Adversarial Machine Learning in Malware Detection: Arms Race between Evasion Attack and Defense

Lingwei Chen, Yanfang Ye*, Thirimachos Bourlai
Department of Computer Science and Electrical Engineering
West Virginia University, Morgantown, WV 26506, USA
lgchen@mix.wvu.edu, yanfang.ye@mail.wvu.edu, thirimachos.bourlai@mail.wvu.edu

***Abstract*—Since malware has caused serious damages and evolving threats to computer and Internet users, its detection is of great interest to both anti-malware industry and researchers. In recent years, machine learning-based systems have been successfully deployed in malware detection, in which different kinds of classifiers are built based on the training samples using different feature representations. Unfortunately, as classifiers become more widely deployed, the incentive for defeating them increases. In this paper, we explore the adversarial machine learning in malware detection. In particular, on the basis of a learning-based classifier with the input of Windows Application Programming Interface (API) calls extracted from the Portable Executable (PE) files, we present an effective evasion attack model (named *EvnAttack*) by considering different contributions of the features to the classification problem. To be resilient against the evasion attack, we further propose a secure-learning paradigm for malware detection (named *SecDefender*), which not only adopts classifier retraining technique but also introduces the security regularization term which considers the evasion cost of feature manipulations by attackers to enhance the system security. Comprehensive experimental results on the real sample collections from Comodo Cloud Security Center demonstrate the effectiveness of our proposed methods.**

I. Introduction

Malware (e.g., viruses, worms, trojans, ransomware) is ***mal***icious soft***ware*** that is disseminated by attackers to launch a wide range of security attacks, such as stealing user's private information, hijacking devices remotely to deliver massive spam emails, and infiltrating user' online account credentials. Malware has caused serious damages and significant financial loss to many computer and Internet users. A recent study [17] showed that nearly half of Internet users encountered malicious software, in which 80% malware attacks caused problems for users and almost 30% resulted in money loss. In order to protect the legitimate users against the evolving threats malware poses, its detection is of utmost concern to both anti-malware industry and researchers. Recently, systems using machine learning techniques have been increasingly applied and successfully deployed in malware detection [34], [20], [1], [23], [33]. In these systems, based on different feature representations, various kinds of classification methods are used for model construction to detect malware. The effectiveness of machine learning techniques relies on the assumption that training data and testing data follow the same distribution. However, this hypothesis is likely to be violated by an adversary who may carefully manipulate the input data to exploit specific vulnerabilities of a classifier and thus to compromise its security. In other words, machine learning itself may open the possibility for an adversary who maliciously "mis-trains" a classifier in a malware detection system by simply changing the data distribution or feature importance [3], [4]. When the machine learning system is deployed in a real-world environment, it is of a great interest for malware attackers to actively manipulate the data to make the classifier produce errors (i.e., minimum true positive or minimum true negative).

If we look at the evolution of malware detection techniques [10], [16], [27], [9], [35], [11], attackers and defenders are actually engaged in a never-ending arms race where both of them continually come up with their own optimal strategies of variability and sophistication to overcome the opponents [7], [13]. For example, when defenders widely used signature-based methods for malware detection, attackers adopted code obfuscation or polymorphism to evade the detection and defeat attempts to analyze their inner mechanisms [18]. Currently, the issues of understanding machine learning security in adversarial settings [36] are starting to be leveraged, from either adversarial [22], [13], [28], [7], [5] or defensive [36], [29], [14], [4], [6] perspectives. However, most existing researches for adversarial machine learning rarely conduct practical investigations into malware detection domain. As machine learning based detections become more widely deployed, the adversary incentive for defeating them increases.

In this paper, we go further insight into the arms race between evasion attack and defense and explore the adversarial machine learning in malware detection. In particular, on the basis of a learning-based classifier with the input of Windows API calls extracted from the PE files, we first present an effective evasion attack model (named *EvnAttack*) to thoroughly assess the security of the classifier by considering different contributions of the API calls to the classification problem. To effectively counter such kind of evasion attack, we further propose a resilient yet elegant secure-learning paradigm for malware detection (named *SecDefender*). The major contributions of our work can be summarized as follows:

- *An effective evasion attack model on machine learning-based classifier:* In a machine learning-based detection system, features can differently contribute to the classification problem, which can be automatically weighed by

*Corresponding author.

978-1-5386-2385-5/17 $31.00 © 2017 IEEE
DOI 10.1109/EISIC.2017.21

the learning algorithm based on the training data (either benign or malicious). Considering attackers' different skills and capabilities, we first thoroughly investigate the property of the feature set observed from the real sample collection and their different contributions; then we present an effective evasion attack model (named *EvnAttack*) to assess the security of the classifier.

- *A secure-learning paradigm against evasion attack in malware detection:* To improve the system security and be resilient against the evasion attack, we propose a secure-learning paradigm in malware detection (named *SecDefender*). In the secure-learning model, we not only adopt classifier retraining technique, but also introduce the security regularization term based on the evasion cost of feature manipulations by attackers to enhance the security of the classifier.
- *Comprehensive experimental study on a real sample collection from an anti-malware industry company:* We obtain the dataset from Comodo Cloud Security Center, containing $10,000$ file samples (half benign and half malicious). We then build a practical solution for evasion attack and its detection based on this real sample collection. Comprehensive experimental results demonstrate the effectiveness of our proposed methods.

The rest of the paper is organized as follows. Section II discusses the related work. Section III defines the problem of machine learning-based malware detection. Section IV describes the implementation of an effective evasion attack model. Section V introduces a secure-learning model against the evasion attack in malware detection. In Section VI, we systematically evaluate the effectiveness of the proposed methods. Finally, Section VII concludes.

II. Related Work

Machine learning techniques offer unparalleled flexibility in automatic malware detection [34], [1], [23], [33], [20], [15], [26]. However, machine learning itself can be a target of attack by a malicious adversary [22], [28], [7], [5], [29], [14], [4], [6]. In some cybersecurity domains, there are ample evidences that show adversaries can actively manipulate the data to evade the detection [13], [36], [22], [8], [7]. For example, in the domain of spam email detection, Dalvi et al. [13] examined the cost for measuring each feature of the dataset using Naïve Bayes classifier, and proposed an optimal strategy for the adversary to play against the classifier. Zhang et al. [36] took gradient steps to find the closest evasion point $\mathbf{x}'$ to the malicious sample $\mathbf{x}$. The Adversarial Classifier Reverse Engineering (ACRE) framework [22] was introduced to study how an adversary can learn sufficient information to construct adversarial attacks using minimal adversarial cost. Brückner et al. [8] presented the interaction between the learner and the data generator as a static game, and explored the adversarial conditions and properties to find the equilibrial prediction model in the context of spam email filtering. To combat the evasion attacks, increasing research efforts have been devoted to the security of machine learning [29], [36], [14], [12]. Wu et al. [30] and Wang et al. [29] manipulated the training data distribution so that its distribution could be matched to the test data. Kolcz et al. [19] applied averaging method resting on random subsets of reweighted features to produce a linear ensemble classifier. Debarr et al. [14] explored randomization to generalize learning model by randomly choosing dataset or features, and estimated some parameters that fit data best. However, the application of adversarial machine learning into malware detection domain has been scarce with the exception that Šrndic et al. [28] and Xu et al. [31] both exploited PDF malware as a case study to evaluate the security of learning-based classifiers (e.g., PDFrate and Hidost).

Different from the existing works, in this paper, we explore the security of machine learning in malware detection by developing the theory and approaches to learn the behavioral models for both attackers and defenders, with respect to an effective evasion attack model and a secure-learning paradigm. The proposed methods can be readily applied in other malware detection tasks.

III. Machine Learning-based Malware Detection

A malware detection system using machine learning techniques attempts to identify those "zero-day" malware or variants of known malware through building a classification model based on the labeled training sample set and predefined feature representations. In this section, we introduce a learning-based classifier based on a feature representation approach for malware detection.

A. Feature Representation

PE is designed as a common file format for all flavors of Windows operating system, and malicious PE files are in the majority of the malware in recent years [34]. Since Windows API calls can effectively reflect the behaviors of PE program codes [34] (e.g., the API "*GetFileType*" in "*KERNEL32.DLL*" can be used to retrieve the file type of the specified file, while the API "*GetDlgItemText*" in "*USER32.DLL*" can be utilized to obtain the title or text associated with a control in a dialog box), we extract Windows API calls from the PE files to represent them. If a PE file is previously compressed by a third party binary compress tool such as UPX and ASPack Shell or embedded a homemade packer, it will be decompressed at first using CMDsm developed by Comodo Anti-malware Lab before feature extraction. Note that API calls are exploited here as a case study, while other feature representations, such as binary n-gram and dynamic system calls, are also applicable in our investigation. We further convert the features into a vector space so that it can be fed into the classifier for training or testing.

Based on the extracted features, we denote our dataset D to be of the form $D = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ of n file samples, where $\mathbf{x}_i$ is the set of Windows API calls extracted from file i, and y_i is the class label of file i, where $y_i \in \{+1, -1, 0\}$ ($+1$ denotes malicious, -1 denotes benign, and 0 denotes unknown). Let d be the number of all extracted Windows API calls in the

dataset D. Each of the PE file can be represented by a binary feature vector:

$$\mathbf{x}_i =< x_{i1}, x_{i2}, x_{i3}, ..., x_{id} >, \quad (1)$$

where $\mathbf{x}_i \in \mathbb{R}^d$, and $x_{ij} = \{0, 1\}$ (i.e., if file i includes API_j, then $x_{ij} = 1$; otherwise, $x_{ij} = 0$). Table I shows a sample data set.

TABLE I
AN EXAMPLE DATASET IN MALWARE DETECTION

File	Extracted API Calls	Feature Vector	Label
1	API_3, API_5	$< 0, 0, 1, 0, 1 >$	$+1$
2	API_1, API_2, API_4	$< 1, 1, 0, 1, 0 >$	-1
3	API_2, API_3, API_5	$< 0, 1, 1, 0, 1 >$	0

B. Learning-based Classifier for Malware Detection

The malware detection problem can be stated in the form of: $f : \mathcal{X} \to \mathcal{Y}$ which assigns a label $y \in \mathcal{Y}$ (i.e., -1 or $+1$) to an input file $\mathbf{x} \in \mathcal{X}$ through the learning function f. A general linear classification model for malware detection can be thus denoted as:

$$\mathbf{f} = \text{sign}(f(\mathbf{X})) = \text{sign}(\mathbf{X}^T\mathbf{w} + \mathbf{b}), \quad (2)$$

where $\mathbf{f}$ is a vector, each of whose elements is the label (i.e., malicious or benign) of a file sample to be predicted, each column of matrix $\mathbf{X}$ is the feature vector of a PE file, $\mathbf{w}$ is the weight vector and $\mathbf{b}$ is the biases. More specifically, the linear-based classifier can be formalized as an optimization problem [35]:

$$\underset{\mathbf{f},\mathbf{w},\mathbf{b},\boldsymbol{\xi}}{\text{argmin}} \frac{1}{2}||\mathbf{y}-\mathbf{f}||^2 + \frac{1}{2\beta}\mathbf{w}^T\mathbf{w} + \frac{1}{2\gamma}\mathbf{b}^T\mathbf{b} + \boldsymbol{\xi}^T(\mathbf{f}-\mathbf{X}^T\mathbf{w}-\mathbf{b}) \quad (3)$$

where $\mathbf{y}$ is the labeled information vector, and $\boldsymbol{\xi}$ is Lagrange multiplier which is a strategy for finding the local minima of $\frac{1}{2}||\mathbf{y} - \mathbf{f}||^2$ subject to $\mathbf{f} - \mathbf{X}^T\mathbf{w} - \mathbf{h} = 0$, β and γ are the regularization parameters. Note that Equation 3 is a linear classifier (denoted as *OrgDefender* throughout the paper) that consists of specific loss function and regularization terms. Without loss of generality, the equation can be transformed into different linear models depending on the choices of loss function and regularization terms.

IV. EVASION ATTACK MODEL

In malware detection, there are two types of security violation in the adversarial settings [3], [4]: (1) *Integrity attack* which allows malware being classified as benign; (2) *Availability attack* that creates a denial of service in which benign files are disable to be normally executed. In this paper, we focus on the former attack since it's the most prevalent type of attacks that may be encountered in adversarial settings during system operation. The integrity attack is also called evasion attack, in which malicious samples are modified at test time to evade detection. Given an original malicious file $\mathbf{x} \in \mathcal{X}^+$, the evasion attack attempts to manipulate it to be detected as negative (i.e., $\mathbf{x}' \in \mathcal{X}^-$) with certain evasion cost. In this section, we present how attackers can achieve such attack.

A. Evasion Cost

Considering each file is represented by a binary feature vector (as stated in Section III-A), a typical feature manipulation can be the addition or elimination of each binary. Then, the evasion cost for an attacker to perform an attack can be determined by the number of binaries that are changed from $\mathbf{x}$ to $\mathbf{x}'$, which can be defined as [22]:

$$\mathcal{C}(\mathbf{x}', \mathbf{x}) = \sum_{i=1}^{d} c_i |x'_i - x_i|. \quad (4)$$

where c_i denotes the corresponding cost of changing a feature. The manipulation cost c_i for each feature is different. For example, some specific Windows API calls may affect the structure for intrusive functionality, which are more expensive to be modified (i.e., c_i for changing those features should be higher). In the real operation, it's impractical for an attacker to modify a malware into benign at any cost. For instance, it's infeasible to manipulate a large number of features to evade the detection while its malicious functionalities still being reserved. Accordingly, there is an upper limit of the maximum manipulations that can be made to the original malware $\mathbf{x}$. That is, the manipulation function $\mathcal{A}(\mathbf{x})$ can be formulated as

$$\mathcal{A}(\mathbf{x}) = \begin{cases} \mathbf{x}' & f(\mathbf{x}') < 0 \; and \; \mathcal{C}(\mathbf{x}', \mathbf{x}) \leq \delta_{\max} \\ \mathbf{x} & \text{otherwise} \end{cases}, \quad (5)$$

where the malware is manipulated to be misclassified as negative only if the evasion cost is less than a maximum manipulations $\delta_{\max}$.

Let $\mathbf{f}' = \text{sign}(f(\mathcal{A}(\mathbf{X})))$, then the main idea for an evasion attack is to maximize the total loss of classification (i.e., $\text{argmax } \frac{1}{2}||\mathbf{y}-\mathbf{f}'||^2$), which means that the more malware are misclassified as benign files, the more effective the evasion attack could be. An ideal evasion attack modifies a small but optimal portion of features of the malware with minimal evasion cost, which thus makes the classifier achieve lowest true positive rate. To achieve such adversary goal, it's the most important for the attackers to choose a relevant subset of features applied for addition and elimination. Therefore, to well implement the attack, we take deep insight into the property of the feature set.

B. Property of the Feature Set

As different features (i.e., API calls in our application) differently contribute to the classification of malware and benign files, it's worth to investigate the importance of each feature. We analyze the sample set obtained from Comodo Cloud Security Center, which contains 10,000 labeled files with $3,503$ extracted API calls. Given x representing an API call, and class label y, we use Max-Relevance algorithm ($I(x, y)$) [24] to weigh the importance (i.e., contribution) of each API call in classifying malware and benign files; that is, we calculate the relevance score of each API call for malware and benign file classification respectively. Those API calls with extremely low relevance scores (e.g., about 85% of the extracted API calls with relevance scores lower

than 0.0005) have limited or no contributions in malware detection, thus they won't be considered for the further design of the evasion attack. Table II shows the top ranked API calls related to malware and benign files respectively based on our sample collection, from which we can observe that for those API calls with high relevance scores, some are explicitly relevant to malware, while some have high influence on the classification of benign files. With further analysis, we find that the most important activity in malware is file management [2], which enables them to create, or copy files (themselves or other files) multiple times to spread malware distribution, control the targeted computers, and destroy the integrity of the system (e.g., *CreateFileW* in *KERNEL32.DLL*, *DestroyIcon* in *USER32.DLL*, *CreatePopupMenu* in *USER32.DLL*, *DestroyWindow* in *USER32.DLL*, etc.). To achieve the malicious goals, they also have their own methods to deal with process and registry, which heavily use *VirtualQuery* in *KERNEL32.DLL* to get the virtual address space of the calling process that is intent to hide from or affect. Compared with malware, benign files act normally in file, memory, process, and registry operations.

TABLE II
LIST OF THE TOP RANKED API CALLS

ID	API Contributing to Malware Classification	Rel. Score
178	KERNEL32.DLL,VirtualQuery;	0.0568
124	KERNEL32.DLL,ExitProcess;	0.0459
615	KERNEL32.DLL,CreateFileW;	0.0406
607	KERNEL32.DLL,CompareStringA;	0.0381
8	USER32.DLL,RegisterClassA;	0.0355
1637	USER32.DLL,DestroyIcon;	0.0318
1606	USER32.DLL,TrackPopupMenu;	0.0317
207	KERNEL32.DLL,IsBadCodePtr;	0.0235
1601	USER32.DLL,CreatePopupMenu;	0.0213
235	USER32.DLL,DestroyWindow;	0.0205
ID	API Contributing to Benign File Classification	Rel. Score
80	KERNEL32.DLL,FreeLibrary;	0.1035
57	ADVAPI32.DLL,RegCloseKey;	0.0972
578	ADVAPI32.DLL,RegOpenKeyExW;	0.0964
20	KERNEL32.DLL,lstrlenW;	0.0846
111	KERNEL32.DLL,GetCurrentThreadId;	0.0825
22	KERNEL32.DLL,Sleep;	0.0756
37	KERNEL32.DLL,LocalFree;	0.0756
102	KERNEL32.DLL,GetTickCount;	0.0673
36	KERNEL32.DLL,GetLastError;	0.0538
506	KERNEL32.DLL,SetEvent;	0.0532

Based on the above observed property from the real sample collection, intuitively, to evade the detection with lower evasion cost, attackers may manipulate the API calls by the way of injecting the ones most relevant to benign files while removing the ones with higher relevance scores to malware. To simulate the evasion attack, we rank each API call and group them into two sets: $\mathcal{M}$ (those highly relevant to malware) and $\mathcal{B}$ (those highly relevant to benign files) in the descent order of $I(x,+1)$ and $I(x,-1)$ respectively, where $\mathcal{M}$ is utilized for elimination, while $\mathcal{B}$ is applied for addition.

C. Evasion Attack Model

To implement the evasion attack, we further define a function $g(\mathcal{A}(\mathbf{X}))$ to represent the capability of an attacker:

$$g(\mathcal{A}(\mathbf{X})) = ||\mathbf{y} - \mathbf{f}'||^2, \tag{6}$$

$g(\mathcal{A}(\mathbf{X}))$ implies the number of malware misclassified as benign files. The underlying idea is thus to manipulate a subset of features with minimum evasion cost while maximize the total loss of classification (as specified in Equation 6). In principle, a brute-force method can be applied to select features for manipulation. However, search by exhaustion is extremely expensive for the large-dimensional feature set. To achieve the optimal attack, here we adopt the wrapper method [36] which greedily selects features based on the capability of the attack. Different from the work in [36], we conduct bi-directional feature selection, that is, forward feature addition performed on $\mathcal{B}$ and backward feature elimination performed on $\mathcal{M}$. At each iteration, an API call will be selected for addition or elimination depending on the fact how it influences the value of $g(\mathcal{A}(\mathbf{X}))$. The evasion attack $\boldsymbol{\theta} = \{\boldsymbol{\theta}^+, \boldsymbol{\theta}^-\}$ will be drawn from the iterations, where $\boldsymbol{\theta}^+ \in \{0,1\}^d$ (if API_i is selected for elimination, then $\theta_i^+ = 1$; otherwise, $\theta_i^+ = 0$), and $\boldsymbol{\theta}^- \in \{0,1\}^d$ (if API_i is selected for addition, then $\theta_i^- = 1$; otherwise, $\theta_i^- = 0$). The iterations will terminate at the point where the evasion cost reaches to maximum ($\delta_{\max}$) or the features available for addition and elimination are all manipulated. The implementation of the proposed evasion attack (*EvnAttack*) is given in Algorithm 1.

Algorithm 1: ***EvnAttack*** - An effective evasion attack model for the attackers with different skills and capabilities

Input: Training set $D = \{\mathbf{x}_i, y_i\}_{i=1}^n$, testing set $D_t = \{\mathbf{x}_i, y_i\}_{i=1}^{n_t}$; a_f: cost of changing feature f; c^+, c^-: cost of eliminating and adding features in a file; $\mathcal{S}^+$, $\mathcal{S}^-$: features selected; μ^+, μ^-: number of features selected for elimination and addition.
Output: Evasion attack $\boldsymbol{\theta} = \{\boldsymbol{\theta}^+, \boldsymbol{\theta}^-\}$.
Train a classifier $f(\mathbf{X})$ using n training file samples;
$\mathcal{S}^+ \leftarrow \emptyset$, $\mathcal{S}^- \leftarrow \emptyset, \theta_i^+ = 0, \theta_j^- = 0, (i,j \in (0,1,...,d))$;
while *($c^+ + c^- \leq \delta_{\max}$) and ($\mu^+ < d$ or $\mu^- < d$)* **do**
  $\mathbf{X} \leftarrow \mathbf{X}/\mathcal{S}^+$, $\mathbf{X} \leftarrow \mathbf{X} \bigcup \mathcal{S}^-$;
  for *each feature* $x_i^+ \in \mathcal{M}$ **do**
    $\mathbf{X}_{x^+} \leftarrow \mathbf{X}/\{x_i^+\}$: eliminate x_i^+ from n_t testing file samples; Calculate $g(\mathbf{X}_{x^+})$;
  end
  for *each feature* $x_j^- \in \mathcal{B}$ **do**
    $\mathbf{X}_{x^-} \leftarrow \mathbf{X} \bigcup \{x_j^-\}$: add x_j^- to n_t testing file samples; Calculate $g(\mathbf{X}_{x^-})$;
  end
  $x_{max} = \text{argmax}\ \{g(\mathbf{X}_{x^+}), g(\mathbf{X}_{x^-})\}$;
  if $x_{max} \in \mathcal{M}$ *and* $\mu^+ < d$ **then**
    $\mathcal{S}^+ \leftarrow \mathcal{S}^+ \bigcup \{x_i^+\}$, $c^+ = c^+ + a_{x_i^+}$, $\mu^+ = \mu^+ + 1$;
  end
  if $x_{max} \in \mathcal{B}$ *and* $\mu^- < d$ **then**
    $\mathcal{S}^- \leftarrow \mathcal{S}^- \bigcup \{x_j^-\}$,$c^- = c^- + a_{x_j^-}$,$\mu^- = \mu^- + 1$;
  end
end
Set $\theta_i^+ = 1$ for $x_i^+ \in \mathcal{S}^+$, $\theta_j^- = 1$ for $x_j^- \in \mathcal{S}^-$;
return $\boldsymbol{\theta} = \{\boldsymbol{\theta}^+, \boldsymbol{\theta}^-\}$;

The proposed *EvnAttack* enables the adversary to fully take advantage of the property of the feature set, and get a better chance of evading the targeted classifier. $\mathcal{M}$ and $\mathcal{B}$ significantly decrease the number of searches, and thereby reduce the computational complexity. Given $m = \max(|\mathcal{M}|, |\mathcal{B}|)$, the proposed attack *EvnAttack* requires $O(n_t m(\mu^+ + \mu^-))$ queries, in which n_t is the number of testing malware samples, μ^+ and μ^- are the numbers of selected features for elimination and addition respectively. Note that, this algorithm is applicable to the attackers of different skills and capabilities resting on the feature space, the training data set, and the learning algorithm either surrogate or originally used by the targeted system.

V. A Secure-learning Paradigm against Evasion Attack

A defender usually reacts to the evasion attacks by analyzing the attack and retraining the classifier on the new collected file samples, or modifying features of the training dataset to counter the adversary's strategy [25]. However, retraining with adversarial data typically suffers from a limitation: the retrained model modifies the training data distribution approximate to the testing space through the attack model. After modifying a large number of features and malicious files, the model tends to produce a distribution that is very close to that of the benign files. In this case, the retrained model may not be able to differentiate benign and malicious files accurately. In this paper, we provide a systematic model to formalize the impact of the evasion attack with respect to system security and accuracy in malware detection. To this end, we perform our security analysis of the learning-based classifier resting on the application setting that the defender draws the well-crafted *EvnAttack* from the observed sample space, since the attack is modeled as optimization under generic framework in which the attackers try to (1) maximize the number of malware being classified as benign, and (2) minimizes the evasion cost for optimal attacks over the learning-based classifier [21]. Therefore, in our proposed secure-learning model (*SecDefender*), we exploit the *EvnAttack* $\boldsymbol{\theta}$ to retrain the classifier in a progressive way and apply evasion cost $\mathbf{c}$ to regularize the optimization problem.

Classifier Retraining. Incorporating the evasion attack $\boldsymbol{\theta}$ into the learning algorithm can enables us to provide a significant connection between training and the adversarial action. Instead of manipulating the feature spaces for all the malicious training dataset, we start with the original training data $\mathbf{X}$ and iteratively computing a classifier by injecting the adversarial samples tainted by $\boldsymbol{\theta}$ into the training data that evade the previously computed classifier [21]. The new dataset $\mathbf{X}'$ can be formalized as follows:

$$\mathbf{X}' = \mathbf{X} \bigcup_{i=1}^{n_m} (\mathbf{x}_i + \boldsymbol{\theta}), \tag{7}$$

$$s.t.\ f(\mathbf{x}_i + \boldsymbol{\theta}) < 0, \tag{8}$$

where n_m is the total number of malware samples added during the retraining iteration. The iterations converge when there are no new evasion samples generated through the retrained classifier or the specified number of iterations reaches. Compared to updating all the malicious training dataset, this progressive classifier retraining method effectively increases the importance of malware in training process, and can therefore significantly keep the detection system in a more accurate level.

Security Regularization. Resting on the retrained classifier, in our proposed model, we further enhance the security of the classifier by using a security regularization term over the evasion cost. Our empirical studies demonstrate that even retrained by the updated training dataset, the classifiers are still degraded to some extent. It's recalled that an optimal evasion attack aims to manipulate a subset of features with minimum evasion cost while maximize the total loss of classification. In contrast, to secure the classifier in malware detection, we would like to maximize the evasion cost for the attacks [36]: from the analysis of the adversary problem [5], [19], we can find that the larger the evasion cost, the more manipulations need to be performed, and the more difficult the attack is. If a larger number of features has to be manipulated to evade detection, it may be infeasible to perform such attack. Therefore, to be more resilient against the evasion attack, an ideal secure-learning model is to maximize the evasion cost for the attackers. Accordingly, the security of the learning classifier can be defined as:

$$\mathcal{S}(\mathcal{A}(\mathbf{x}), \mathbf{x}) = \frac{1}{\mathcal{C}(\mathcal{A}(\mathbf{x}), \mathbf{x})}, \tag{9}$$

subject to Equation 4. If $\mathcal{A}(\mathbf{x}_i) = \mathbf{x}_i$ which represents that the file is not manipulated by the adversary, $\mathcal{S}(\mathcal{A}(\mathbf{x}_i), \mathbf{x}_i) = 0$. We then define a diagonal matrix for the adversary action denoted as $\mathbf{S} \in \mathbb{R}^{n \times n}$, where the diagonal element $S_{ii} = s(\mathcal{A}(\mathbf{x}_i), \mathbf{x}_i)$ and the remaining elements in the matrix are 0. Based on the concept of label smoothness [32], we can secure the classifier with the constraint as $\mathbf{f'}^T\mathbf{S}\mathbf{y}$. Since the learning-based malware detection can be formalized as an optimization problem denoted by Equation 3, we can then bring a regularization term to enhance its security. This constraint penalizes parameter choices, smooths the effects the attack may cause, and in turn helps to promote the optimal solution for the local minima in the optimization problem. Therefore, to minimize classifier sensitivity to feature manipulation, we can minimize the security regularization term. Based on Equation 3, we can formulate a secure-learning model against the evasion attack as:

$$\begin{aligned}\underset{\mathbf{f'},\mathbf{w},\mathbf{b};\boldsymbol{\xi}}{\operatorname{argmin}}\ \mathcal{L}(\mathbf{f'}, \mathbf{w}, \mathbf{b}; \boldsymbol{\xi}) = \underset{\mathbf{f'},\mathbf{w},\mathbf{b};\boldsymbol{\xi}}{\operatorname{argmin}}\ \frac{1}{2}||\mathbf{y} - \mathbf{f'}||^2 + \frac{\alpha}{2}\mathbf{f'}^T\mathbf{S}\mathbf{y} + \\ \frac{1}{2\beta}\mathbf{w}^T\mathbf{w} + \frac{1}{2\gamma}\mathbf{b}^T\mathbf{b} + \boldsymbol{\xi}^T(\mathbf{f'} - \mathbf{X'}^T\mathbf{w} - \mathbf{b}).\end{aligned} \tag{10}$$

where α is the regularization parameter for the security constraint. As the substitution and derivation from $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0$, $\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = 0$, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}} = 0$, $\frac{\partial \mathcal{L}}{\partial \mathbf{f'}} = 0$, we can get the final secure-learning problem as:

$$((\beta\mathbf{X'}^T\mathbf{X'} + \gamma\mathbf{I}) + \mathbf{I})\mathbf{f'} = (\mathbf{I} - \frac{\alpha}{2}\mathbf{S})(\beta\mathbf{X'}^T\mathbf{X'} + \gamma\mathbf{I})\mathbf{y}. \tag{11}$$

Since the size of $\mathbf{X}'$ is $d \times n$, the computational complexity for Equation 11 is $O(n^3)$. To solve the secure-learning problem (Equation 11), we use conjugate gradient descent method and the implementation of *SecDefender* is shown in Algorithm 2.

Algorithm 2: ***SecDefender*** - A secure-learning model against the evasion attack

Input: Training data set $D = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ and testing set $D_t = \{\mathbf{x}_i, y_i\}_{i=1}^{n_t}$; Evasion attack $\boldsymbol{\theta}$.
Output: $\mathbf{f}'$: the labels of the input files.
Iteratively train classifier $f(\mathbf{X} \bigcup_i (\mathbf{x}_i + \boldsymbol{\theta}))$ to get $\mathbf{X}'$;
$\mathbf{f}'_0 = 0$;
$\mathbf{A} = (\beta \mathbf{X}'^T \mathbf{X}' + \gamma \mathbf{I}) + \mathbf{I}$;
$\mathbf{r}_0 = (\mathbf{I} - \frac{\alpha}{2}\mathbf{S})(\beta \mathbf{X}'^T \mathbf{X}' + \gamma \mathbf{I})\mathbf{y} - \mathbf{A}\mathbf{f}'_0$;
$\mathbf{p}_0 = \mathbf{r}_0$;
$k = 0$;
while $||\mathbf{r}_k|| > \varepsilon$ **do**
  $\lambda_k = (\mathbf{r}_k^T \mathbf{r}_k)/(\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k)$;
  $\mathbf{f}'_{k+1} = \mathbf{f}'_k + \lambda_k \mathbf{p}_k$;
  $\mathbf{r}_{k+1} = \mathbf{r}_k - \lambda_k \mathbf{A} \mathbf{p}_k$;
  $\zeta_k = (\mathbf{r}_{k+1}^T \mathbf{r}_{k+1})/(\mathbf{r}_k^T \mathbf{r}_k)$;
  $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \zeta_k \mathbf{p}_k$;
  $k = k + 1$;
end
return $\mathbf{f}'_k$;

VI. Experimental Results and Analysis

In this section, to empirically validate the proposed methods, we present four sets of experiments: (1) In the first set of experiments, we evaluate the proposed evasion attack model *EvnAttack* with different evasion costs; (2) In the second set of experiments, we then compare *EvnAttack* with other feature manipulation methods; (3) In the third set of experiments, we further evaluate the effectiveness of our proposed secure-learning model *SecDefender* against the evasion attack; (4) In the last set of experiments, we compare the performance of *SecDefender* against the evasion attack with other widely used anti-malware products.

A. Experimental Setup

The sample set obtained from Comodo Cloud Security Center contains $10,000$ file samples with $3,503$ extracted API calls, where $5,000$ are malware, and $5,000$ are benign files. In our experiments, we randomly select 90% of the samples for training, while the remaining 10% is used for testing. To quantitatively validate the effectiveness of the proposed methods, we use the performance measures shown in Table III.

B. Evaluation of EvnAttack with Different Costs

In this set of experiments, we evaluate how different settings of the maximum evasion cost $\delta_{\max}$ may influence the performance of the proposed evasion attack model *EvnAttack*. Since not all of the API calls will contribute to the classification problem as analyzed in Section IV-B, those with extremely low relevance scores (0.0005 in our application) will be excluded for feature manipulations. Therefore, $\mathcal{M}$ = 810, and $\mathcal{B}$ = $1,183$. Considering different manipulation costs for API calls, we optimize c_i for each feature by applying its normalized relevance score for the classification ($c_i \in [0, 1]$). Based on the Cumulative Distribution Function (CDF) of the number of API calls the file samples include, we exploit the average number of API calls that each file possesses, which is 109, to define $\delta_{\max}$. We run our evaluation of *EvnAttack* on the $1,000$ testing file samples with $\delta_{\max}$ varies in {5%, 10%, 15%, 20%, 50%} of the average number of API calls that each file possesses, which is {5, 11, 16, 22, 55}.

TABLE III
Performance measures in malware detection

Measure	Description
TP	Number of files correctly classified as malicious
TN	Number of files correctly classified as benign
FP	Number of files mistakenly classified as malicious
FN	Number of files mistakenly classified as benign
FNR	$FN/(TP + FN)$
TPR/Recall	$TP/(TP + FN)$
Precision	$TP/(TP + FP)$
ACC	$(TP + TN)/(TP + TN + FP + FN)$
F1	$2 \times Precision \times Recall/(Precision + Recall)$

We conduct *EvnAttack* on the testing sample set. From Table IV, we can see that the attack performances are different when the evasion costs vary. Since we just manipulate the features on the testing malicious files (i.e., benign files remain unchanged), all *FP*s and *TN*s after attacks in the experiments keep the same as before attack (i.e., *FP* is 21, and *TN* is 423). The experimental results demonstrate that the performance of the evasion attack with $\delta_{\max} = 22$ is superior to other attacks with the lower values of $\delta_{\max}$ (i.e., 5, 11, 16), since it's more likely that the attack may better evade the detection if more features can be manipulated. To further validate this, we extend the evasion cost $\delta_{\max}$ and show that the attackers can achieve perfect attack (i.e., *FNR* almost reaches to 1, which means almost malware samples are misclassified) when the maximum evasion cost is set to the value of 55 of the average number of API calls.

TABLE IV
Evaluation of *EvnAttack* with different costs

ID	Method	Avg	TP	FN	ACC	FNR
0	*OrgDefender*	0	534	22	95.70%	0.0396
1	$\delta_{\max} = 5$	3	391	165	81.40%	0.2968
2	$\delta_{\max} = 11$	7	275	281	69.80%	0.5054
3	$\delta_{\max} = 16$	10	213	343	63.60%	0.6169
4	$\delta_{\max} = 22$	12	168	388	59.10%	0.6978
5	$\delta_{\max} = 55$	25	13	543	43.60%	0.9766

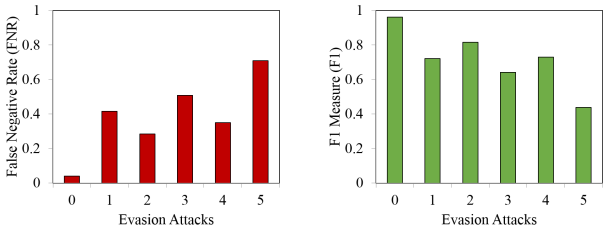
Remark: Avg denotes the average feature manipulation for all 556 testing malware.

As stated above, an ideal evasion attack is to manipulate a subset of features with minimum evasion cost while maximize the total loss of classification. It can be observed from Table IV that when we set $\delta_{\max} = 22$, the average feature manipulation of each file is about 10% of its extracted API calls, which

could be empirically considered as a reasonable trade-off to conduct an evasion attack considering the feature number and manipulation cost. Therefore, in the following experiments, we will set $\delta_{\max} = 22$.

C. Comparisons of EvnAttack and Other Evasion Attacks

Based on the same dataset described in Section VI-A, in this section, as discussed above, given $\delta_{\max} = 22$, we compare our proposed evasion attack *EvnAttack* with other feature manipulation methods including: (1) only manipulating API calls from $\mathcal{B}$ for addition (*Method 1*); (2) only manipulating API calls from $\mathcal{M}$ for elimination (*Method 2*); (3) sequentially selecting $(1/2 \times \delta_{\max})$ API calls from $\mathcal{B}$ for addition and $(1/2 \times \delta_{\max})$ API calls from $\mathcal{M}$ for elimination (*Method 3*); (4) randomly manipulating API calls for addition and elimination (*Method 4*); (5) *EvnAttack*.



(a) False negative rates (b) F1 measures

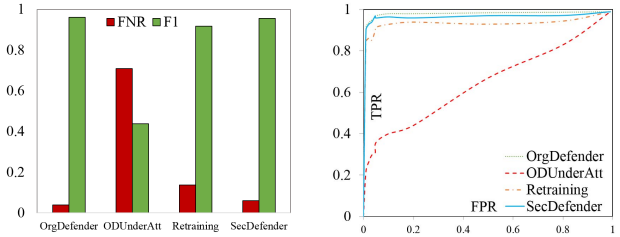
Fig. 1. Comparisons of *EvnAttack* and other evasion attacks: *OrgDefender* (0), different evasion attacks (Method 1 - 4) and *EvnAttack* (5).

The experimental results shown in Fig. 1 illustrate that the attack performances rely on different feature manipulation methods with certain evasion cost $\delta_{\max}$: (1) The manipulation of only feature elimination (*Method 2*) performs worst with lowest *FNR* of 0.2842 (i.e., only 28.42% of the testing malware are misclassified); (2) The manipulation which sequentially selecting features for addition and elimination (*Method 3*) performs better than i) only using feature addition (*Method 1*) or elimination (*Method 2*), and ii) the random attack (*Method 4*); (3) Our proposed evasion attack *EvnAttack* (*Method 5*) can greatly improve the *FNR* to 0.6978 and degrade the detection *F1* measure of the classifier to 0.4384, which outperforms other four feature manipulation methods, due to its well-crafted attack strategy.

D. Evaluation of SecDefender against Evasion Attack

In this section, we further assess the effectiveness of *SecDefender* against the evasion attack. We use *EvnAttack* to taint the malware in the testing sample set, and validate the classification performance in different ways, including: (1) the baseline before attack (*OrgDefender*); (2) the *OrgDefender* under attack (*ODUnderAtt*); (3) the *OrgDefender* retrained using the updated training dataset (i.e., $\mathbf{x} + \boldsymbol{\theta}$) (*Retraining*); (4) *SecDefender*.

The comparisons of the effectiveness of these classifiers are shown in Fig. 2. It can be observed that the retrained classifier only applying the evasion attack $\boldsymbol{\theta}$ to transform the malware in the training dataset from $\mathbf{x}$ to $\mathbf{x} + \boldsymbol{\theta}$ can somehow be resilient to the evasion attacks, but the *FNR* and *F1* still remain unsatisfied. In contrast, our secure-learning model *SecDefender* proposed in Section V can well decrease *FNR* while improve *F1*, and bring the malware detection system back up to the desired performance level, the detection F1 measure of which is 0.9561, approaching the detection results before the attack (i.e., 0.9613). It may also be interested to know how robust that our learning system can combat the random attacks. We conduct this attack by randomly selecting the features for addition or elimination as described in Experiment VI-C (*Method 4*). Under the random attack, the defender has zero knowledge of what the attack is, while we assume the attacker has perfect knowledge of the learning system (i.e., the feature space, the training data set, and the learning algorithm). Even in such case, *SecDefender* can still improve the detection *F1* measure from 0.7304 to 0.8830. Based on these properties, *SecDefender* can be a resilient solution in malware detection even the attackers have perfect knowledge of the learning system while defenders have zero information regarding the attack.



(a) Detection performances (b) ROC Curves for the classifiers

Fig. 2. Comparisons of *SecDefender* and other classification models.

E. Comparisons with Different Anti-malware Scanners

In this set of experiments, we evaluate the performance of *SecDefender* against the evasion attack in comparison with some other popular commercial anti-malware scanners such as Kaspersky (K), McAfee (M), Symantec (S), and TrendMicro (T). For the comparisons, we use all the latest versions of the security products. We use 556 malware samples from the testing dataset described in Section VI-A for evaluation. The testing malware are first tainted by *EvnAttack*, and then scanned by these anti-malware products. The detection results are illustrated in Table V. Compared with these typical anti-malware scanners, *SecDefender* can effectively sustain the *TPR* to 0.9335, and performs the most accurate detection.

VII. Conclusion

In this paper, we explore the arms race between evasion attack and defense to better understand the adversarial machine learning in malware detection. We first define and implement an effective evasion attack model *EvnAttack* which manipulates an optimal portion of the features in a bi-directional way to evade the detection, based on the observation that the

TABLE V
COMPARISONS OF DIFFERENT ANTI-MALWARE SCANNERS

Malware	K	M	S	T	*SecDefender*
1	×	√	×	×	√
2	√	×	√	×	×
3	×	×	×	×	√
4	×	√	×	×	√
5	×	×	×	×	√
6	×	×	×	√	√
7	×	×	×	×	√
8	×	×	×	×	×
9	√	×	√	×	√
10	√	×	×	×	√
⋮	⋮	⋮	⋮	⋮	⋮
556	×	√	√	√	√
TP	508	503	511	498	519
TPR	0.9136	0.9046	0.9190	0.8957	0.9335

API calls differently contribute to the classification of malware and benign files. Accordingly, a secure-learning model *SecDefender*, composed of classifier retraining technique and the security regularization term considering the evasion cost of feature manipulations by attackers, is presented against the evasion attack. Four sets of experiments based on the real sample collection obtained from Comodo Cloud Security Center are conducted to empirically validate the proposed methods. The experimental results demonstrate that the evasion attack model *EvnAttack* with reasonable cost can greatly degrade the detection accuracy of the classifier. To stay resilient against the evasion attack, *SecDefender* can be a robust solution in malware detection even the attackers have perfect knowledge of the learning system. The proposed method can also be readily applied to other malware detection tasks.

ACKNOWLEDGMENTS

The authors would also like to thank the anti-malware experts of Comodo Security Lab for the data collection, as well as the helpful discussions and supports. L. Chen and Y. Ye's work is partially supported by the U.S. National Science Foundation under grant CNS-1618629 and WVU Senate Grants for Research and Scholarship (R-16-043).

REFERENCES

[1] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," in *RAID '07*, 2007, pp. 178–197.

[2] U. Baldangombo, N. Jambaljav, and S.-J. Horng, "A static malware detection system using data mining methods," *CoRR Journal*, 2013.

[3] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Machine Learning*, 2010.

[4] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *ASIACCS '06*, 2006, pp. 16–25.

[5] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *ECML PKDD '13*, 2013, pp. 387–402.

[6] B. Biggio, G. Fumera, and F. Roli, "Design of robust classifiers for adversarial environments," in *SMC '11*, 2011, pp. 977–982.

[7] B. Biggio, F. Roli, and G. Fumera, "Security evaluation of pattern classifiers under attack," *IEEE TKDE*, vol. 26, no. 4, pp. 984–996, 2014.

[8] M. Bruckner, C. Kanzow, and T. Scheffer, "Static prediction games for adversarial learning problems," *JMLR*, vol. 13, no. 1, 2012.

[9] D. H. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos, "Polonium: Tera-scale graph mining for malware detection," in *SDM '11*, 2011, pp. 131–142.

[10] L. Chen, W. Hardy, Y. Ye, and T. Li, "Analyzing file-to-file relation network in malware detection," in *WISE '15*, 2015, pp. 415–430.

[11] L. Chen, T. Li, M. Abdulhayoglu, and Y. Ye, "Intelligent malware detection based on file relation graphs," in *ICSC '15*, 2015, pp. 85–92.

[12] D. Chinavle, P. Kolari, T. Oates, and T. Finin, "Ensembles in adversarial classification for spam," in *CIKM '09*, 2009, pp. 2015–2018.

[13] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," in *KDD '04*, 2004, pp. 99–108.

[14] D. Debarr, H. Sun, and H. Wechsler, "Adversarial spam detection using the randomized hough transform-support vector machine," in *ICMLA '13*, 2013, pp. 299–304.

[15] R. A. Dunne, *A Statistical Approach to Neural Networks for Pattern Recognition*. Wiley-Interscience, 1st edition, 2007.

[16] E. Filiol, G. Jacob, and M. Liard, "Evaluation methodology and theoretical model for antiviral behavioural detection strategies," *Journal in Computer Virology*, vol. 3, no. 1, pp. 23–37, 2007.

[17] KasperskyLab, "4 in 5 malware attacks cause problems for users and 1 in 3 result in money loss," in *http://www.kaspersky.com/about/news/virus/2015/4-in-5-Malware-Attacks-Cause-Problems-for-Users-and-1-in-3-Result-in-Money-Loss*, 2015.

[18] C. Kolbitsch, E. Kirda, and C. Kruegel, "The power of procrastination: detection and mitigation of execution-stalling malicious code," in *CCS '11*, 2011, pp. 285–296.

[19] A. Kolcz and C. H. Teo, "Feature weighting for improved classifier robustness," in *CEAS '09*, 2009.

[20] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in *KDD '04*, 2004, pp. 470–478.

[21] B. Li, Y. Vorobeychik, and X. Chen, "A general retraining framework for scalable adversarial classification," in *NIPS 2016 Workshop on Adversarial Training*, 2016.

[22] D. Lowd and C. Meek, "Adversarial learning," in *KDD '05*, 2005.

[23] M. M. Masud, T. M. Al-Khateeb, K. W. Hamlen, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "Cloud-based malware detection for evolving data streams," *ACM TMIS*, vol. 2, no. 3, pp. 16:1–16:27, 2011.

[24] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[25] F. Roli, B. Biggio, and G. Fumera, "Pattern recognition systems under attack," in *CIARP '13*, 2013, pp. 1–8.

[26] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *SP '01*, 2001.

[27] A. H. Sung, J. Xu, P. Chavez, and S. Mukkamala, "Static analyzer of vicious executables (save)," in *ACSAC '04*, 2004, pp. 326–334.

[28] N. Šrndic and P. Laskov, "Practical evasion of a learning-based classifier: A case study," in *SP '14*, 2014, pp. 197–211.

[29] F. Wang, W. Liu, and S. Chawla, "On sparse feature attacks in adversarial learning," in *ICDM '14*, 2014, pp. 1013–1018.

[30] Y. Wu, T. Ren, and L. Mu, "Importance reweighting using adversarial-collaborative training," in *NIPS '16*, 2016.

[31] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers a case study on pdf malware classifiers," in *NDSS '16*, 2016.

[32] P. Yang and P. Zhao, "A min-max optimization framework for online graph classification," in *CIKM '15*, 2015, pp. 643–652.

[33] Y. Ye, D. Wang, T. Li, and D. Ye, "Imds: Intelligent malware detection system," in *KDD '07*, 2007, pp. 1043–1047.

[34] Y. Ye, D. Wang, T. Li, D. Ye, and Q. Jiang, "An intelligent pe-malware detection system based on association mining," *Journal in Computer Virology*, vol. 4, no. 4, pp. 323–334, 2008.

[35] Y. Ye, T. Li, S. Zhu, W. Zhuang, E. Tas, U. Gupta, and M. Abdulhayoglu, "Combining file content and file relations for cloud based malware detection," in *KDD '11*, 2011, pp. 222–230.

[36] F. Zhang, P. P. K. Chan, B. Biggio, D. S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 766–777, 2015.