An Evasion and Counter-Evasion Study in Malicious Websites Detection

Li Xu[†] Zhenxin Zhan[†] Shouhuai Xu[†] Keying Ye[‡]
[†]Deptartment of Computer Science, University of Texas at San Antonio
[‡]Department of Management Science and Statistics, University of Texas at San Antonio

***Abstract*—Malicious websites are a major cyber attack vector, and effective detection of them is an important cyber defense task. The main defense paradigm in this regard is that the defender uses some kind of machine learning algorithms to train a detection model, which is then used to classify websites in question. Unlike other settings, the following issue is inherent to the problem of malicious websites detection: the attacker essentially has access to the same data that the defender uses to train his/her detection models. This 'symmetry' can be exploited by the attacker, at least in principle, to evade the defender's detection models. In this paper, we present a framework for characterizing the evasion and counter-evasion interactions between the attacker and the defender, where the attacker attempts to evade the defender's detection models by taking advantage of this symmetry. Within this framework, we show that an adaptive attacker can make malicious websites evade powerful detection models, but *proactive training* can be an effective counter-evasion defense mechanism. The framework is geared toward the popular detection model of decision tree, but can be adapted to accommodate other classifiers.**

***Index Terms*—Malicious websites, static analysis, dynamic analysis, evasion, adaptive attacks, proactive training.**

I. Introduction

Compromising websites and abusing them to launch further attacks (e.g., drive-by-download [7], [20]) have become one of the mainstream attack vectors. Unfortunately, it is infeasible, if not impossible, to completely eliminate such attacks, meaning that we must have competent solutions that can detect compromised/malicious websites as soon as possible. The *dynamic* approach, which is often based on client honeypots or variants, can detect malicious websites with high accuracy, but is limited in terms of its scalability. The *static* approach, which often analyzes the website contents and then uses some detection models (e.g., decision trees) to classify them into benign/malicious classes, is very efficient, but suffers from its limited success in dealing with sophisticated attacks (e.g. JavaScript obfuscation). This hints that there is perhaps some inherent limitation in the trade-off between scalability and detection effectiveness.

In this paper, we bring up another dimension of the problem, which may have a fundamental impact on the aforementioned inherent limitation. Unlike in other settings, the following issue is inherent to the problem of malicious websites detection: The attacker essentially has access to the same data that the defender uses to train its detection models. This 'symmetry' could be exploited by the attacker to evade the defender's detection models. This is because the attacker can effectively train and obtain (almost) the same detection models, and then exploit them to make other malicious websites evasive. This is possible because the attacker can manipulate the contents of malicious websites during the course of compromising them, or after they are compromised but before they are analyzed by the defender's detection models. This is feasible because the attacker controls the malicious websites.

More specifically, we make two contributions. First, we propose a framework for characterizing the evasion and counter-evasion interactions between the attacker and the defender. The framework accommodates a set of adaptive attacks against a class of detection models known as decision trees [22], which have been widely used in this problem domain. The framework also accommodates the novel idea of *proactive training* as the counter-evasion mechanism against the adaptive attacks, where the defender proactively trains its detection models while taking adaptive attacks into consideration. Although the framework is geared towards decision trees, it can be adapted to accommodate other kinds of classifiers.

Second, we use a dataset that was collected during the span of 40 days to evaluate the evasion power of adaptive attacks and the counter-evasion effectiveness of proactive training. Experimental results show that an adaptive attacker can make malicious websites evade powerful detection models, but *proactive training* can be an effective counter-evasion defense mechanism. In order to deepen our understanding of the evasion and counter-evasion interactions, we also investigate which features (or attributes) of websites have a high security significance, namely that their manipulation causes the misclassification of malicious websites. Surprisingly, we find that the features of high security significance are almost different from the features that would be selected by the standard feature selection algorithms. This suggests that we might need to design new machine learning algorithms to best fit the domain of security problems. Moreover, we find that the detection accuracy of proactively-trained detection models increases with the degree of the defender's proactiveness (i.e. the number of training iterations). Finally, we find that if the defender does not know the attacker's adaptation strategy, the defender should adopt the `full` adaptation strategy that will be described later.

The rest of the paper is organized as follows. Section II briefly reviews the context of the present study. Section III investigates the framework of evasion and counter-evasion interactions. Section IV evaluates the effectiveness of the framework. Section V discusses related prior work. Section VI concludes the paper.

II. Preliminaries

In order to illustrate the power of adaptive attacks and the effectiveness of our counter-measure against them, we need to consider some concrete detection scheme. Since J48 classifiers are known to be successful in detecting malicious websites [3], [29], [30], [15], [6], [14], we adopt the detection scheme we proposed in [30] as the starting point of the present study. We showed in [30] that J48 classifier outperforms Naive Bayes, Logistic and SVM classifiers.

We also inherit the data collection method described in [30]. At a high level, a crawler is used to fetch the website content corresponding to an input URL, benign and malicious alike. Each URL is described by 105 application-layer features and 19 network-layer features [30]. We now briefly review the following 16 features that will be encountered later: `URL_length` (length of URL); `Content_length` (the content-length field in HTTP header, which may be manipulated by malicious websites); `#Redirect` (number of redirects); `#Scripts` (number of scripts); `#Embedded_URL` (number of URLs embedded); `#Special_character` (number of special characters in a URL); `#Iframe` (number of iframes); `#JS_function` (number of JavaScript functions in a website); `#Long_string` (number of strings with 51 or more letters in embedded JavaScript programs); `#Src_app_bytes` (number of bytes communicated from crawler to website); `#Local_app_packet` (number of crawler-to-website IP packets, including redirects and DNS queries); `Dest_app_bytes` (volume of website-to-crawler communications); `Duration` (the time it takes for the crawler to fetch the contents of a website, including rediects); `#Dist_remote_tcp_port` and `#Dist_remote_IP` (number of distinct TCP ports and IP addresses the crawler uses to fetch websites contents, respectively); `#DNS_query` (number of DNS queries); `#DNS_answer` (number of DNS server's responses).

The main notations are summarized as follows.

MLA	machine learning algorithm
fv	feature vector representing a website
X_z	feature X_z's domain is $[\min_z, \max_z]$
$M_0, \ldots, M_\gamma$	defender's detection schemes (e.g., J48 classifier)
D_0'	training data (feature vectors) for learning M_0
D_0	$D_0 = D_0.malicious \cup D_0.benign$, where malicious feature vectors in $D_0.malicious$ may have been manipulated
$D_0^\dagger$	feature vectors used by defender to proactively train $M_1, \ldots, M_\gamma$; $D_0^\dagger = D_0^\dagger.malicious \cup D_0^\dagger.benign$
α, γ	number of adaptation iterations
$M_i(D_\alpha)$	applying detection scheme M_i to classify feature vectors D_α
$M_{0\text{-}\gamma}(D_\alpha)$	majority vote of $M_0(D_\alpha), \ldots, M_\gamma(D_\alpha)$
ST, C, F	adaptation strategy ST, manipulation algorithm F, manipulation constraints C
$s \overset{R}{\leftarrow} S$	assigning s as a random member of set S

III. Evasion and Counter-Evasion Framework

Adaptive attacks are possible because an attacker can collect the same data as what is used by the defender to train a detection scheme. The attacker also knows the machine learning algorithm(s) the defender uses or even the defender's detection scheme. To accommodate the worst-case scenario, we assume there is a single attacker that coordinates the compromise of websites (possibly by many sub-attackers). This means that the attacker knows which websites are malicious, while the defender aims to detect them. In order to evade detection, the attacker can manipulate some features of the malicious websites. The manipulation operations can take place during the course of compromising websites, or after compromising websites but before they are classified by the defender's detection scheme.

A. Framework Overview

We describe adaptive attacks and countermeasures in a modular fashion, by using eight algorithms whose caller-callee relation is highlighted in Figure 1. Algorithm 1 is the attacker's main algorithm, which calls Algorithm 2 for preprocessing, and calls Algorithm 4 or Algorithm 5 for selecting features to manipulate and for determining the manipulated values for the selected features. Both Algorithm 4 and Algorithm 5 call Algorithm 3 to compute the *escape intervals* for the features that are to be manipulated. An escape interval defines the interval from which the manipulated value of a feature should be taken so as to evade detection.

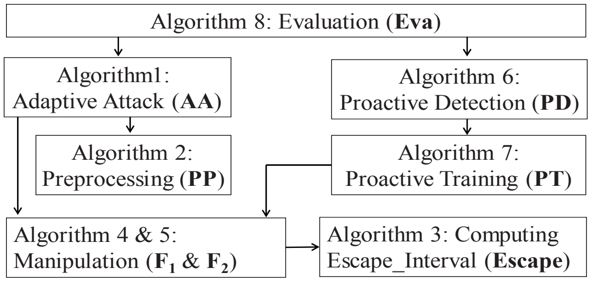

Algorithm 8: Evaluation (**Eva**)
Algorithm1: Adaptive Attack (**AA**)
Algorithm 6: Proactive Detection (**PD**)
Algorithm 2: Preprocessing (**PP**)
Algorithm 7: Proactive Training (**PT**)
Algorithm 4 & 5: Manipulation ($\mathbf{F_1}$ & $\mathbf{F_2}$)
Algorithm 3: Computing Escape_Interval (**Escape**)

Fig. 1. Caller-callee relation between the algorithms.

Algorithm 6 is the defender's main algorithm, which calls Algorithm 7 for proactive training of detection schemes. For this purpose, the defender can have access to its own proactive manipulation algorithm. In our experiments, we let the defender have access to the manipulations algorithms that are available to the attacker, namely Algorithm 4 and Algorithm 5. This is sufficient for the purpose of understanding the effectiveness of proactive training and detection against adaptive attacks under various algorithm/parameter possibilities, such as: the defender correctly or incorrectly "guess" the manipulation algorithm or parameters that are used by the attacker, and the relatively more effective proactive training strategy against a class of adaptive attacks. In order to evaluate the effectiveness of proactive training and detection against adaptive attacks, we use an "artificial" Algorithm 8, which is often implicit in most real-life defense operations.

B. Evasion Model and Algorithms

In our model, a website is represented by a feature vector. We call the feature vector representing a benign website *benign feature vector*, and the feature vector representing a malicious website *malicious feature vector*. Denote by D'_0 the defender's *training data*, namely a set of feature vectors corresponding to a set of benign websites (denoted by $D'_0.benign$) and malicious websites (denoted by $D'_0.malicious$). The defender uses a machine learning algorithm MLA to learn a detection scheme M_0 from D'_0 (i.e., M_0 is learned from one portion of D'_0 and tested via the other portion of D'_0). As mentioned above, the attacker is given M_0 to accommodate the worst-case scenario. Denote by D_0 the set of feature vectors that are to be classified by M_0 to determine which feature vectors (i.e., the corresponding websites) are malicious. The attacker's objective is to manipulate the malicious feature vectors in D_0 into some D_α so that $M_0(D_\alpha)$ has a high false-negative rate, where $\alpha > 0$ represents the number of iterations (or rounds) the attacker conducts the manipulation operations.

Algorithm 1 Adaptive attack $\mathsf{AA}(\mathsf{MLA}, M_0, D_0, \mathsf{ST}, \mathsf{C}, \mathsf{F}, \alpha)$

INPUT:M_0 is defender's detection scheme, $D_0 = D_0.malicious \cup D_0.benign$ where malicious feature vectors ($D_0.malicious$) are to be manipulated (to evade detection of M_0), ST is attacker's adaptation strategy, C is a set of manipulation constraints, F is attacker's manipulation algorithm, α is attacker's number of adaptation rounds
OUTPUT: D_α

```
initialize array $D_1, \ldots, D_\alpha$
for $i$=1 to $\alpha$ do
    if ST == parallel-adaptation then
        $D_i \leftarrow \mathsf{F}(M_0, D_0, \mathsf{C})$   {manipulated version of $D_0$}
    else if ST == sequential-adaptation then
        $D_i \leftarrow \mathsf{F}(M_{i-1}, D_{i-1}, \mathsf{C})$  {manipulated version of $D_0$}
    else if ST == full-adaptation then
        $\mathcal{D}_{i-1} \leftarrow \mathsf{PP}(D_0, \ldots, D_{i-2})$  {see Algorithm 2}
        $D_i \leftarrow \mathsf{F}(M_{i-1}, \mathcal{D}_{i-1}, \mathsf{C})$  {manipulated version of $D_0$}
    end if
    if $i < \alpha$ then
        $M_i \leftarrow \mathsf{MLA}(D_i)$ {$D_1, \ldots, D_{\alpha-1}, M_1, \ldots, M_{\alpha-1}$ are not used when ST==parallel-adaptation}
    end if
end for
return $D_\alpha$
```

Algorithm 1 describes the adaptive attack. As highlighted in Figure 2, we consider three basic adaptation strategies.

- ST == `parallel-adaptation`: The attacker sets the manipulated $D_i = \mathsf{F}(M_0, D_0, \mathsf{C})$, where $i = 1, \ldots, \alpha$, and F is a randomized manipulation algorithm, meaning that $D_i = D_j$ for $i \neq j$ is unlikely.
- ST == `sequential-adaptation`: The attacker sets the manipulated $D_i = \mathsf{F}(M_{i-1}, D_{i-1}, \mathsf{C})$ for $i = 1, \ldots, \alpha$, where detection schemes $M_1, \ldots, M_\alpha$ are respectively learned from $D_1, \ldots, D_\alpha$ using the defender's machine learning algorithm MLA (also known to the attacker).
- ST == `full-adaptation`: The attacker sets the manipulated $D_i = \mathsf{F}(M_{i-1}, \mathsf{PP}(D_0, \ldots, D_{i-1}), \mathsf{C})$ for $i = 1, 2, \ldots$, where $\mathsf{PP}(\cdot, \ldots)$ is a preprocessing algorithm for "aggregating" sets of feature vectors $D_0, D_1, \ldots$ into a single set of feature vectors, F is a manipulation algorithm, $M_1, \ldots, M_\alpha$ are learned respectively from $D_1, \ldots, D_\alpha$ by the attacker.

Algorithm 2 is a concrete preprocessing algorithm. Its basic idea is the following: since each malicious website corresponds to m malicious feature vectors that respectively belong to $D_0, \ldots, D_{m-1}$, the preprocessing algorithm randomly picks one of the m malicious feature vectors to represent the malicious website in $\mathcal{D}$. It is worth mentioning that one can derive some hybrid attack strategies from the above three basic strategies. We also note that the attack strategies and the manipulation constraints are independent of the detection schemes, but the manipulation algorithms would be specific to the detection schemes.

Algorithm 2 Preprocessing $\mathsf{PP}(D_0, \ldots, D_{m-1})$

INPUT: m sets of feature vectors $D_0, \ldots, D_{m-1}$ where the zth malicious website corresponds to $D_0.malicious[z], \ldots, D_{m-1}.malicious[z]$
OUTPUT: $\mathcal{D}$

```
$\mathcal{D} \leftarrow \emptyset$
size $\leftarrow$ sizeof($D_0.malicious$)
for $z = 1$ to size do
    $\mathcal{D}[z] \overset{R}{\leftarrow} \{D_0.malicious[z], \ldots, D_{m-1}.malicious[z]\}$
    $\mathcal{D} \leftarrow \mathcal{D} \cup D_0.benign$
end for
return $\mathcal{D}$
```

Manipulation Constraints. For a feature X whose value is to be manipulated, the attacker needs to compute $X.escape\_interval$, which is a subset of feature X's domain $domain(X)$ and can possibly cause the malicious feature vector to evade detection. Feature X's manipulated value is randomly chosen from its $escapte\_interval$, which is calculated using Algorithm 3, while taking as input X's domain constraints and semantics constraints.

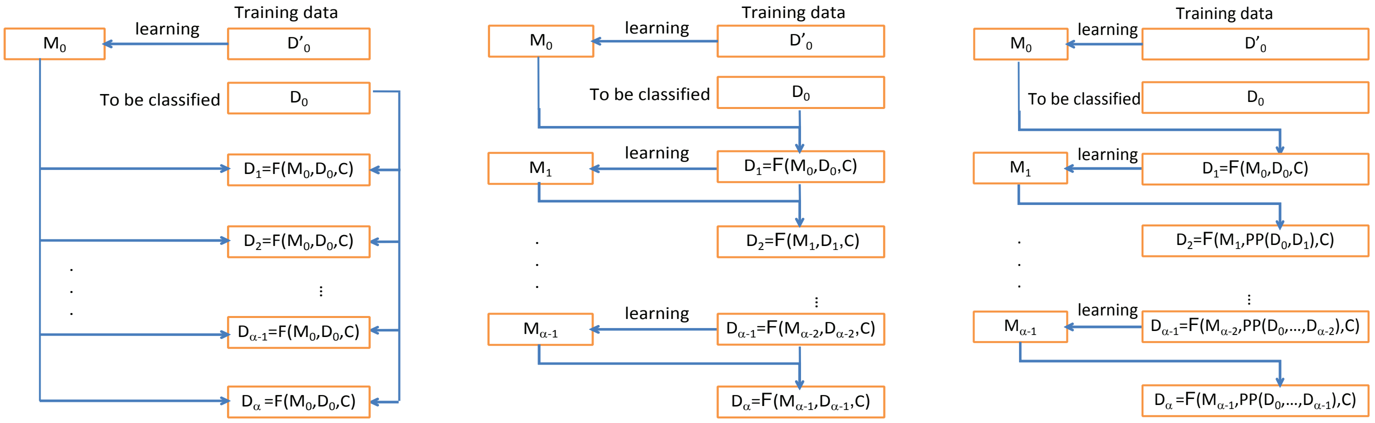
Algorithm 3 X's escape_interval $\mathsf{Escape}(X, M, \mathsf{C})$

INPUT: X is feature for manipulation, M is detection scheme, C represents constraints
OUTPUT: X's $escape\_interval$

```
$domain\_constraint \leftarrow \mathsf{C}.domain\_map(X)$
$semantics\_constraint \leftarrow \mathsf{C}.semantics\_map(X)$   {$\emptyset$ if $X$ cannot be manipulate due to semantics constraints}
$escape\_interval \leftarrow domain\_constraint \cap semantics\_constraint$
return $escape\_interval$
```

Algorithm 3 is called because the manipulation algorithm needs to compute the interval from which a feature's manipulated value should be taken. Specifically, the constraints are the following.

- **Domain constraints**: Each feature has its own domain of possible values. This means that the new value of a feature after manipulation must fall into the domain of the feature. Let $\mathsf{C}.domain\_map$ be a table of $(key, value)$


Training data
learning
M_0
D'_0
To be classified
D_0
D_1=F(M_0,D_0,C)
D_2=F(M_0,D_0,C)
$D_{\alpha-1}$=F(M_0,D_0,C)
D_α =F(M_0,D_0,C)
M_1
$M_{\alpha-1}$
D_2=F(M_1,D_1,C)
$D_{\alpha-1}$=F($M_{\alpha-2}$,$D_{\alpha-2}$,C)
D_α=F($M_{\alpha-1}$,$D_{\alpha-1}$,C)
D_2=F(M_1,PP(D_0,D_1),C)
$D_{\alpha-1}$=F($M_{\alpha-2}$,PP(D_0,…,$D_{\alpha-2}$),C)
D_α=F($M_{\alpha-1}$,PP(D_0,…,$D_{\alpha-1}$),C)

(a) Parallel adaptation strategy

(b) Sequential adaptation strategy

(c) Full adaptation strategy

Fig. 2. Adaptive attack algorithm $\mathsf{AA}(\mathsf{MLA}, M_0, D_0, \mathsf{ST}, \mathsf{C}, \mathsf{F}, \alpha)$, where D'_0 is the defender's training data, M_0 is the defender's detection scheme that is learned from D'_0 by using MLA, D_0 is the feature vectors that are examined by M_0 in the absence of adaptive attacks, ST is the attacker's adaptation strategy, C is a set of manipulation constraints, F is the attacker's (deterministic or randomized) manipulation algorithm that maintains the set of constraints C, α is the number of rounds the attacker runs its manipulation algorithms. D_α is the manipulated version of D_0 with malicious feature vectors $D_0.malicious$ manipulated. The attacker's objective is make $M_0(D_\alpha)$ have high false-negative rate.

pairs, where key is feature name and $value$ is the feature's domain constraint. Let $\mathsf{C}.domain\_map(X)$ return feature X's domain as defined in $\mathsf{C}.domain\_map$.

- **Semantics constraints**: The manipulation of feature values should have no side-effect to the attack, or at least cannot invalidate the attacks. For example, if a malicious website needs to use script to launch the drive-by-download attack, the feature indicating the number of scripts cannot be manipulated to 0. Let $\mathsf{C}.semantics\_map$ be a table of $(key, value)$ pairs, where key is feature name and $value$ is the feature's *semantics constraints*. Let $\mathsf{C}.semantics\_map(X)$ return feature X's semantics constraints as specified in $\mathsf{C}.attack\_map$.

In general, constraints might have to be manually identified based on feature definitions and domain knowledge.

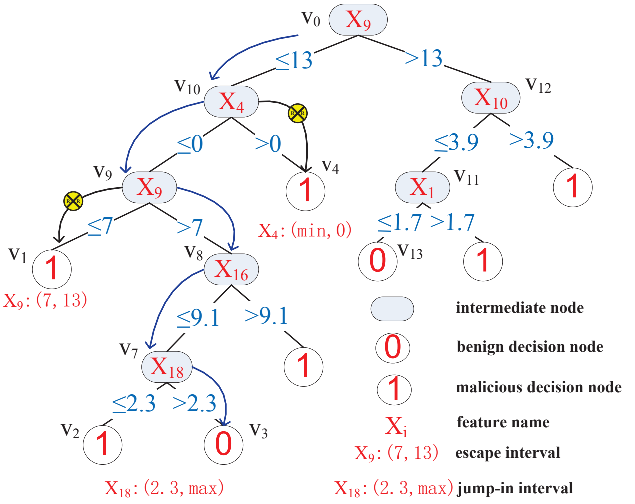


Fig. 3. Example J48 classifier and feature manipulation. For inner node v_{10} on the $benign\_path$ ending at $benign\_leaf$ v_3, we have $v_{10}.feature = $ "X_4" and $v_{10}.feature.value = X_4.value$.

Manipulation Algorithms. As mentioned in Section II, we adopt the J48 classifier detection scheme, where a J48 classifier is trained by concatenating the application- and network-layer features corresponding to the same URL [30]. We present two manipulation algorithms, called F_1 and F_2, which exploit the defender's J48 classifier to guide the manipulation of features. Both algorithms neither manipulate the benign feature vectors (which are not controlled by the attacker), nor manipulate the malicious feature vectors that are already classified as benign by the defender's detection scheme (i.e., false-negative). Both algorithms may fail, while brute-forcing may fail as well because of the manipulation constraints.

Since the manipulation algorithms are inevitably complicated, in the following we will present their basic ideas and sketched algorithms. The notations used in the algorithms are: for node v in the classifier, $v.feature$ is the feature associated to node v, and $v.value$ is $v.feature$'s "branching" value as specified by the classifier (a binary tree with all features numericalized).

Manipulation Algorithm F_1 is described as Algorithm 4. The basic idea underlying this manipulation algorithm is the following: for every malicious feature vector in D, there is a unique path (in the J48 classifier M) that leads to a *malicious leaf*, which indicates that the feature vector is malicious. We call the path leading to malicious leaf a *malicious path*, and the path leading to a *benign leaf* (which indicates a feature vector as benign) a *benign path*.

By examining the path from the malicious leaf to the root, say $malicious\_leaf \rightarrow v_2 \rightarrow \ldots \rightarrow root$, and identifying the first inner node, namely v_2, the algorithm attempts to manipulate $\text{fv}.(v_2.feature).value$ so that the classification can lead to $malicious\_leaf$'s sibling, say $v_{2,another\_child}$, which is guaranteed to exist (otherwise, v_2 cannot be an inner node). Note that there must be a sub-path rooted at $v_{2,another\_child}$ that leads to a $benign\_leaf$ (otherwise, v_2 cannot be an inner node as well), and that manipulation of values of the features corresponding to the nodes on the sub-tree rooted at

Algorithm 4 Manipulation algorithm $\mathsf{F}_1(M, D, \mathsf{C})$

```
INPUT: J48 classifier $M$, feature vector set $D$(malicious $\cup$ benign), manipulation constraints $C$
OUTPUT: manipulated feature vectors
for all feature vector $\in D$.malicious do
    $v$ be the root node of $M$
    maintain an interval for every feature in feature vector.
    while $v$ is not benign leaf do
        if $v$ is an inner node then
            $v \leftarrow v.Child$ based on decision tree rule
            update corresponding feature interval
        else if $v$ is a malicious leaf then
            compute the corresponding feature's escape_interval by calling Algorithm 3
            pick a value $z$ in the escape_interval uniformly at random
            set the corresponding feature's value to $z$
            $v \leftarrow v.sibling$
        end if
    end while
end for
return manipulated feature vectors $D$
```

$v_{2,another\_child}$ will preserve the postfix $v_2 \rightarrow \ldots \rightarrow root$.

To help understand the manipulation algorithm, let us look at one example. At a high-level, the attacker runs $\mathsf{AA}(\text{“}J48\text{”}, M_0, D_0, \mathsf{ST}, \mathsf{C}, \mathsf{F}_1, \alpha = 1)$ and therefore $\mathsf{F}_1(M_0, D_0, \mathsf{C})$ to manipulate the feature vectors, where ST can be any of the three strategies because they cause no difference when $\alpha = 1$ (see Figure 2). Consider the example J48 classifier M in Figure 3, where features and their values are for illustration purpose, and the leaves are decision nodes with class 0 indicating *benign leaves* and 1 indicating *malicious leaves*. A website with feature vector

$$(X_4 = -1, X_9 = 5, X_{16} = 5, X_{18} = 5)$$

is classified as malicious because it leads to decision path

$$v_0 \xrightarrow{X_9 \leq 13} v_{10} \xrightarrow{X_4 \leq 0} v_9 \xrightarrow{X_9 \leq 7} v_1,$$

which ends at malicious leaf v_1. The manipulation algorithm first identifies malicious leaf v_1's parent node v_9, and manipulates X_9's value to fit into v_1's sibling (v_8). Note that X_9's *escape_interval* is as:

$$([\min_9, \max_9] \setminus [\min_9, 7]) \cap [\min_9, 13] = (7, 13],$$

where $Domain(X_9) = [\min_9, \max_9]$, $[\min_9, 7]$ corresponds to node v_9 on the path, and $[\min_0, 13]$ corresponds to node v_0 on the path. The algorithm manipulates X_9's value to be a random element from X_9's $escapte\_interval$, say $8 \in (7, 13]$, which causes the manipulated feature vector to evade detection because of decision path:

$$v_0 \xrightarrow{X_9 \leq 13} v_{10} \xrightarrow{X_4 \leq 0} v_9 \xrightarrow{X_9 > 7} v_8 \xrightarrow{X_{16} \leq 9.1} v_7 \xrightarrow{X_{18} > 2.3} v_3$$

and ends at benign leaf v_3.

Manipulation Algorithm F_2 is described as Algorithm 5. The basic idea underlying this manipulation algorithm is to first extract all benign paths. For each feature vector $\mathsf{fv} \in D.malicious$, F_2 keeps track of the mismatches between fv and all benign paths. The algorithm attempts to manipulate as few "mismatched" features as possible to evade M.

Algorithm 5 Manipulation algorithm $\mathsf{F}_2(M, D, \mathsf{C})$

```
INPUT: J48 classifier $M$, feature vector set $D$(malicious $\cup$ benign), manipulation constraints $C$
OUTPUT: manipulated feature vectors
create interval vector for features along every benign path and store in Paths.
for all feature vector $\in D$.malicious do
    for all $Path \in$ Paths do
        compare feature vector to $Path$ and record the mismatch feature number
    end for
    sort Paths in ascending order of mismatch feature number
    for all $Path \in$ Paths until successfully manipulated do
        for all mismatch feature $\in$ Paths do
            get escape interval by calling Algorithm 3
            pick a value $n$ in escape interval at random
            set feature vector's corresponding feature value to $n$
        end for
    end for
end for
return manipulated feature vectors $D$
```

To help understand this manipulation algorithm, let us look at another example. Consider feature vector

$$(X_4 = .3, X_9 = 5.3, X_{16} = 7.9, X_{18} = 2.1, X_{10} = 3, X_1 = 2.3),$$

which is classified as malicious because of path

$$v_0 \xrightarrow{X_9 \leq 13} v_{10} \xrightarrow{X_4 > 0} v_4.$$

To evade detection, the attacker can compare the feature vector to the matrix of two benign paths. For the benign path $v_3 \rightarrow v_7 \rightarrow v_8 \rightarrow v_9 \rightarrow v_{10} \rightarrow v_0$, the feature vector has three mismatches, namely features X_4, X_9, X_{18}. For the benign path $v_{13} \rightarrow v_{11} \rightarrow v_{12} \rightarrow v_0$, the feature vector has two mismatches, namely X_9 and X_1. The algorithm first processes the benign path ending at node v_{13}. The algorithm will try to manipulate X_9 and X_1's values to reach the benign leaf. Suppose on the other hand, that X_{10} cannot be manipulated without violating the constraints. The algorithm stops with this benign path and considers the benign path end at node v_3. If the algorithm fails with this benign path again, the algorithm will not manipulate the feature vector and leave it to be classified as malicious.

C. Counter-Evasion Algorithms

We have showed that adaptive attacks can ruin the defender's (non-proactive) detection schemes. Now we investigate counter-measure against adaptive attacks. The counter-measure is based on the idea of *proactive training & detection*. Algorithm 6 describes the *proactive detection* algorithm. The basic idea of this algorithm is to call the *proactive training* algorithm to generate a set of proactively trained detection schemes, denoted by $M_1^\dagger, \ldots, M_\gamma^\dagger$. These detection schemes are derived from, among other things, M_0, which is learned from D_0' using MLA. It is important to note that $D_0' = D_0'.benign \cup D_0'.malicious$, where $D_0'.benign$ is a set of websites that are known to

Algorithm 6 Proactive detection $\mathsf{PD}(\mathsf{MLA}, M_0, D_0^\dagger, D_\alpha, \mathsf{ST}_D, \mathsf{C}, \mathsf{F}_D, \gamma)$

```
INPUT: $M_0$ is learned from $D_0'$ using $\mathsf{MLA}$, $D_0^\dagger = D_0^\dagger.benign \cup D_0^\dagger.malicious$, $D_\alpha$ ($\alpha$ unknown to defender) is set of feature vectors for classification (where the malicious websites may have been manipulated by the attacker), $\mathsf{ST}_D$ is defender's adaptation strategy, $\mathsf{F}_D$ is defender's manipulation algorithm, $\mathsf{C}$ is set of constraints, $\gamma$ is defender's number of adaptations rounds
OUTPUT: malicious vectors $\mathsf{fv} \in D_\alpha$
$M_1^\dagger, \ldots, M_\gamma^\dagger \leftarrow \mathsf{PT}(\mathsf{MLA}, M_0, D_0^\dagger, \mathsf{ST}_D, \mathsf{C}, \mathsf{F}_D, \gamma)$  {see Algorithm 7}
$\mathsf{malicious} \leftarrow \emptyset$
for all $\mathsf{fv} \in D_\alpha$ do
    if ($M_0(\mathsf{fv})$ says $\mathsf{fv}$ is malicious) OR (majority of $M_0(\mathsf{fv}), M_1^\dagger(\mathsf{fv}), \ldots, M_\gamma^\dagger(\mathsf{fv})$ say $\mathsf{fv}$ is malicious) then
        $\mathsf{malicious} \leftarrow \mathsf{malicious} \cup \{\mathsf{fv}\}$
    end if
end for
return $\mathsf{malicious}$
```

be benign (ground truth) and cannot be manipulated by the attacker, $D_0'.malicious$ is a set of websites that are known to be malicious (ground truth). A website is classified as malicious if the non-proactive detection scheme M_0 classifies it as malicious, or at least $\lfloor(\gamma+1)/2\rfloor+1$ of the proactively trained detection schemes classify it as malicious. This is meditated to accommodate that the defender does not know *a priori* whether the attacker is adaptive or not, When the attacker is not adaptive, M_0 can effectively deal with D_0.

Algorithm 7 describes the proactive training algorithm. This algorithm is similar to the adaptive attack algorithm AA because it also consider three kinds of adaptation strategies. Specifically, this algorithm aims to derive detection schemes $M_1^\dagger, \ldots, M_\gamma^\dagger$ from the starting-point detection scheme M_0.

Algorithm 7 Proactive training $\mathsf{PT}(\mathsf{MLA}, M_0, D_0^\dagger, \mathsf{ST}_D, \mathsf{C}, \mathsf{F}_D, \gamma)$

```
INPUT: same as in Algorithm 6
OUTPUT: $M_1^\dagger, \ldots, M_\gamma^\dagger$
$M_0^\dagger \leftarrow M_0$  {for simplifying notations}
initialize $D_1^\dagger, \ldots, D_\gamma^\dagger$ and $M_1^\dagger, \ldots, M_\gamma^\dagger$ respectively as empty sets and empty classifiers
for $i$=1 to $\gamma$ do
    if $\mathsf{ST}_D$ == parallel-adaptation then
        $D_i^\dagger.malicious \leftarrow \mathsf{F}_D(M_0^\dagger, D_0^\dagger.malicious, \mathsf{C})$
    else if $\mathsf{ST}_D$ == sequential-adaptation then
        $D_i^\dagger.malicious \leftarrow \mathsf{F}_D(M_{i-1}^\dagger, D_{i-1}^\dagger.malicious, \mathsf{C})$
    else if $\mathsf{ST}_D$ == full-adaptation then
        $\mathcal{D}_{i-1}^\dagger.malicious \leftarrow \mathsf{PP}(D_0^\dagger, \ldots, D_{i-2}^\dagger)$
        $D_i^\dagger.malicious \leftarrow \mathsf{F}_D(M_{i-1}^\dagger, \mathcal{D}_{i-1}^\dagger, \mathsf{C})$
    end if
    $D_i^\dagger.benign \leftarrow D_0^\dagger.benign$
    $M_i^\dagger \leftarrow \mathsf{MLA}(D_i^\dagger)$
end for
return $M_1^\dagger, \ldots, M_\gamma^\dagger$
```

Algorithm 8 describes the algorithm for evaluating the effectiveness of the counter-measure against the adaptive attacks. Essentially, the evaluation algorithm calls the defender's protection detection to generate a set of proactively trained detection schemes, and calls the attacker's adaptive attack algorithm to manipulate the malicious websites (i.e., selecting some of their features and manipulating their values to evade a given detection scheme). By varying the adaptation strategies and the parameters, we can evaluate the effectiveness of proactive training & detection against the adaptive attacks. The parameter space of the evaluation algorithm includes at least 108 scenarios: the basic adaptation strategy space $\mathsf{ST}_A \times \mathsf{ST}_D$ is 3×3 (i.e., not counting any hybrids of `parallel-adaptation`, `sequential-adaptation` and `full-adapatation`), the manipulation algorithm space $\mathsf{F}_A \times \mathsf{F}_B$ is 2×2, and the adaptation round parameter space is at least 3 ($\alpha >, =, < \gamma$).

Algorithm 8 Proactive defense vs. adaptive attack evaluation $\mathsf{Eva}(\mathsf{MLA}, M_0, D_0^\dagger, D_0, \mathsf{ST}_A, \mathsf{F}_A, \mathsf{ST}_D, \mathsf{F}_D, \mathsf{C}, \alpha, \gamma)$

```
INPUT: detection scheme $M_0$ (learned from $D_0'$, as in Algorithm 7), $D_0^\dagger$ is set of feature vectors for defender's proactive training, $D_0 = D_0.malicious \cup D_0.benign$, $\mathsf{ST}_A$ ($\mathsf{ST}_D$) is attacker's (defender's) adaptation strategy, $\mathsf{F}_A$ ($\mathsf{F}_D$) is attacker's (defender's) manipulation algorithm, $\mathsf{C}$ is the constraints, $\alpha$ ($\gamma$) is the number of attacker's (defender's) adaptation rounds
OUTPUT: ACC, FN, TP and FP
if $\alpha > 0$ then
    $D_\alpha \leftarrow \mathsf{AA}(\mathsf{MLA}, M_0, D_0, \mathsf{ST}_A, \mathsf{C}, \mathsf{F}_A, \alpha)$
    {call Algorithm 1}
end if
$M_1^\dagger, \ldots, M_\gamma^\dagger \leftarrow \mathsf{PT}(\mathsf{MLA}, M_0, D_0^\dagger, \mathsf{ST}_D, \mathsf{C}, \mathsf{F}_D, \gamma)$
{call Algorithm 7}
$\mathsf{malicious} \leftarrow \mathsf{PD}(\mathsf{MLA}, M_0, D_0^\dagger, D_\alpha, \mathsf{ST}_D, \mathsf{C}, \mathsf{F}_D, \gamma)$
{call Algorithm 6}
$\mathsf{benign} \leftarrow D_\alpha \setminus \mathsf{malicious}$
calculate ACC, FN, TP and FP w.r.t. $D_0$
return ACC, FN, TP and FP
```

IV. Evaluating Effectiveness of the Counter-Evasion Algorithms

We use the standard metrics, including false-negative and false-positive rates [30], to evaluate the effectiveness of counter-evasion algorithms.

A. Data Description

The dataset used in this paper consists of a 40-day URLs. Malicious URLs are downloaded from blacklists: compuweb.com/url-domain-bl.txt, malware.com.br, malwaredomainlist.com, zeustracker.abuse.ch and spyeyetracker.abuse.ch and further confirmed with the high-interaction client honeypot Capture-HPC[24]. Benign URLs are obtained from alexa.com, which lists the top 10,000 websites that are supposed to be well protected. The test of blacklist URLs using high-interaction client honeypot confirmed our observation that some or many blacklist URLs are not accessible any more and thus should not be counted as malicious URLs.

The daily average number of malicious websites listed in blacklists is 6763 and the daily average number of malicious websites after being verified by Capture-HPC is 838. The total number of distinct malicious websites we found in 40 days are 17091. The daily average number of benign websites is 10,000. By eliminating non-accessible benign websites, the daily average number of benign websites is 9501 in 40 days. According to [26] and our experiment results, it can achieve best detection rate, when the rate between the number of benign websites and malicious websites are 4:1. So we choose all malicious websites and 4 times of benign websites as our training and testing data.

B. Effectiveness of the Evasion Attacks

Table I summarizes the results of adaptive attack $\mathsf{AA}(\text{``}J48\text{''}, M_0, D_0, \mathsf{ST}, \mathsf{C}, \mathsf{F}, \alpha = 1)$ based on the 40-day dataset mentioned above. The experiment can be more succinctly represented as $M_0(D_1)$, meaning that the defender is static (or non-proactive) and the attacker is adaptive with $\alpha = 1$, where D_1 is the manipulated version of D_0. Note that in the case of $\alpha = 1$, the three adaptation strategies lead to the same D_1 as shown in Figure 2. From Table I, we find that both manipulation algorithms can effectively evade detection by manipulating on average 4.31-7.23 features while achieving false-negative rate 87.6%-94.7% for F_1, and by manipulating on average 4.01-6.19 features while achieving false-negative rate 89.1%-95.3% for F_2

TABLE I
EXPERIMENT RESULTS WITH $M_0(D_1)$ IN TERMS OF AVERAGE FALSE-NEGATIVE RATE (FN), AVERAGE NUMBER OF MANIPULATED FEATURES (#MF), AVERAGE PERCENTAGE OF FAILED ATTEMPTS (FA).

	F_1			F_2		
Detection Scheme	FN	#MF	FA	FN	#MF	FA
J48 Decision Tree	87.6%	7.23	12.6%	89.1%	6.19	11.0%

Having observed the phenomenon that manipulation of some features' values can essentially make the detection schemes useless, it would be natural to ask ***which features are often manipulated for evasion?*** To look into the question, we notice that many features are manipulated over the 40 days, but only a few are manipulated often.

F_1 most often (i.e., $>$ 150 times each day for over the 40 days) manipulates three application-layer features — `URL_length`, `Content_length`, `#Embedded_URLs` — and two network-layer features — `Duration` and `#Local_app_packet`. On the other hand, F_2 most often (i.e., $>$ 150 times) manipulates two application-layer features — `#Special_characters` and `Content_length` — and one network-layer feature — `Duration`.

The above discrepancy between the frequencies that features are manipulated can be attributed to the design of the manipulation algorithms. Specifically, F_1 seeks to manipulate features that are associated to nodes that are close to the leaves. In contrast, F_2 emphasizes on the mismatches between a malicious feature vector and an entire benign path, which represents a kind of global search and also explains why F_2 manipulates fewer features.

We also want to know ***why these features have such high security/evasion significance?*** The issue is important because identifying the "important" features could lead to deeper insights. We compare the manipulated features to the features that would be selected by a feature selection algorithm for the purpose of training classifiers. To be specific, we use the $\mathsf{InfoGain}$ feature selection algorithm because it ranks the contributions of individual features [30]. We find that among the manipulated features, `URL_length` is the only feature among the five $\mathsf{InfoGain}$-selected application-layer features, and `#Dist_remote_TCP_port` is the only feature among the four $\mathsf{InfoGain}$-selected network-layer features. This suggests that the feature selection algorithm does not necessarily offer good insights into the importance of features from a security perspective.

The standard feature-selection algorithms are almost useless to find indicative features from the perspective of evading classifiers. There is still gap between our results and the "optimal" solutions based on security semantics; it's an open problem to bridge the gap, because classifiers are "black-box" that don't really accommodate "security semantics" of features.

C. Effectiveness of the Counter-Evasion Algorithms

Table II summarizes the effectiveness of proactive defense against adaptive attacks. We make the following observations. First, if the defender is proactive (i.e., $\gamma > 0$) but the attacker is non-adaptive (i.e., $\alpha = 0$), the false-negative rate drops from 0.79% in the baseline case to some number belonging to interval $[0.23\%, 0.56\%]$. The price is: the detection accuracy drops from 99.68% in the baseline case to some number belonging to interval $[99.23\%, 99.68\%]$ the false-positive rate increases from 0.14% in the baseline case to some number belonging to interval $[0.20\%, 0.93\%]$, The above observations suggest: **the defender can always use proactive detection without worrying about side-effects (e.g., when the attacker is not adaptive)**. This is because the proactive detection algorithm PD uses $M_0(D_0)$ as the first line of detection.

Second, when $\mathsf{ST}_A = \mathsf{ST}_D \neq 0$, it has a significant impact whether or not they use the same manipulation algorithm. This phenomenon also can be explained by that the features that are often manipulated by F_1 are very different from the features that are often manipulated by F_2. More specifically, when $\mathsf{F}_A = \mathsf{F}_D$, the proactively learned classifiers $M_1^\dagger, \ldots, M_\gamma^\dagger$ would capture the "maliciousness" information embedded in the manipulated data D_α; this would not be true when $\mathsf{F}_A \neq \mathsf{F}_D$. Moreover, the `sequential` adaptation strategy appears to be more "oblivious" than the other two strategies in the sense that D_α preserves less information about D_0. what adaptation strategy should the defender use to counter $\mathsf{ST}_A =$ `sequential`? Table III shows that the attacker does not have an obviously more effective counter `full` adaptation strategy. This hints that the `full` strategy may be a kind of equilibrium strategy because both attacker and defender have no significant gains by deviating from it. This inspires an important problem for future research: **Is the `full` adaptation strategy (or variant of it) an equilibrium strategy?**

Third, Table II shows that when $\mathsf{ST}_D = \mathsf{ST}_A$, the attacker can benefit by increasing its adaptiveness α. Table III exhibits the same phenomenon when $\mathsf{ST}_D \neq \mathsf{ST}_A$. In order to see the

TABLE II

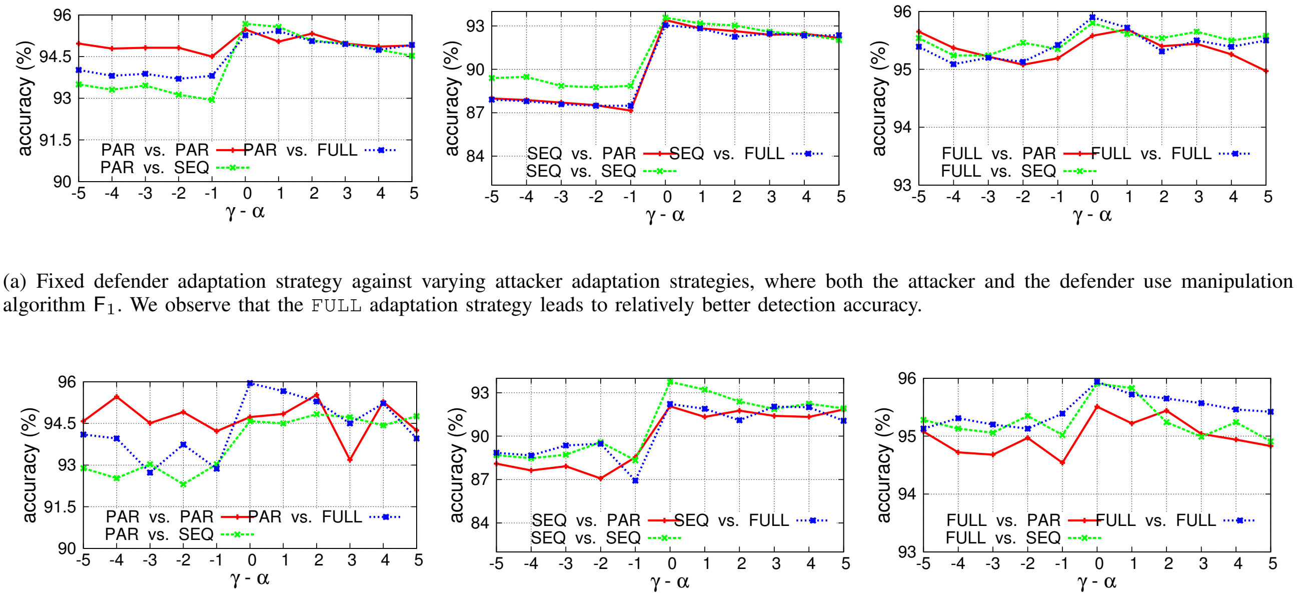
CROSS-LAYER PROACTIVE DETECTION WITH $\mathsf{ST}_A = \mathsf{ST}_D$. FOR BASELINE CASE $M_0(D_0)$, ACC = 99.68%, TRUE-POSITIVE RATE TP =99.21%, FALSE-NEGATIVE RATE FN=0.79%, AND FALSE-POSITIVE RATE FP=0.14%.

Strategy	Manipulation algorithm	$M_{0\text{-}8}(D_0)$				$M_{0\text{-}8}(D_1)$				$M_{0\text{-}8}(D_9)$			
		ACC	TP	FN	FP	ACC	TP	FN	FP	ACC	TP	FN	FP
$\mathsf{ST}_A = \mathsf{ST}_D$ $= \texttt{parallel}$	$\mathsf{F}_D = \mathsf{F}_1$ vs. $\mathsf{F}_A = \mathsf{F}_1$	99.59	99.71	0.29	0.39	95.58	92.03	7.97	3.62	95.39	92.00	8.00	3.83
	$\mathsf{F}_D = \mathsf{F}_1$ vs. $\mathsf{F}_A = \mathsf{F}_2$	99.27	99.77	0.23	0.77	78.51	25.50	74.50	9.88	78.11	32.18	67.82	11.48
	$\mathsf{F}_D = \mathsf{F}_2$ vs. $\mathsf{F}_A = \mathsf{F}_1$	99.16	99.76	0.24	0.93	76.33	19.32	80.68	11.17	78.96	39.77	60.23	12.14
	$\mathsf{F}_D = \mathsf{F}_2$ vs. $\mathsf{F}_A = \mathsf{F}_2$	99.59	99.62	0.38	0.39	93.66	90.25	9.75	5.59	96.17	92.77	7.23	3.08
$\mathsf{ST}_A = \mathsf{ST}_D$ $= \texttt{sequential}$	$\mathsf{F}_D = \mathsf{F}_1$ vs. $\mathsf{F}_A = \mathsf{F}_1$	99.52	99.69	0.31	0.45	93.44	77.48	22.52	3.05	92.04	59.33	30.67	2.99
	$\mathsf{F}_D = \mathsf{F}_1$ vs. $\mathsf{F}_A = \mathsf{F}_2$	99.23	99.70	0.30	0.82	74.24	20.88	79.22	14.06	79.43	30.03	69.97	9.38
	$\mathsf{F}_D = \mathsf{F}_2$ vs. $\mathsf{F}_A = \mathsf{F}_1$	99.27	99.67	0.33	0.80	77.14	29.03	70.97	12.33	82.72	40.93	59.07	7.83
	$\mathsf{F}_D = \mathsf{F}_2$ vs. $\mathsf{F}_A = \mathsf{F}_2$	99.52	99.53	0.47	0.50	93.44	78.70	21.30	2.10	92.04	62.30	37.70	2.11
$\mathsf{ST}_A = \mathsf{ST}_D$ $= \texttt{full}$	$\mathsf{F}_D = \mathsf{F}_1$ vs. $\mathsf{F}_A = \mathsf{F}_1$	99.68	99.44	0.56	0.20	96.92	96.32	3.68	2.89	95.73	92.03	7.97	3.27
	$\mathsf{F}_D = \mathsf{F}_1$ vs. $\mathsf{F}_A = \mathsf{F}_2$	99.27	99.58	0.42	0.72	85.68	40.32	59.68	4.38	78.11	29.99	70.01	11.00
	$\mathsf{F}_D = \mathsf{F}_2$ vs. $\mathsf{F}_A = \mathsf{F}_1$	99.60	99.66	0.34	0.40	85.65	51.84	48.16	6.93	87.61	72.99	27.01	9.01
	$\mathsf{F}_D = \mathsf{F}_2$ vs. $\mathsf{F}_A = \mathsf{F}_2$	99.68	99.60	0.40	0.28	96.92	95.60	4.40	2.88	95.73	90.09	9.91	2.83

TABLE III

PROACTIVE DETECTION AGAINST ADAPTIVE ATTACKS WITH $\mathsf{F}_D = \mathsf{F}_A$. FOR THE BASELINE CASE $M_0(D_0)$, WE HAVE ACC = 99.68%, TP =99.21%, FN=0.79%, FP=0.14%.

ST_D vs. ST_A	$M_{0\text{-}\gamma}(D_\alpha)$	$\mathsf{ST}_A = \texttt{parallel}$				$\mathsf{ST}_A = \texttt{sequential}$				$\mathsf{ST}_A = \texttt{full}$			
		ACC	TP	FN	FP	ACC	TP	FN	FP	ACC	TP	FN	FP
Manipulation algorithm $\mathsf{F}_D = \mathsf{F}_A = \mathsf{F}_1$													
$\mathsf{ST}_D = \texttt{parallel}$	$M_{0\text{-}8}(D_1)$	95.58	92.03	7.97	3.62	94.25	90.89	9.11	4.96	94.91	92.08	7.92	4.32
	$M_{0\text{-}8}(D_9)$	95.39	92.00	8.00	3.83	92.38	80.03	19.97	4.89	93.19	84.32	15.68	4.54
$\mathsf{ST}_D = \texttt{sequential}$	$M_{0\text{-}8}(D_1)$	92.15	74.22	25.78	3.93	93.44	77.48	22.52	3.05	92.79	76.32	23.68	3.07
	$M_{0\text{-}8}(D_9)$	89.20	58.39	41.61	4.11	92.04	59.33	30.67	2.99	88.42	57.89	42.11	3.91
$\mathsf{ST}_D = \texttt{full}$	$M_{0\text{-}8}(D_1)$	96.24	94.98	5.02	3.42	96.46	94.99	5.01	3.15	96.92	96.32	3.68	2.89
	$M_{0\text{-}8}(D_9)$	94.73	90.01	9.99	4.21	94.70	90.03	9.97	4.23	95.73	92.03	7.97	3.27
Manipulation algorithm $\mathsf{F}_D = \mathsf{F}_A = \mathsf{F}_2$													
$\mathsf{ST}_D = \texttt{parallel}$	$M_{0\text{-}8}(D_1)$	93.66	90.25	9.75	5.59	94.25	88.91	11.09	3.98	94.91	89.77	10.23	3.53
	$M_{0\text{-}8}(D_9)$	96.17	92.77	7.23	3.08	92.38	77.89	22.11	4.32	93.19	81.32	18.68	3.38
$\mathsf{ST}_A = \texttt{sequential}$	$M_{0\text{-}8}(D_1)$	90.86	70.98	29.02	4.82	93.44	78.70	21.30	2.10	92.79	72.32	27.68	4.02
	$M_{0\text{-}8}(D_9)$	88.43	53.32	46.68	3.97	92.04	62.30	37.70	2.11	88.42	57.88	42.12	3.17
$\mathsf{ST}_A = \texttt{full}$	$M_{0\text{-}8}(D_1)$	95.69	93.89	6.11	3.88	96.46	94.98	5.02	3.03	96.92	95.60	4.40	2.88
	$M_{0\text{-}8}(D_9)$	96.06	91.46	8.54	2.89	94.70	90.99	9.01	2.32	95.73	90.09	9.91	2.83



(a) Fixed defender adaptation strategy against varying attacker adaptation strategies, where both the attacker and the defender use manipulation algorithm F_1. We observe that the FULL adaptation strategy leads to relatively better detection accuracy.

(b) Fixed defender adaptation strategy against varying attacker adaptation strategies, where both the attacker and the defender use manipulation algorithm F_2. We observe that the FULL adaptation strategy leads to relatively better detection accuracy.

Fig. 4. Impact of defender's proactiveness γ vs. attacker's adaptiveness α on detection accuracy (average over the 40 days) under various "$\mathsf{ST}_D \times \mathsf{ST}_A$" combinations, where $\alpha \in [0, 8]$, $\gamma \in [0, 9]$, PAR, SEQ and FULL respectively stand for parallel, sequential and full adaptation strategy, "SEQ vs. APR" means $\mathsf{ST}_D = \texttt{sequential}$ and $\mathsf{ST}_A = \texttt{parallel}$ etc. Note that $\gamma - \alpha = a$ is averaged over all possible combinations of (α, γ) as long as $\alpha \in [0, 8]$ and $\gamma \in [0, 9]$, and that the detection accuracy is averaged over the 40 days. We observe that the detection accuracy in most cases there is a significant increase in detection accuracy when the defender's proactiveness matches the attacker's adaptiveness.

impact of defender's proactiveness as reflected by γ against the defender's adaptiveness as reflected by α, we plot in Figure 4 how the detection accuracy with respect to $(\gamma - \alpha)$ under the condition $\mathsf{F}_D = \mathsf{F}_A$ and under various $\mathsf{ST}_D \times \mathsf{ST}_A$ combinations. We observe that roughly speaking, as γ increases to exceed the attacker's adaptiveness α (i.e., γ changes from $\gamma < \alpha$ to $\gamma = \alpha$ to $\gamma > \alpha$), the detection accuracy may have a significant increase at $\gamma - \alpha = 0$. Moreover, we observe that when $\mathsf{ST}_D = \texttt{full}$, $\gamma - \alpha$ has no significant impact on the detection accuracy. This suggest that **the defender should always use the `full` adaptation strategy to alleviate the uncertainty about the attacker's adaptiveness α.**

V. Related Work

The problem of malicious websites detection has been an active research topic (e.g., [3], [5], [11]). The dynamic detection approach has been investigated in [7], [31], [4], [12]. The static detection approach has been investigated in [2], [6], [16]. The hybrid dynamic-static approach has been investigated in [3], [30], [9]. Loosely related to the problem of malicious websites detection are the detection of Phishing websites [16], [17], [10], the detection of spams [26], [28], [21], [16], [17], the detection of suspicious URLs embedded in twitter message streams [23], and the detection of browser-related attacks [25], [13]. However, none of these studies considered the issue of evasion by adaptive attackers.

The evasion attack is closely related to the problem of *adversarial machine learning*, where the attacker aims to evade an detection scheme that is derived from some machine learning method [1], [27]. Perdisci et al. [19] investigated how to make the detection harder to evade. Nelson et al. [18] assumed the attacker has black-box access to the detection mechanism. Dalvi et al. [8] used Game Theoretic method to study this problem in the setting of spam detection by assuming the attacker has access to the detection mechanism. Our model actually gives attacker more freedom because the attacker knows the data defender collected.

VI. Conclusion and Future Work

We formulated a model of adaptive attacks by which the attacker can manipulate the malicious websites to evade detection. We also formulated a model of proactive defense against the adaptive attacks. Experimental results based on a 40-day dataset showed that adaptive attacks can evade non-proactive defense, but can be effectively countered by proactive defense.

In the full version of the present paper, we will address *correlation constraints* between features, which is omitted due to space limitation. This study also introduces a set of interesting research problems, including: Are the same kinds of results/insights applicable to classifiers other than decision trees? Is the `full` adaptation strategy indeed a kind of equilibrium strategy? What is the optimal manipulation algorithm (if exists)? How can we precisely characterize the evadability caused by adaptive attacks in this context? What is the optimal time resolution at which the defender should proactively train its detection schemes (e.g., hour or day)?

Acknowledgement. This work was supported in part by ARO Grant #W911NF-13-1-0141. This study was approved by IRB.

References

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

[2] P. K. C. Seifert, I. Welch. Identification of malicious web pages with static heuristics. In *ATNAC 2008*, pages 91–96.

[3] D. Canali, M. Cova, G. Vigna, and C. Kruegel. Prophiler: a fast filter for the large-scale detection of malicious web pages. In *WWW'11*.

[4] K. Z. Chen, G. Gu, J. Nazario, X. Han, and J. Zhuge. WebPatrol: Automated collection and replay of web-based malware scenarios. In *ASIACCS'11*, 2011.

[5] H. Choi, B. B. Zhu, and H. Lee. Detecting malicious web links and identifying their attack types. In *WebApps'11*.

[6] P. K. C. U. A. Christian Seifert, Lan Welch and B. Endicott-Popovsky. Identification of malicious web pages through analysis of underlying dns and web server relationships. In *LCN 2008*, pages 935–941.

[7] M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious javascript code. In *WWW'10*.

[8] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *KDD'04*, pages 99–108, 2004.

[9] B. Eshete. Effective analysis, characterization, and detection of malicious web pages. WWW '13, pages 355–360.

[10] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In *WORM'07*.

[11] L. Invernizzi, S. Benvenuti, M. Cova, P. M. Comparetti, C. Kruegel, and G. Vigna. Evilseed: A guided approach to finding malicious web pages. In *S&P'12*, pages 428–442.

[12] A. Kapravelos, Y. Shoshitaishvili, M. Cova, C. Kruegel, and G. Vigna. Revolver: An Automated Approach to the Detection of Evasive Web-based Malware. In *USENIX Security*, 2013.

[13] G. Y. Lei Liu, Xinwen Zhang and S. Chen. Chrome extensions: Threat analysis and countermeasures. In *NDSS'13*.

[14] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang. Knowing your enemy: Understanding and detecting malicious web advertising. In *CCS'12*.

[15] C. Ludl, S. Mcallister, E. Kirda, and C. Kruegel. On the effectiveness of techniques to detect phishing sites. In *DIMVA'07*.

[16] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *KDD'09*.

[17] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious urls: an application of large-scale online learning. In *ICML'09*.

[18] B. Nelson, B. I. P. Rubinstein, L. Huang, A. D. Joseph, and J. D. Tygar. Classifier evasion: models and open problems. In *ECML PKDD'11*.

[19] R. Perdisci, G. Gu, and W. Lee. Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In *ICDM'06*, pages 488–498.

[20] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All your iframes point to us. In *USENIX Security*, 2008.

[21] Z. Qian, Z. M. Mao, Y. Xie, and F. Yu. On network-level clusters for spam detection. In *NDSS'10*.

[22] R. Quinlan. *C4.5:Programs for Machine Learning*. 1993.

[23] J. K. Sangho Leey. Warningbird: Detecting suspicious urls in twitter stream. In *NDSS'12*.

[24] C. Seifert and R. Steenson. Capture - Honeypot Client (Capture-HPC). https://projects.honeynet.org/capture-hpc, 2006.

[25] V. S. Sooel Son. The postman always rings twice: Attacking and defending postmessage in html5websites. In *NDSS'13*.

[26] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. Design and Evaluation of a Real-Time URL Spam Filtering Service. In *S&P'11*.

[27] S. Venkataraman, A. Blum, and D. Song. Limits of learning-based signature generation with adversaries. In *NDSS'08*.

[28] C. Whittaker, B. Ryner, and M. Nazif. Large-scale automatic classification of phishing pages. In *NDSS'10*.

[29] G. Xiang, J. Hong, C. P. Rose, and L. Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Trans. Inf. Syst. Secur.*, 2011.

[30] L. Xu, Z. Zhan, S. Xu, and K. Ye. Cross-layer detection of malicious websites. In *ACM CODASPY'13*, pages 141–152.

[31] J. Zhang, C. Seifert, J. W. Stokes, and W. Lee. Arrow: Generating signatures to detect drive-by downloads. In *WWW'11*, pages 187–196.