^H_{rbf}	LR^L	LR^H	RR^L	RR^H	*average*
AdverSPAM	1	1	.91	.91	1	1	.98	1	1	1	1	1	.98
FGSM	.76	1	1	1	1	1	.35	1	.94	1	.99	1	.92
DF	1	.99	.93	.93	1	1	.98	1	1	1	1	1	.99
C&W	.19	.36	.03	.03	.29	.68	.10	.34	.18	.36	.26	.66	.29
Cheng	0	.51	.48	.48	.88	.93	0	.95	0	.55	.54	1	.53
Peng	0	0	.06	.06	0	.02	0	.05	0	0	0	0	.02
average	.49	.64	.57	.57	.70	.77	.40	.72	.52	.65	.63	.78	

LR^H	NN^L	NN^H	RF^L	RF^H	SVM^L_{lnr}	SVM^H_{lnr}	SVM^L_{rbf}	SVM^H_{rbf}	LR^L	LR^H	RR^L	RR^H	*average*
AdverSPAM	1	1	.90	.90	.99	.99	1	.99	1	1	1	1	.98
FGSM	.99	1	1	1	.86	1	.96	.78	1	1	1	1	.97
DF	1	1	1	1	1	1	1	1	1	1	1	1	1
C&W	.91	.97	.44	.44	.96	.98	.97	.91	.99	.99	.99	.99	.88
Cheng	.26	.99	.59	.59	1	1	0	1	.02	1	1	1	.70
Peng	.40	.41	.23	.23	.44	.43	0	.44	.44	.44	.44	.44	.36
average	.76	.90	.69	.69	.88	.90	.66	.85	.74	.91	.91	.91	

(a) (b)

applies to almost all surrogates and targets, except for a few cases where the performance of the low- and high-complexity versions are equivalent. Among these are the *FNR*s measured against Random Forest (RF), which are quite low (in average) for all attacks.

Nevertheless, the transfer capacity of AdverSPAM over *RF* is about 80% for SVM^L_{lnr} and 90% considering all other surrogates. These values are better than all competitors except *FGSM* and *DF*. The reason is that *RF* does not use a differentiable learning function, but rather relies on a set of decision trees that establish the class of a sample based on the values assumed by individual features. Thus, if a group of features is altered significantly (i.e., exceeds the threshold value learned by the tree), the probability of misclassification is very high. This is the case of *FGSM* and *DF*, that apply to each feature a perturbation equal to ϵ_{FGSM} and proportional to η_{DF}, respectively. Then, the altered feature values differ greatly from the original ones (defined in $[0, 1]$), so producing a high transfer rate.

By observing the results of the *C&W* attack, it can be noticed that this is quite effective when the surrogate is *LR*, while it frequently fails in the case of *SVM*s. This can be explained because the decision boundaries learned from Logistic models are farther from the ω^+ samples than those computed by *SVM*s; this results in higher L_2 values and thus greater transferability of the samples. The low performances of both *Cheng* and *Peng* methods depend on the idea at their basis: in order to moderately perturb the input, they generate adversarial samples that slightly cross the decision boundary. This impacts on transferability as the examples will fail to evade the target model whenever its decision boundary is different from that of the surrogate.

Actually, *FNR*s alone are not good predictors of the attack's performance as excessively altered samples could still lead to high transfer rates. To better investigate this aspect, we measured the quality of the manipulated samples in terms of their distance from the original ones. The results shown in Figure 5 summarize the average L_2 and L_∞ values calculated for the six attacks, while varying the underlying surrogate model.

By considering the results of Figure 5 and Table 8 together, we can see that the two methods having *FNR*s comparable with *AdverSPAM* are characterized by *L2* values that are always worse than our attack (as in the case of *FGSM*), or highly variable depending on the surrogate used (*DF*). On the contrary, *AdverSPAM* exhibits fairly controlled L_2 variations for every surrogate; this is due to the intrinsic nature of our algorithm that adjusts the maximum allowed distance of the adversarial samples through the ψ parameter, which is automatically tuned on the specific surrogate.

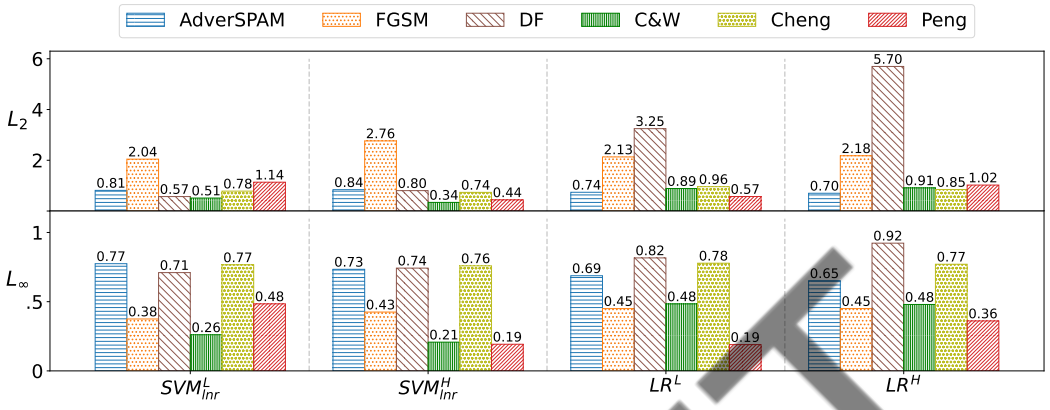


Fig. 5. The average L_2 and L_∞ distances calculated for the six attacks while varying the surrogate models.

As regards L_∞, the values measured on *DF* and *Cheng* are comparable with *AdverSPAM*, and in some cases higher (i.e., worse); conversely, the distances obtained by applying *FGSM*, *Peng*, and *C&W* attacks are generally lower. These differences depend on how each approach defines the search region for the adversarial sample. For instance, the L_∞ distances observed for *FGSM* and *DF* are very similar to the values of ϵ_{FGSM} and η_{DF}. The same considerations apply to *C&W* and *Cheng* approaches, whereas a more thorough evaluation is required for *Peng*, which exhibits the lowest L_∞ among all the baselines and is the only method explicitly accounting for feature correlation maintenance.

In order to further examine this aspect, a final comparative analysis was carried out between *AdverSPAM* and *Peng* to measure their ability to preserve the *statistical correlation* between features. As indicator, we chose the Frobenius norm:

$$||A - B||_F = \sqrt{\sum_i \sum_j |a_{i,j} - b_{i,j}|^2}, \tag{17}$$

where A and B are the Pearson correlation matrices of the features in the actual *spammers* and in the generated adversarial samples, respectively. Furthermore, in order to highlight the contribution of the *semantic dependency*, the *AdverSPAM* analysis is differentiated by whether *statistical correlation* alone, or both properties are considered. Table 9 summarizes the obtained results, where higher Frobenius values correspond to stronger changes in the correlations among features of the adversarial samples. We can observe that *AdverSPAM* exhibits a lower average correlation distance w.r.t. *Peng* (1.45 vs 3.23); this also holds for the single chosen surrogate models. Moreover, when the semantic dependencies are considered, the distortion is even lower (0.85).

In conclusion, AdverSPAM provides better average *FNR* values than the considered baselines, which translates to higher success rates on most of the tested target models. When the performances are equivalent to the competitors, the analysis of distance metrics suggests that AdverSPAM perturbs the samples less drastically, which is a desired property for any AML strategy. Finally, maintaining statistical correlation and semantic dependency helps to keep the core properties of the feature set unchanged before and after the attack.

Table 9. Comparing the correlation distances induced by *AdverSPAM* and *Peng* methods. The best values are shown in **bold**, whereas the second best values in *italics*. In the header of the table, a single **s** stands for statistical correlation preservation, whereas **ss** signals both statistical and semantic maintainment.

Surrogate \ Attack	**AdverSPAM (s)**	**AdverSPAM (ss)**	**Peng (s)**
SVM^{L}_{lnr}	*1.28*	**0.49**	3.59
SVM^{H}_{lnr}	*2.15*	**1.11**	2.57
LR^{L}	*1.27*	**0.91**	3.06
LR^{H}	*1.08*	**0.89**	3.71
average	*1.45*	**0.85**	3.23

Table 10. Example of features describing a spammer account. Original values (first row) and variations produced by AdverSPAM and the five baselines. Stronger differences are highlighted with darker colors.

Features	*Metedata*				*Content*								*Interaction*					*Community*	
	RR	AR	TSD	TISD	UUR	UMR	CHS	UR	MR	HTR	AUR	ATS	FR	MFFFR	FBR	R	CC	CBR	CBCC
Original	0	0	0	0	0	0	0	.5	1	0	0	0	.04	0	.75	.03	0	.44	.08
AdverSPAM	.04	.05	.02	0	.04	.04	.01	.44	.84	0	0	.03	0	0	.2	.02	0	.39	0
FGSM	.45	.45	.45	0	.45	.45	.45	.05	.55	0	0	.45	0	0	.3	0	0	0	0
DF	1	1	1	0	1	1	.67	0	1	0	0	1	0	0	0	0	0	0	0
C&W	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	.03	0	0	0
Cheng	.03	.04	.01	0	.03	.03	.01	.45	1	0	0	.02	0	0	.31	.02	0	.4	0
Peng	.04	.22	.15	0	.03	.28	0	.45	1	.01	0	.03	0	0	.34	0	0	.39	0

5.4 Impact of AdverSpam: a concrete case study

AdverSPAM belongs to that category of adversarial attacks that aim to perturb the input of the target model, without directly addressing the so-called *inverse feature-mapping problem* [43], i.e., how to modify the original data given the perturbed feature vector. Hence, the attacker must find on his own the set of actions to manipulate the input (the OSN account in our case) in order to obtain the feature vector suggested by AdverSPAM. This can be achieved by relying either on a manual or an automatic strategy.

The former, also known as *opportunistic* [81], consists in performing a certain action (e.g., add/remove data), measuring the resulting feature values, and reiterating until the desired values are reached. As an example, let us consider Table 10 which reports the changes made by AdverSpam and its competitors to the feature vector describing one of the spammers available in the dataset. The AdverSPAM row indicates that the most perturbed features are the *Mention Ratio* (*MR*) and the *Follower-based Reputation* (*FBR*), while the other values are quite close to the original ones. The feature *MR*, which is defined as:

$$MR(account) = \frac{number\_of\_mentions}{number\_of\_posts}, \tag{18}$$

is quite simple to alter. Since the modification suggested by AdverSPAM is to lower *MR* to 0.84, the attacker could replace tweets containing a mention with others that have the same content, but not the mention. As the number of posts do not change, the other content-based features are not influenced by this action. However, altering the tweets may change some metadata-based features that capture the tweet time, such as *TSD* and *TISD*; then, tweet replacement should follow the same posting frequency that characterized the original account. Please note that the attacker could

also use an existing account only to get $\tilde{x}$ from AdverSPAM, and then create a *new* account that is described by that perturbed feature vector.

A similar strategy can be adopted to alter the *FBR* of an account, which is defined as the average of the reputation of its followers [31]:

$$FBR(account) = \frac{\sum_{followers} reputation(follower)}{|followers|}. \tag{19}$$

Then, in order to change *FBR* from 0.75 to 0.2, the attacker could i) reduce the overall reputation of its followers (numerator), or ii) increase the number of followers (denominator). The reputation of the followers can be modified by *replacing* a subset of n followers with others having a lower reputation; this allows to keep the number of followers unaltered, which is essential to not impact on the other interaction-based features. Such a modification is quite simple to achieve by exploiting black markets, where followers having specific characteristics can be bought [61, 62]. Conversely, increasing the number of followers is somewhat more complicated as it should be done while guaranteeing that the new followers have a similar reputation to those already existing, so that the numerator of Eq. 19 does not change, nor do other related features such as *MFFFR* or *R*.

It is worth noting that attacks modifying the feature vectors in a more extensive way are difficult to manage with an opportunistic strategy, because the more values are changed the more constraints must be met while altering the account. In these cases, automatic strategies can help in finding the optimal set of actions that lead to the desired feature values. A common approach, for instance, is to model a numerical optimization algorithm which follows the negative gradient of the objective function [55]. However, it is not possible to directly apply gradient descent-based techniques when the feature space is not invertible, nor differentiable. A more general strategy consists in modeling the inverse mapping problem as a game. In [9], for instance, Monte Carlo Tree Search (MCTS) is adopted to generate a chain of mutations to be applied on malware with the goal of bypassing the target API-based classifier. In our scenario, the possible mutations could be those provided by the Twitter's API to post and remove tweets, or to control the followers and followings of an account.

5.5 Mitigation Strategies

As a final evaluation of AdverSPAM, we tested its ability to accomplish the attack when a defense mechanism is available. The remainder of this section discusses the results obtained by considering three mitigation strategies, namely, Adversarial Training, Magnet, and a confidence-based defense we designed to counter the specificities of AdverSPAM.

5.5.1 Adversarial Training.

Adversarial Training (AT) [33] is one of the most widely adopted [12, 16, 17, 52, 65, 73–75, 78] defenses against adversarial attacks because of its effectiveness and theoretical simplicity. The idea is to enable the target model to identify adversarial samples by incorporating them into the training process. The creation of *AT* samples is commonly based on the Projected Gradient Descent (PGD) attack [42], which is an iterative variant of FGSM. We applied this approach to strengthen the target models and test whether their robustness to adversarial attacks is improved.

Since the maximum perturbation amount that PGD can apply to a sample depends on a parameter ϵ_{PGD}, the assessment was repeated while varying ϵ_{PGD} in $[0.005, 0.1]$, with step 0.005. Results are shown in Figure 6a, where the different curves indicate the average FNRs achieved by AdverSPAM and the five baselines (see Section 5.3 for references) when *AT* is performed.

In terms of the reference FNRs, i.e., those measured without using any defense, the three best performing attacks are AdverSPAM, DF and FGSM (see the figure legend). After applying *AT*, FGSM is heavily penalized since, like PGD, exploits the gradient of the loss function; thus, the samples

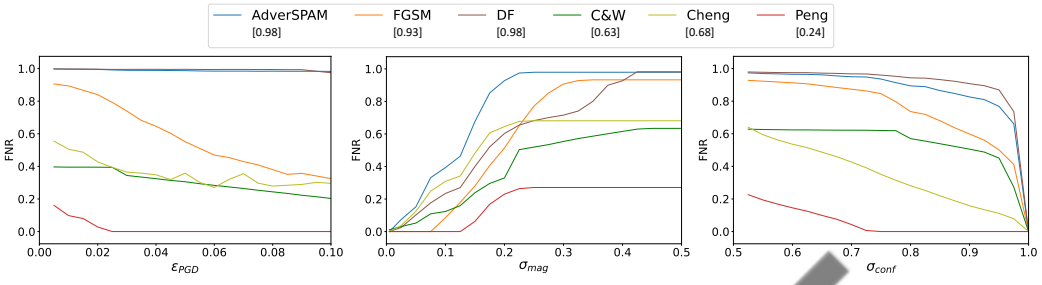


Fig. 6. Average FNRs achieved by each attack when (a) Adversarial Training, (b) Magnet, and (c) Confidence-Based defenses are used. The legend also reports the reference FNRs (indicated between square brackets) achieved by the attacks without using any mitigation technique.

generated by PGD are very similar to those created by FGSM. On the other hand, the performances of AdverSPAM and DF do not change significantly. This may be because the samples generated by PGD are very different from those created by the two attacks, and therefore the *AT*-based defense cannot exploit them to recognize the adversarial samples. As regards the other baselines, all trends are decreasing as the perturbations increase. Similarly to FGSM, this may depend on the fact that PGD samples are alike to those generated by C&W, Cheng, and Peng.

It is worth noting that, since PGD needs to access the gradient of the loss function of the model being attacked, *RF* and *RR* classifiers are not eligible for this type of attack. In order to face this limitation, we tested two other defense techniques which can be applied to all the adopted target models; the results are discussed in the following subsections.

5.5.2 *Magnet.*

Magnet [46] aims to detect adversarial samples by leveraging autoencoders, i.e., neural networks trained for learning and reproducing the exact same distribution of the data while minimizing the reconstruction error of the inputs in the training set. In particular, Magnet pre-trains an autoencoder over the distribution of the ground truth samples; then, when a new sample is proposed to the target model, this is fed into the autoencoder in order to compute its reconstruction error, which roughly represents its similarity to the data distribution observed during the training phase. If the reconstruction error exceeds a prefixed input threshold σ_{mag}, the input sample is recognized as an out-of-distribution sample and will be rejected; this prevents adversarial samples from being evaluated by the target model. We implemented Magnet according to the specifications provided in [46], and we explored $\sigma_{mag} \in [0.05, 0.5]$ with step 0.025.

Figure 6b shows that higher values of FNR are obtained as σ_{mag} increases; this means that if the threshold is set too high, the defense mechanism will not be able to reject any input and therefore its effect is null. In fact, for $\sigma_{mag} > 0.3$, all attacks reach the reference FNRs. On the other hand, choosing a value of σ_{mag} too restrictive, although it guarantees to cope with the adversarial attacks, might be impractical in a real scenario as it would result in a high number of non adversarial data rejection.

5.5.3 *Confidence-Based Defense.*

As a third defense mechanism, we focused on one of the characteristics that most differentiates AdverSPAM from other baselines, namely the fact that perturbed samples are chosen as close as possible to the decision boundary. However, some target classifiers could implement a mechanism

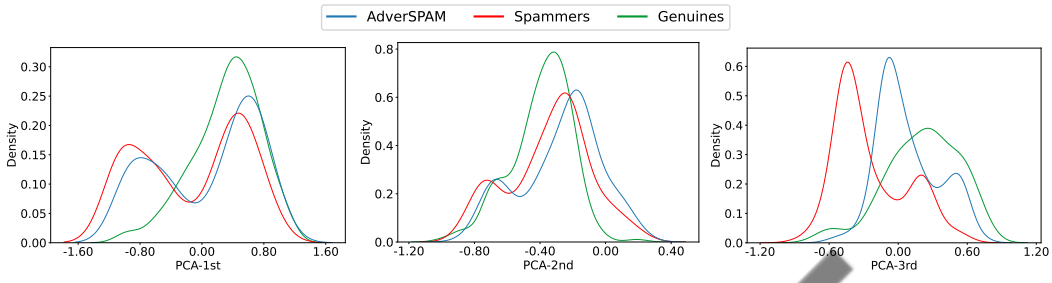


Fig. 7. Kernel Density Estimation (KDE) of the AdverSPAM samples compared to the ground genuines and spammers, computed for the first (a) second (b) and third (c) principal components describing the feature space.

to check the *confidence* of their prediction: the higher the distance of the predicted sample from the learned decision boundary, the greater the confidence in the prediction. According to these considerations, the proposed *Confidence-Based Defense* (CBD) aims to reject the "genuine" class predictions of the target model when the confidence lies below a fixed threshold σ_{conf}.

Results shown in Figure 6c indicate that as the confidence threshold increases, the effectiveness of the attacks decreases. For AdverSPAM, FGSM, DF and C&W it is possible to identify a cutoff when σ_{conf} is about 0.75. Since Cheng and Peng produce samples that lie close to the targeted decision boundary, the confidence in the prediction of such adversarial samples is relatively low, and a moderate cutoff threshold is sufficient for defending against these attacks. Finally, as in the case of Magnet, choosing high σ_{conf} values could result in high rejection rate. Therefore, good trade-off values could be those in the range $\sigma_{conf} \in [0.8, 0.9]$, where adversarial samples may be effectively filtered out while non discarding non adversarial data.

6 ADVERSPAM: ADVANTAGES AND LIMITATIONS

The experimental analysis presented so far allowed to measure the performances of AdverSPAM and compare them with some significant baselines. The results showed the effectiveness of the proposed approach in attacking different target models while applying perturbations that preserve both statistical correlation and semantic dependence between features.

Besides these results, a further aspect to consider is whether the perturbation of one or more features could impair the malicious behavior of spam accounts. In order to make it easier to visualize the properties of the feature space, Principal Component Analysis (PCA) [68] was used to reduce its dimensionality. Experiments were conducted by considering the three principal components that characterize the set of feature vectors; then, the Kernel Density Estimation (KDE) of each component was computed in order to evaluate the distributions representing both the original samples (*genuines* and *spammers*) and those generated by AdverSPAM.

By observing Figure 7, it is possible to appreciate how the densities of the genuine samples (green) strongly differ from the others. Conversely, the density estimation of the spammers (red) and AdverSPAM (blue) samples are highly similar, apart from a slight shift introduced by the adversarial perturbations. Such analysis confirms the capability of AdverSPAM in generating adversarial samples that, while preserving the key characteristics of the spam accounts, are able to evade unknown classifiers.

Although experimentally proved to be a promising adversarial attack, our approach exhibits some intrinsic limitations. The first is related to the need for choosing a surrogate model with a linear decision boundary. Such requirement is closely tied to the necessity of formulating the attack by considering linear constraints, which makes AdverSPAM particularly efficient.

Adapting AdverSPAM to different scenarios, or even different feature sets, may be computationally burdensome, as the number of constraints in our optimization problem increases according to the number of features pairs standing in a semantic dependence relationship. Moreover, operating in scenarios where there are no significant statistical correlations would tamper the effectiveness of our approach. Furthermore, if the relationships between features are non-linear, AdverSPAM would require a change in its constraints and a different statistical indicator from the Pearson's Correlation coefficient. For instance, the accuracy of the best-fitting second-order polynomial can be used in the case of quadratic correlations between data.

We studied the robustness of AdverSPAM against three different adversarial defenses, however, another interesting approach a defender may follow is the one outlined in [27], where an analysis of the sequence of consecutive requests originated from the same source is conducted. Such a countermeasure could inhibit the attack when an offline surrogate model cannot be built, and it is therefore necessary to query the online target model directly. In this extreme case, the iterative nature of the optimization solver would imply making multiple queries to the target model, which would result in a straightforward attack detection.

7 CONCLUSIONS

In this work, we presented an AML algorithm designed to deceive OSNs spammer detection systems. The strategy attack is formulated as an optimization problem whose constraints capture the essence of AML, i.e., finding the optimal perturbation to fool the target model, while preserving the inter-relationships among the features that describe the user behavior.

OSNs security in the context of AML has barely been investigated in the literature; to the best of our knowledge, this paper is probably the first addressing the spam account manipulation problem. The proposed algorithm explicitly imposes the maintenance of the statistical correlation and the semantic dependency of related features. By neglecting this aspect, the attack would produce numerically admissible perturbations, which could not actually be implemented by the attacker.

The effectiveness of *AdverSPAM* was assessed by considering Twitter as case study, although the attack can be easily applied to any other AML scenario where preserving the consistency of the feature space is mandatory.

Experimental analysis was conducted on a well-known public dataset of spammer and genuine accounts. Experiments involved the assessment of the method by testing four distinct surrogate models and twelve possible target models, as well as an overall evaluation of the performance of *AdverSPAM* as compared to five state-of-the-art attacks.

Results revealed how *AdverSPAM* is a solid competitor in terms of transferability, with *FNR* indicators that outperformed all the baselines on 10 out of 12 considered target (black-box) models. With regard to the forged adversarial features, *AdverSPAM* exhibited the lowest distortion degree introduced in data, thanks to the formulation of constraints explicitly accounting for the maintenance of features dependencies. Moreover, we tested the effectiveness of AdverSPAM against three different adversarial defenses, proving the robustness of our approach to state-of-the-art mitigations.

As part of our future work, we plan to test *AdverSPAM* effectiveness on multiple public datasets, and evaluate its ability to be more widely applicable to different scenarios characterized by similar AML requirements, such as Network Intrusion Detection. We also plan to extend the constraint on

the decision boundary of the surrogate classifier to include also non-linear classifiers, and test their efficiency as sources of adversarial samples.

REFERENCES

[1] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. 2001. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *Database Theory — ICDT 2001*, Jan Van den Bussche and Victor Vianu (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 420–434. https://doi.org/10.1007/3-540-44503-X_27

[2] Elie Alhajjar, Paul Maxwell, and Nathaniel Bastian. 2021. Adversarial machine learning in Network Intrusion Detection Systems. *Expert Systems with Applications* 186 (2021), 115782. https://doi.org/10.1016/j.eswa.2021.115782

[3] Eirini Anthi, Lowri Williams, Matilda Rhode, Pete Burnap, and Adam Wedgbury. 2021. Adversarial attacks on machine learning cybersecurity defences in Industrial Control Systems. *Journal of Information Security and Applications* 58 (2021), 102717. https://doi.org/10.1016/j.jisa.2020.102717

[4] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. *Pearson Correlation Coefficient.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–4. https://doi.org/10.1007/978-3-642-00296-0_5

[5] Sourav Bhattacharya, Dionysis Manousakas, Alberto Gil C. P. Ramos, Stylianos I. Venieris, Nicholas D. Lane, and Cecilia Mascolo. 2020. Countering Acoustic Adversarial Attacks in Microphone-Equipped Smart Home Devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 2, Article 73 (jun 2020), 24 pages. https://doi.org/10.1145/3397332

[6] Battista Biggio, Giorgio Fumera, Gian Luca Marcialis, and Fabio Roli. 2017. Statistical Meta-Analysis of Presentation Attacks for Secure Multibiometric Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 3 (2017), 561–575. https://doi.org/10.1109/TPAMI.2016.2558154

[7] Battista Biggio, Giorgio Fumera, and Fabio Roli. 2010. Multiple classifier systems for robust classifier design in adversarial environments. *International Journal of Machine Learning and Cybernetics* 1, 1 (01 Dec 2010), 27–41. https://doi.org/10.1007/s13042-010-0007-7

[8] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* 84 (2018), 317–331. https://doi.org/10.1016/j.patcog.2018.07.023

[9] John Boutsikas, Maksim Ekin Eren, Charles K. Varga, Edward Raff, Cynthia Matuszek, and Charles Nicholas. 2021. Evading Malware Classifiers via Monte Carlo Mutant Feature Discovery. In *12th Annual Malware Technical Exchange Meeting.*

[10] Cameron Buckner. 2020. Understanding adversarial examples requires a theory of artefacts for deep learning. *Nature Machine Intelligence* 2, 12 (01 Dec 2020), 731–736. https://doi.org/10.1038/s42256-020-00266-y

[11] N. Carlini and D. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP).* IEEE Computer Society, Los Alamitos, CA, USA, 39–57. https://doi.org/10.1109/SP.2017.49

[12] Guoqin Chang, Haichang Gao, Zhou Yao, and Haoquan Xiong. 2023. TextGuise: Adaptive adversarial example attacks on text classification model. *Neurocomputing* 529 (2023), 190–203. https://doi.org/10.1016/j.neucom.2023.01.071

[13] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357. https://doi.org/10.1613/jair.953

[14] Jinyin Chen, Xiang Lin, Ziqiang Shi, and Yi Liu. 2020. Link Prediction Adversarial Attack Via Iterative Gradient Attack. *IEEE Transactions on Computational Social Systems* 7, 4 (2020), 1081–1094. https://doi.org/10.1109/TCSS.2020.3004059

[15] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. 2019. Query-Efficient Hard-label Black-box Attack: An Optimization-based Approach. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019.* OpenReview.net.

[16] Yupeng Cheng, Qing Guo, Felix Juefei-Xu, Shang-Wei Lin, Wei Feng, Weisi Lin, and Yang Liu. 2022. Pasadena: Perceptually Aware and Stealthy Adversarial Denoise Attack. *IEEE Transactions on Multimedia* 24 (2022), 3807–3822. https://doi.org/10.1109/TMM.2021.3108009

[17] Alesia Chernikova and Alina Oprea. 2022. FENCE: Feasible Evasion Attacks on Neural Networks in Constrained Environments. *ACM Trans. Priv. Secur.* 25, 4, Article 34 (jul 2022), 34 pages. https://doi.org/10.1145/3544746

[18] Federico Concone, Giuseppe Lo Re, Marco Morana, and Sajal K. Das. 2022. SpADe: Multi-Stage Spam Account Detection for Online Social Networks. *IEEE Transactions on Dependable and Secure Computing* (2022), 1–16. https://doi.org/10.1109/TDSC.2022.3198830

[19] Federico Concone, Giuseppe Lo Re, Marco Morana, and Claudio Ruocco. 2019. Assisted Labeling for Spam Account Detection on Twitter. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP).* 359–366. https://doi.org/10.1109/SMARTCOMP.2019.00073

[20] Federico Concone, Giuseppe Lo Re, Marco Morana, and Claudio Ruocco. 2019. Twitter Spam Account Detection by Effective Labeling. In *Proceedings of the Third Italian Conference on Cyber Security, Pisa, Italy, February 13-15, 2019 (CEUR Workshop Proceedings, Vol. 2315)*, Pierpaolo Degano and Roberto Zunino (Eds.). CEUR-WS.org.

[21] Stefano Cresci, Marinella Petrocchi, Angelo Spognardi, and Stefano Tognazzi. 2022. Adversarial Machine Learning for Protecting Against Online Manipulation. *IEEE Internet Computing* 26, 2 (2022), 47–52. https://doi.org/10.1109/MIC.2021.3130380

[22] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2018. Social Fingerprinting: Detection of Spambot Groups Through DNA-Inspired Behavioral Modeling. *IEEE Transactions on Dependable and Secure Computing* 15, 4 (2018), 561–576. https://doi.org/10.1109/TDSC.2017.2681672

[23] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Shafi'i Muhammad Abdulhamid, Adebayo Olusola Adetunmbi, and Opeyemi Emmanuel Ajibuwa. 2019. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon* 5, 6 (2019), e01802. https://doi.org/10.1016/j.heliyon.2019.e01802

[24] Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. 2020. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Archives of Computational Methods in Engineering* 27, 4 (01 Sep 2020), 1071–1092. https://doi.org/10.1007/s11831-019-09344-w

[25] Kalyanmoy Deb. 2005. Multi-Objective Optimization. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Edmund K. Burke and Graham Kendall (Eds.). Springer US, Boston, MA, 273–316. https://doi.org/10.1007/0-387-28356-0_10

[26] Luca Demetrio, Battista Biggio, Giovanni Lagorio, Fabio Roli, and Alessandro Armando. 2021. Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware. *IEEE Transactions on Information Forensics and Security* 16 (2021), 3469–3478. https://doi.org/10.1109/TIFS.2021.3082330

[27] Luca Demetrio, Scott E. Coull, Battista Biggio, Giovanni Lagorio, Alessandro Armando, and Fabio Roli. 2021. Adversarial EXEmples: A Survey and Experimental Evaluation of Practical Attacks on Machine Learning for Windows Malware Detection. *ACM Trans. Priv. Secur.* 24, 4, Article 27 (sep 2021), 31 pages. https://doi.org/10.1145/3473039

[28] Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. 2019. Yes, Machine Learning Can Be More Secure! A Case Study on Android Malware Detection. *IEEE Transactions on Dependable and Secure Computing* 16, 4 (2019), 711–724. https://doi.org/10.1109/TDSC.2017.2700270

[29] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. 2019. Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks. In *Proceedings of the 28th USENIX Conference on Security Symposium* (Santa Clara, CA, USA) *(SEC'19)*. USENIX Association, USA, 321–338. https://doi.org/10.5555/3361338.3361361

[30] Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. 2020. Benchmarking Adversarial Robustness on Image Classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 318–328. https://doi.org/10.1109/CVPR42600.2020.00040

[31] Mohd Fazil and Muhammad Abulaish. 2018. A Hybrid Approach for Detecting Automated Spammers in Twitter. *IEEE Transactions on Information Forensics and Security* 13, 11 (2018), 2707–2719. https://doi.org/10.1109/TIFS.2018.2825958

[32] Salvatore Gaglio, Andrea Giammanco, Giuseppe Lo Re, and Marco Morana. 2022. Adversarial Machine Learning in e-Health: Attacking a Smart Prescription System. In *AIxIA 2021 – Advances in Artificial Intelligence*, Stefania Bandini, Francesca Gasparini, Viviana Mascardi, Matteo Palmonari, and Giuseppe Vizzari (Eds.). Springer International Publishing, Cham, 490–502. https://doi.org/10.1007/978-3-031-08421-8_34

[33] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples.

[34] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box Adversarial Attacks with Limited Queries and Information. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 2137–2146.

[35] Christian Janiesch, Patrick Zschech, and Kai Heinrich. 2021. Machine learning and deep learning. *Electronic Markets* 31, 3 (01 Sep 2021), 685–695. https://doi.org/10.1007/s12525-021-00475-2

[36] Arindam Jati, Chin-Cheng Hsu, Monisankha Pal, Raghuveer Peri, Wael AbdAlmageed, and Shrikanth Narayanan. 2021. Adversarial attack and defense strategies for deep speaker recognition systems. *Computer Speech & Language* 68 (2021), 101199. https://doi.org/10.1016/j.csl.2021.101199

[37] Bhargav Kuchipudi, Ravi Teja Nannapaneni, and Qi Liao. 2020. Adversarial Machine Learning for Spam Filters. In *Proceedings of the 15th International Conference on Availability, Reliability and Security* (Virtual Event, Ireland) *(ARES '20)*. Association for Computing Machinery, New York, NY, USA, Article 38, 6 pages. https://doi.org/10.1145/3407023.3407079

[38] Rahul Kumar, Ridhi Arora, Vipul Bansal, Vinodh J. Sahayasheela, Himanshu Buckchash, Javed Imran, Narayanan Narayanan, Ganesh N. Pandian, and Balasubramanian Raman. 2022. Classification of COVID-19 from chest x-ray images using deep features and correlation coefficient. *Multimedia Tools and Applications* 81, 19 (Aug. 2022), 27631–27655. https://doi.org/10.1007/s11042-022-12500-3

[39] Wassila Lalouani, Mohamed Younis, and Uthman Baroudi. 2021. Countering radiometric signature exploitation using adversarial machine learning based protocol switching. *Computer Communications* 174 (2021), 109–121. https://doi.org/10.1016/j.comcom.2021.04.007

[40] Guanzhi Li, Aining Zhang, Qizhi Zhang, Di Wu, and Choujun Zhan. 2022. Pearson Correlation Coefficient-Based Performance Enhancement of Broad Learning System for Stock Price Prediction. *IEEE Transactions on Circuits and Systems II: Express Briefs* 69, 5 (2022), 2413–2417. https://doi.org/10.1109/TCSII.2022.3160266

[41] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into Transferable Adversarial Examples and Black-box Attacks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[42] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

[43] Davide Maiorca, Battista Biggio, and Giorgio Giacinto. 2019. Towards Adversarial Malware Detection: Lessons Learned from PDF-Based Attacks. *ACM Comput. Surv.* 52, 4, Article 78 (aug 2019), 36 pages. https://doi.org/10.1145/3332184

[44] Davide Maiorca, Igino Corona, and Giorgio Giacinto. 2013. Looking at the Bag is Not Enough to Find the Bomb: An Evasion of Structural Methods for Malicious PDF Files Detection. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security* (Hangzhou, China) *(ASIA CCS '13)*. Association for Computing Machinery, New York, NY, USA, 119–130. https://doi.org/10.1145/2484313.2484327

[45] Davide Maiorca, Ambra Demontis, Battista Biggio, Fabio Roli, and Giorgio Giacinto. 2020. Adversarial Detection of Flash Malware: Limitations and Open Issues. *Computers & Security* 96 (2020), 101901. https://doi.org/10.1016/j.cose.2020.101901

[46] Dongyu Meng and Hao Chen. 2017. MagNet: A Two-Pronged Defense against Adversarial Examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 135–147. https://doi.org/10.1145/3133956.3134057

[47] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 2574–2582. https://doi.org/10.1109/CVPR.2016.282

[48] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal Adversarial Perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 86–94. https://doi.org/10.1109/CVPR.2017.17

[49] Akm Iqtidar Newaz, Nur Imtiazul Haque, Amit Kumar Sikder, Mohammad Ashiqur Rahman, and A. Selcuk Uluagac. 2020. Adversarial Attacks to Machine Learning-Based Smart Healthcare Systems. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 1–6. https://doi.org/10.1109/GLOBECOM42002.2020.9322472

[50] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (Abu Dhabi, United Arab Emirates) *(ASIA CCS '17)*. Association for Computing Machinery, New York, NY, USA, 506–519. https://doi.org/10.1145/3052973.3053009

[51] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. 2016. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv preprint arXiv:1605.07277* (2016).

[52] Bowen Peng, Bo Peng, Jie Zhou, Jianyue Xie, and Li Liu. 2022. Scattering Model Guided Adversarial Examples for SAR Target Recognition: Attack and Defense. *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), 1–17. https://doi.org/10.1109/TGRS.2022.3213305

[53] Xiao Peng, Weiqing Huang, and Zhixin Shi. 2019. Adversarial Attack Against DoS Intrusion Detection: An Improved Boundary-Based Method. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. 1288–1295. https://doi.org/10.1109/ICTAI.2019.00179

[54] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. 2020. Intriguing Properties of Adversarial ML Attacks in the Problem Space. In *2020 IEEE Symposium on Security and Privacy (SP)*. 1332–1349. https://doi.org/10.1109/SP40000.2020.00073

[55] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. 2020. Intriguing Properties of Adversarial ML Attacks in the Problem Space. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 1308–1325. https://doi.org/10.1109/SP40000.2020.00073

[56] M. J. D. Powell. 1994. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*. Springer Netherlands, Dordrecht, 51–67. https://doi.org/10.1007/978-94-015-8330-5_4

[57] Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel. 2019. Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 5231–5240.

[58] Jonas Rauber, Roland Zimmermann, Matthias Bethge, and Wieland Brendel. 2020. Foolbox Native: Fast adversarial attacks to benchmark the robustness of machine learning models in PyTorch, TensorFlow, and JAX. *Journal of Open Source Software* 5, 53 (2020), 2607. https://doi.org/10.21105/joss.02607

[59] Paolo Russu, Ambra Demontis, Battista Biggio, Giorgio Fumera, and Fabio Roli. 2016. Secure Kernel Machines against Evasion Attacks. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security* (Vienna, Austria) *(AISec '16)*. Association for Computing Machinery, New York, NY, USA, 59–69. https://doi.org/10.1145/2996758.2996771

[60] Yi Shi, Kemal Davaslioglu, and Yalin E. Sagduyu. 2021. Generative Adversarial Network in the Air: Deep Adversarial Learning for Wireless Signal Spoofing. *IEEE Transactions on Cognitive Communications and Networking* 7, 1 (2021), 294–303. https://doi.org/10.1109/TCCN.2020.3010330

[61] Gianluca Stringhini, Manuel Egele, Christopher Kruegel, and Giovanni Vigna. 2012. Poultry Markets: On the Underground Economy of Twitter Followers. *SIGCOMM Comput. Commun. Rev.* 42, 4 (sep 2012), 527–532. https://doi.org/10.1145/2377677.2377781

[62] Gianluca Stringhini, Gang Wang, Manuel Egele, Christopher Kruegel, Giovanni Vigna, Haitao Zheng, and Ben Y. Zhao. 2013. Follow the Green: Growth and Dynamics in Twitter Follower Markets. In *Proceedings of the 2013 Conference on Internet Measurement Conference* (Barcelona, Spain) *(IMC '13)*. Association for Computing Machinery, New York, NY, USA, 163–176. https://doi.org/10.1145/2504730.2504731

[63] Octavian Suciu, Radu Mărginean, Yiğitcan Kaya, Hal Daumé, and Tudor Dumitraş. 2018. When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks. In *Proceedings of the 27th USENIX Conference on Security Symposium* (Baltimore, MD, USA) *(SEC'18)*. USENIX Association, USA, 1299–1316.

[64] Jiaxi Tang, Hongyi Wen, and Ke Wang. 2020. Revisiting Adversarially Learned Injection Attacks Against Recommender Systems. In *Proceedings of the 14th ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 318–327. https://doi.org/10.1145/3383313.3412243

[65] Sanli Tang, Xiaolin Huang, Mingjian Chen, Chengjin Sun, and Jie Yang. 2021. Adversarial Attack Type I: Cheat Classifiers by Significant Changes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 3 (2021), 1100–1109. https://doi.org/10.1109/TPAMI.2019.2936378

[66] Qiuhua Wang, Hui Yang, Guohua Wu, Kim-Kwang Raymond Choo, Zheng Zhang, Gongxun Miao, and Yizhi Ren. 2022. Black-box adversarial attacks on XSS attack detection model. *Computers & Security* 113 (2022), 102554. https://doi.org/10.1016/j.cose.2021.102554

[67] Qian Wang, Baolin Zheng, Qi Li, Chao Shen, and Zhongjie Ba. 2021. Towards Query-Efficient Adversarial Attacks Against Automatic Speech Recognition Systems. *IEEE Transactions on Information Forensics and Security* 16 (2021), 896–908. https://doi.org/10.1109/TIFS.2020.3026543

[68] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems* 2, 1 (1987), 37–52. https://doi.org/10.1016/0169-7439(87)80084-9 Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[69] Jie Wu, Na Li, Yan Zhao, and Jujie Wang. 2022. Usage of correlation analysis and hypothesis test in optimizing the gated recurrent unit network for wind speed forecasting. *Energy* 242 (2022), 122960. https://doi.org/10.1016/j.energy.2021.122960

[70] Tingmin Wu, Sheng Wen, Yang Xiang, and Wanlei Zhou. 2018. Twitter spam detection: survey of new approaches and comparative study. *Computers and Security* 76 (July 2018), 265–284. https://doi.org/10.1016/j.cose.2017.11.013

[71] Mingfu Xue, Chengxiang Yuan, Can He, Jian Wang, and Weiqiang Liu. 2021. NaturalAE: Natural and robust physical adversarial examples for object detectors. *Journal of Information Security and Applications* 57 (2021), 102694. https://doi.org/10.1016/j.jisa.2020.102694

[72] Chao Yang, Robert Harkreader, and Guofei Gu. 2013. Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers. *IEEE Transactions on Information Forensics and Security* 8, 8 (2013), 1280–1293. https://doi.org/10.1109/TIFS.2013.2267732

[73] Zhenqin Yin, Yue Zhuo, and Zhiqiang Ge. 2023. Transfer adversarial attacks across industrial intelligent systems. *Reliability Engineering & System Safety* 237 (2023), 109299. https://doi.org/10.1016/j.ress.2023.109299

[74] Dazhi Zhan, Yexin Duan, Yue Hu, Lujia Yin, Zhisong Pan, and Shize Guo. 2023. AMGmal: Adaptive mask-guided adversarial attack against malware detection with minimal perturbation. *Computers & Security* 127 (2023), 103103. https://doi.org/10.1016/j.cose.2023.103103

[75] Chunkai Zhang, Xiaofeng Luo, and Peiyi Han. 2022. On-manifold adversarial attack based on latent space substitute model. *Computers & Security* 120 (2022), 102770. https://doi.org/10.1016/j.cose.2022.102770

[76] Fei Zhang, Patrick P. K. Chan, Battista Biggio, Daniel S. Yeung, and Fabio Roli. 2016. Adversarial Feature Selection Against Evasion Attacks. *IEEE Transactions on Cybernetics* 46, 3 (2016), 766–777. https://doi.org/10.1109/TCYB.2015.2415032

[77] Jiale Zhang, Bing Chen, Xiang Cheng, Huynh Thi Thanh Binh, and Shui Yu. 2021. PoisonGAN: Generative Poisoning Attacks Against Federated Learning in Edge Computing Systems. *IEEE Internet of Things Journal* 8, 5 (2021), 3310–3322. https://doi.org/10.1109/JIOT.2020.3023126

[78] Xinyu Zhang, Yang Xu, Sicong Zhang, and Xiaojian Li. 2022. A Highly Stealthy Adaptive Decay Attack Against Speaker Recognition. *IEEE Access* 10 (2022), 118789–118805. https://doi.org/10.1109/ACCESS.2022.3220639

[79] Ping Zhao, Haojun Huang, Xiaohui Zhao, and Daiyu Huang. 2020. P3: Privacy-Preserving Scheme Against Poisoning Attacks in Mobile-Edge Computing. *IEEE Transactions on Computational Social Systems* 7, 3 (2020), 818–826. https://doi.org/10.1109/TCSS.2019.2960824

[80] Daniel Zügner, Oliver Borchert, Amir Akbarnejad, and Stephan Günnemann. 2020. Adversarial Attacks on Graph Neural Networks: Perturbations and Their Patterns. *ACM Trans. Knowl. Discov. Data* 14, 5, Article 57 (jun 2020), 31 pages. https://doi.org/10.1145/3394520

[81] Nedim Šrndić and Pavel Laskov. 2014. Practical Evasion of a Learning-Based Classifier: A Case Study. In *2014 IEEE Symposium on Security and Privacy*. 197–211. https://doi.org/10.1109/SP.2014.20