JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 36, 597-619 (2020)
DOI: 10.6688/JISE.202005_36(3).0008

A Robust Algorithm for Predicting Attacks Using Collaborative Security Logs

AMIR REZAPOUR AND WEN-GUEY TZENG
Department of Computer Science
National Chiao Tung University
Hsinchu, 30010 Taiwan
E-mail: {rezapour; wgtzeng}@cs.nctu.edu.tw

As networks become ubiquitous in our daily lives, users rely more on networks for exchanging data and communication. However, numerous new and sophisticated attacks that endanger security of users have been reported. In practice, blacklisting illicit sources has been a fundamental defense strategy in recent years. In this paper, we propose a predictor that is based on the observations from a centralized log-sharing infrastructure. Our observations include the direct relation between attackers and victims, victim similarities, and attacker correlations. We compile a customized blacklist for each *Dshield.org* contributor using a weighted function of direct and indirect relations between victims and attackers. This list not only offers a significantly higher prediction ratio, but also includes source addresses with potentially higher threats. We evaluate our predictor using two months of malicious activities acquired from *Dshield.org*. The experimental results demonstrate a significant improvement over previous algorithms.

Keywords: prediction algorithms, IP blacklisting, network security, association rule mining, data mining, machine learning

1. INTRODUCTION

Developing network-based intrusion detection systems (NIDS) has become common practice to respond to Internet security concerns. Generally, the goals of NIDSs are to detect attacks and prevent potentially harmful interaction with an entity from taking place. Several NIDS approaches have been proposed for providing the first level of protection. As a result, network administrators frequently utilize security logs produced by NIDSs to prevent malicious activities from traveling in and out their networks. Yet, further analysis of the collaborative security logs contributed by various NIDSs worldwide can provide a secondary level of protection against broader attacks.

Our work makes use of a central repository, such as *Dshield.org*, of shared security logs from NIDS or network firewalls of victims all over the Internet. The intention for sharing security logs is to help produce better prediction of future malicious activities. In practice, such data have been used to compile a blacklist: a collection of source IP addresses that are suspected to be involved in illicit and malicious activities. For instance, *Dshield.org* [1] regularly processes NIDS logs contributed by thousands of victim networks worldwide and publishes the most prolific attack sources seen by its contributors.

Blacklists need to be frequently updated to keep pace with the attempt of attackers who utilize various IPs in various attack phases. Blacklisting all offenders is not an option. Firstly, such a long blacklist creates a nuisance for administrators and may cause an

Received April 29, 2018; accepted May 23, 2018.
Communicated by Hung-Min Sun.

unacceptable delay in the network. Secondly, not necessarily all offenders worldwide have the same interest in all networks. This is coupled with the fact that each network has a different payoff for offenders to attack. Therefore, a customized blacklist needs to be compiled for each individual with respect to its similar victims and attack history.

In recent years, a large number of organizations have adopted blacklists to combat attackers. These blacklists include the source IPs of spam senders[1] and malicious web pages[2]. Earlier research about blacklisting can be categorized into two types: Local Worst Offensive List (LWOL) and Global Worst Offensive List (GWOL) [2]. GWOL leverages large online repositories (*e.g. Dshield.org*) and offers a list of highly prolific and global attackers. Such an approach may not necessarily deliver a valuable service to victims. The reason is that the highly prolific attackers reported in security logs may be irrelevant to a victim's local network. In some cases GWOL may cause a high miss rate for those attackers who choose their targets more strategically. Such attackers may target just a few networks and hence are not prolific at all [3]. In LWOL, each individual victim creates its own history of malicious activities. LWOL that uses a private security logs may fail to predict the malicious activities that have never attacked a certain network in advance. Therefore, LWOL is reactive in the sense that it cannot predict attack sources before these sources reach a network.

From the above discussion, a high quality prediction algorithm should be coupled with similarity among victims, in addition to correlation among attackers. Furthermore, predictions should be proactive – that is, they should incorporate those IP sources even when they have not been seen previously in the victim network. Our goal is to deliver a better prediction algorithm with higher prediction ratios[3] by uncovering the correlations among victims and attackers. We study prediction of malicious activities at the IP level for each individual victim using the information of attacker IP's, victim networks and time stamps of malicious activities in the logs. Our prediction algorithm integrates a novel time-based logistic regression model to address direct interaction between an attacker and a victim on evolution over time basis. We use similarity analysis to assign weights to victims and attackers based on their closeness. Finally, we use a weighted function of direct and indirect relations among victims and attackers to compile a customized blacklist for each *Dshield.org* contributor.

Our work improves upon the results [4] (SLM). We evaluate our algorithm using a one-month data October 2008 (*DS*1) and a recent one October 2016 (*DS*2) from *Dshield.org*. *DS*1 is used to compare with previous results and *DS*2 shows that our algorithm still works today. Our experimental results indicate that for all of contributors, our blacklist has significantly higher prediction ratios on all testing days. Over *DS*1, our pre- diction ratios are higher than 70% on most days, while the prediction ratios of the SLM algorithm are all lower than 40%. Overall, our average prediction ratio is 62%, which is almost twice that of the SLM algorithm. Over *DS*2, our average prediction ratio is 64.7%, which is 1.3 times higher that of the SLM algorithm.

We measure robustness of our algorithm against noises with respect to noise percentages and the number of fake contributors. The prediction ratio decreases almost linearly as the percentage of noisy reports increases. Moreover, the prediction ratio decreases slightly as the number of fake contributors increases.

---

[1] SpamCop
[2] PhishTank, SafeBrowsing
[3] We define prediction ratio as a ratio of the hit count over the total number of attacks.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents some preliminary notations and formulates the prediction problem. Section 4 covers our prediction algorithm. Section 5 presents an experimental evaluation of our algorithm using a real-world dataset of security logs. Finally, the paper concludes in Section 6.

2. RELATED WORK

Recently a number of research papers have proposed algorithms to generate blacklists against specified attacks. For instance, spam constitutes a noticeable portion of attacks that users directly experience. Therefore, some researchers have attempted to solve the phishing problem at the e-mail level by generating a blacklist of spammers [5-7]. In a slightly different attempt, some companies (*e.g.* Google, Microsoft and McAfee) scour millions of web pages to identify and maintain blacklists of malicious URLs.

Dshield.org is a centralized repository providing a daily malicious activity collected from a large number of contributors all over the Internet. *Dshield.org* dataset consists of a set of contributors each has its own share of information. Each contributor can communicate with others and also provide its information to a central authority. Contributors can choose to submit their logs without a validation technique at the expense of anonymity. Therefore, a *Sybil* attack may occur when an adversary introduces many counterfeit identities corresponding to a single contributor. Using *Sybil* contributors, an adversary may provide bogus reports for his benefits which reduce the utility of *Dshield.org* dataset. The current research to deal with *Sybil* attacks [8-13] has gone into the study of trust relationships in social networks to reduce the influence of *Sybil* attacks. Our method differs from *Sybil* attacks detection schemes. It aims to neither detect the *faulty* contributors nor locate the *correct* contributors. Nevertheless, *Sybil* attack detection schemes can be applied as a preprocessing module in order to sanitize the *Dshield.org* dataset.

Some recent research analyzes social network relationships to identify *faulty* entities, in particular, the attackers in the *Sybil* attack. In the *Sybil* attack, a single entity controls multiple identities in order to defeat security mechanisms and attack its users [14]. Some defense mechanisms [8-13] have been developed and evaluated through trust relationships in social networks. They identify *faulty* entities (attackers) by exploring direct and indirect social-network relationships among entities. Nevertheless, they all assume existence of *static* social networks, the trust relationships remain unchanged after establishment. We make no such assumption. Our prediction model can deal with temporal dynamics and relationships among victims and attackers.

Another line of research aims to improve the quality of the blacklisting algorithm with a broader perspective. Zhang *et al.* [2] (HPB) argued a multipurpose blacklist of malicious source IPs that are predicted to be harmful for an organization. They improved IP level blacklisting by compiling a compact and customized blacklist which is more likely to be relevant to a victim. The main idea is that victims can share their security logs and similarity is defined as the number of common attackers. They used a relevance ranking process to propagate the likelihood of the given attacker to attack other victims, given a correlation attack graph. To this end, they represented the correlation as a graph that captures the relationships between victims. Finally, they utilized a severity assessment algorithm to compile the final blacklist.

Soldo *et al.* [4] extended the HPB results by reformulating the problem as a recommendation system. They included a temporal consideration in the prediction algorithm and quantified the direct relations between attackers and victims as a ground truth. Furthermore, they developed a variation of Pearson similarity to measure the victim similarity. That is, when victims share attacks from the same source address within the same time intervals, they are more related to each other than the victims sharing common attackers but from different time intervals. Therefore, giving higher priority to attacks occurring within the same time interval captures a stronger correlation among victims. Additionally, they clustered attackers and victims jointly in order to discover the strength of correlation among the attackers and victims within the group. Clusters are regarded as regions in the attacker-victim graph in which the nodes are dense. A highly dense region resembles a complete attacker-victim graph, which indicating strong correlation. That is, the denser a region is, the more probable that its attackers will attack its victims.

Zhang *et al.* [2] and SLM focused on blacklisting at the IP level by finding correlations among victims and attackers from shared security logs. They differ in both modeling and underlying techniques. The experimental results of SLM completely dominate the prediction ratios of previous work, including HPB, LOWL and GOWL. Thus, we compare our experimental results with the previous result (SLM) [4].

We use a novel temporal predictor function with an exponentially updating parameter to reflect direct and timely relation between an attacker and a victim. In addition, we tackle the victim/attacker similarity problem by the method of association rule mining. Our technique is very efficient at unscrambling the similarities.

Finally, there are studies loosely related to ours, such as predicting cyber attack rates [15, 16], Internet background radiation [17, 18] and one-way traffic [19]. Predicting cyber attacks rates aims to predict the intensity of cyber attacks against a target. This prediction enables the defender to efficiently allocate defense resources on demand to cope with the extraordinary large cyber attack rates. The other two studies propose to classify the data as scanning, peer-to-peer applications, misconfigurations, worms *etc.* and do not provide a prediction.

3. PROBLEM DEFINITION

3.1 Notation

An uppercase boldface letter represents matrix B of $n \times m$ dimension. A lowercase boldface letter denotes a vector $\mathbf{x} = (x_1, x_2, \ldots, x_m)$. $\mathbf{B}^\top$ and $\mathbf{x}^\top$ are the transpose matrices of **B** and **x**. Let A denote the set of attackers and V the set of victims. The number of elements in a set is denoted by $|A|$. t indicates the time that an attack was reported in the log files. We treat a log dataset, such as the *Dshield.org* dataset, as a set $R = \{(a, v, t) \mid a \in A, v \in V, t \in T\}$, where the tuple (a, v, t) represents that attacker $a \in A$ attacked $v \in V$ at time $t \in T$. The attacks occur non-uniformly over time. Hence, we chop the dataset according to small time intervals of size Δt, where Δt could be an hour or a day. Let τ_i be the starting time of the ith time interval and $\Delta_i = [\tau_i, \tau_i + \Delta t]$ the ith time interval, $i \geq 1$. Let $r_{a,v}(\tau_i)$ represent the number of attacks from attacker a to victim v within time interval $[\tau_i, \tau_i + \Delta t]$. The $|A| \times |V|$-dimensional binary matrix $\mathbf{B}_i$ indicates occurrence of illicit activities in the ith time interval Δ_i. That is, the (a, v)-entry $b_i(a, v) = 1$ if some $(a, v, t) \in R$ and $t \in [\tau_i, \tau_i + \Delta t]$, and $b_i(a, v) = 0$, otherwise.

3.2 Prediction

We are interested in finding the most probable attackers that aim to harm victim v within the time interval $\Delta_{\tau+1}$ or later, where τ denote the latest time interval[4]. Thus, given w as the window size, *training window* T_{train} is $\{\mathbf{B}_{\tau-w+1}, \ldots, \mathbf{B}_{\tau-1}, \mathbf{B}_{\tau}\}$ and the *testing window* T_{test} is $\{\mathbf{B}_{\tau+1}, \ldots\}$ for verifying the results of the prediction algorithm.

The prediction algorithm generates a blacklist $BL(v) = \{a_1, a_2, ..., a_N\} \subset A$ that contains top N probable attackers to victim v in time interval $\Delta_{\tau+1}$. Given such a blacklist, we use the *hit count* as the number of correctly predicted attackers reported in T_{test} to measure effectiveness of the prediction algorithm.

3.3 Association Rule Mining

Association rules mining (ARM) is developed to analyze the transactional sale data. The goal of ARM is to locate the items that are frequently bought together in the same transaction. The sets of items are known as itemsets. The goal of our analysis is to find the similarity among attackers and victims and capture them as rules. Hence, throughout this paper, items are victims/attackers.

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of m items. Let $D = \{tid_1, tid_2, \ldots, tid_n\}$ denote a database of n transactions, where $tid_i \subseteq I$. A transaction $tid \in D$ is said to contain an itemset $I \subset I$ if $I \subseteq tid$.

1. The *support* of an itemset $I \subset I$ is defined as $support(I) = |\{tid_i | I \subseteq tid_i, tid_i \in D\}| / |\{tid_i \in D\}|$.
2. A *rule* $I_1 \Rightarrow I_2$ is defined as a conditional implication among itemsets, where itemsets $I_1, I_2 \subset I$ and $I_1 \cap I_2 = \phi$.
3. The *support* of a rule $I_1 \Rightarrow I_2$ is the percentage of transactions in D containing $I_1 \cup I_2$.
4. The *confidence* of an association rule r: $I_1 \Rightarrow I_2$ is the conditional probability that a transaction contains I_2, given that it contains I_1. Thus, $confidence(r) = support(I_1 \cup I_2) / support(I_1)$.
5. The *lift* of a rule is defined as: $lift(I_1 \Rightarrow I_2) = support(I_1 \cup I_2) / support(I_1) \times support(I_2)$. *lift* assesses the rule $I_1 \Rightarrow I_2$ in terms of applicability and relevance. A $lift(I_1 \Rightarrow I_2) > 1$ implies that the probability of occurrence of I_1 and I_2 are positively dependent on one another. If $lift(I_1 \Rightarrow I_2) < 1$, then I_1 and I_2 appear less often together than expected. A value of 1 implies that I_1 and I_2 are independent and the occurrence of I_1 has almost no effect on the occurrence of I_2.

Given a transactional database D, the objective of ARM is to extract the rules of form

$I_1 \Rightarrow I_2$[support, confidence, lift]

satisfying user-defined thresholds for the admissible minimum support, confidence and lift. Eventually, we use the *lift* metric to discard weak rules. For further details, we refer interested readers to [20].

We design our ARM based on the Apriori algorithm [21]. It identifies all frequent itemsets of two items. Hereafter, we use our ARM algorithm for mining association rules.

[4] Intervals before $\Delta_{\tau+1}$ are used for training.

4. MODEL OVERVIEW

We define the prediction function of estimating the possibility that attacker $a \in A_{train}$ will attack $v \in V_{train}$ in time interval $\Delta_{\tau+1}$ as follows:

$$P_{a,v}(\Delta_{\tau+1}) : \boldsymbol{B}_{\tau-w+1}, \ldots, \boldsymbol{B}_{\tau} \rightarrow [0,1] \in \mathbb{R} \tag{1}$$

where A_{train} and V_{train} are the sets of attackers and victims in T_{train}, respectively. We include all attackers and victims into our computation since we observe that any small correlation among attackers and victims is not a negligible phenomenon.

In the rest of this section, we investigate the relations between attackers and victims including direct relation, victim similarity and attacker correlation. For each case, we observe the data and design a prediction method.

4.1 Temporal Attack Predictor

We begin by quantifying the likelihood that an attacker $a \in A$ attacks a victim $v \in V$. Each $b_i(a, v)$ shows the interaction between the given attacker a and the victim v within time interval Δ_i. For every pair (a, v), we extract a signal from the training data:

$$b_1(a, v), b_2(a, v), \ldots, b_n(a, v) \tag{2}$$

that shows its evolution over time.

Observation. By the *Dshield.org* dataset, an attacker usually has two attack strategies. One is to attack only once and the other is to attack multiple times. For the latter strategy, the attacks are more likely to happen within a small time period. The data shows that the majority of consecutive attacks happen within a time period between 3 to 10 minutes and the rest scatter throughout the rest of the day [22].

This observation indicates that the prediction should mostly rely on recent past activities. Thus, the recent past activities are more important than the old ones. Furthermore, the attacks vary over time as the number of reports submitted by victims vary on different days. Hence, the evolution of attacks over time is also an important phenomenon [4]. Consequently, Eq. (2) accounts for both the time when an attack was reported and its evolution over time.

Design. We use a parametric predictor to capture the aforementioned observation. The prediction problem is to estimate from past information the one-step conditional probability

$$P(b_{\tau+1}(a,v) = 1 \mid b_{\tau}(a,v), b_{\tau-1}(a,v), \ldots). \tag{3}$$

We wish to approximate the conditional probability at time $\tau+1$ by the past w observations with a predictor function h that yields the lowest error when predicting $b_{\tau+1}(a, v)$. Let $T_{a,v}(\tau, w) = (b_{\tau-w+1}(a, v), b_{\tau-w+2}(a, v), \ldots, b_{\tau}(a, v))$ denote a sequence of w observations known to the observer at time τ. We define predictor function h as

$$h(T_{a,v}(\tau, w)). \tag{4}$$

Algorithm 1 shows the description of predictor function h, where σ is the sigmoid function and γ is a discount parameter. The idea is to update the model parameter θ in order to reduce the error for the next prediction. Our formula makes the update rate as an exponentially decreasing function. Therefore, the last iterations cause large changes in the parameter, while the first ones do only fine-tuning.

It begins by initializing the prediction parameter $\boldsymbol{\theta} = [\theta_1\ \theta_0]$ with θ_0 as the scalar bias term. In each iteration (lines 5 to 6), it uses $x_{i\text{-}1}$ and current parameter $\boldsymbol{\theta}$ to compute $\sigma(z)$ the next prediction x_i. The value $((\sigma(z) - x_i)\cdot\nabla_{\boldsymbol{\theta}}\,\sigma(z)\cdot[x_{i\text{-}1}\ 1])$ updates $\boldsymbol{\theta}$ according to the gradient of the error w.r.t. the single training instance $x_{i\text{-}1}$. The magnitude of the update is proportional to the gradient term $\nabla_{\boldsymbol{\theta}}\ \sigma(z)\cdot[x_{i\text{-}1}\ 1]$. For instance, if we are encountering a training example on which our prediction does not match the actual value of x_i, *i.e.* $(\sigma(z) - x_i) \neq 0$, then we update $\boldsymbol{\theta}$ proportionally to the gradient term. Furthermore, the exponentially decreasing function $\gamma(1-\gamma)^{t\text{-}i\text{-}1}$ associates an adaptive update rate to the overall change of parameter $\boldsymbol{\theta}$. The significant advantage of our predictor is that the prediction is more dependent on recent past observations than the past ones. For old data, *i.e.* $i = 1$, $\gamma(1-\gamma)^{t\text{-}i\text{-}1} \simeq 0$, it assigns a small update rate, and for the recent data, *i.e.* $i = t-1$, $\gamma(1-\gamma)^{t\text{-}i\text{-}1} = \gamma$, it assigns a large update rate to the prediction parameter $\boldsymbol{\theta}$. Therefore, the future activity strongly relies upon the recent past activities. Finally, Algorithm 1 uses the optimal prediction parameter and the latest observation x_t to approximate the conditional probability (3). The value $h(.)$ is the probability that the attacker a will attack the victim v in time interval $\Delta_{\tau+1}$.

Algorithm1: Predictor Function h

```
1  Function h(x1, x2, ..., xt)
2      θ = [θ1 θ0].Initialize();
3      x0 = 1
4      for i = 1 to t−1 do
5          z = θ^T·[xi−1 1]
6          θ ← θ − γ(1 − γ)^(t-i-1)·((σ(z) − xi)·∇θ σ = (z)·[xi-1 1])
7      end
8  return σ(θ^T·[xt 1]);
```

Another issue is to initialize γ. We choose γ in the range [0.5 0.8]. If γ is close to 1, the recent attacks aggressively change $\boldsymbol{\theta}$. The initialization value for $\boldsymbol{\theta}$ doesn't have any significant impact. If γ is close to 0, it acts as a smoothing factor. We empirically investigate the range of γ to support our observations.

$h(.)$ is a LWOL approach that captures the attacker in its active time by observing recent past activities. It complies with the observation that a large number of attackers are highly active within 1 to 3 days, and during this period they conduct many attacks. In fact, there is a correlation between the active time and the number of attacks [22].

Moreover, our predictor quickly excludes inactive attackers after observing the recent past activity of the attackers. We remove inactive attackers from the blacklist quickly and fill the blacklist with active attackers. This guarantees that there is sufficient space for putting more serious attackers on the list since the length of blacklist is fixed.

4.2 Victim Similarity

The prediction solely based on Eq. (4) cannot capture the correlation among attackers (or victims). A persistent attacker can frequently switch its targets and evade the level of prediction by Eq. (4). Hence, we introduce victim similarity (N) and attacker correlation (C).

The neighborhood model is an effective way to predict user behavior in recommendation systems [23]. The idea is to estimate the likelihood of a victim v being attacked using other victims, called neighbors. Such victims build a group called neighborhood. A victim gets prediction about those attackers that have not been reported before, but were already reported by victims in its neighborhood. That is, when a victim v reports an illicit activity, the neighborhood model propagates the effect of the attack to similar victims of v.

Observation. An analysis on the number of common attackers between a victim and its neighbors in the *Dshield.org* dataset illustrates that similar victims do share some common attackers. Majority of similar victims share 20 to 60 common attackers [22]. This is because similar victims might have identical benefit for an attacker or share the same vulnerability. When a victim shares common attackers with another victim, it is more likely for both victims to receive attacks from the same attacker within a very short period of time. In other words, when attacker a attacks many victims, it likely attacks them at about the same time [24].

Definition 1: We define victims v and u to be similar (close) if they both report an attack within some time intervals Δ_i. The degree of their similarity (closeness) is measured by a conditional probability $P(u|v) = p(u, v)/p(v)$, where the joint probability $p(u, v)$ is the $p(v)$ probability that both v and u report illicit activities within some intervals over the number of intervals. $p(v)$ is the marginal probability of v.

Design. We wish to estimate the probability of victim v being attacked, given that its neighbors were attacked. Let the set $N_v = \{v_1, v_2, \ldots, v_k\}$ contain similar victims of v. We compute the corresponding conditional probabilities as:

$$P(v \mid v_i) = Confidence(v_i \Rightarrow v), i = 1, 2, .., k \tag{5}$$

where $P(v|v_i)$ as in Definition 1 is the probability of victim v to be threatened given that its neighbor v_i was attacked.

The problem of finding similar victims can be solved by the association rule mining. A similarity algorithm constructs a function that maps a pair of objects x and y to a number in [0, 1]. This function measures the degree of similarity between x and y. In terms of association rules, the objects can be treated as items. We find similar victims as follows.

We prepare a transactional database TD. A transaction has a unique transaction id *tid* and a list of items. Consider T_{train} with a window size of length w. There are w binary matrices $\mathbf{B}_{\tau-w+1}, \ldots, \mathbf{B}_{\tau-1}, \mathbf{B}_{\tau}$. A transaction $\langle tid_i, v_1, v_2, ..., v_l \rangle$ associated with a tid_i is a list of victims that reported at least one attack in $\mathbf{B}_i$. TD is the set of all such transactions. This captures our intuition that victims that have been attacked within the same time intervals are more similar to each other.

We invoke the ARM algorithm with input TD, *min_support*, *min_confidence*, and *min_lift* to determine similar victims N_v. In the basket mining problem, the generated rules

$$v_1 \Rightarrow v\,[\text{Support}(v_1 \Rightarrow v), \text{Confidence}(v_1 \Rightarrow v), \text{Lift}(v_1 \Rightarrow v)]$$
$$\ldots$$
$$v_k \Rightarrow v\,[\text{Support}(v_k \Rightarrow v), \text{Confidence}(v_k \Rightarrow v), \text{Lift}(v_k \Rightarrow v)]$$

indicate that item v should be recommended to the customer by observing the set $\{v_1, v_2, ..., v_k\}$ in the basket. In our reasoning, $N_v = \{v_1, v_2, ..., v_k\}$ retrieves the most frequent victims that were attacked at about the same time as victim v. Therefore, $\{v_1, v_2, \ldots, v_k\}$ are treated as similar victims of the given victim v. Moreover, Confidence($v_i \Rightarrow v$) as in Eq. (5) estimates the conditional probability of v being attacked given that its neighbor v_i was attacked.

We calculate the neighborhood influence of victim v as follows:

$$N_{a,v}(\Delta_{\tau+1}) = \frac{\sum_{i=1}^{k} Confidence(v_i \Rightarrow v) \cdot h(T_{a,v_i}(\tau, w))}{\sum_{i=1}^{k} Confidence(v_i \Rightarrow v)}. \quad (6)$$

Notice that if v and v_i are similar victims, Confidence($v_i \Rightarrow v$) outputs a higher value for stronger neighborhood influence. $h(T_{a,vi}(\tau, w))$ estimates the probability that neighbor v_i will be attacked by attacker a in time interval $\Delta_{\tau+1}$. Malicious activities fluctuate over time. Eq. (6) models the intuition that victims receiving an attack from the same source within the same period of time are more similar to each other. This implies that victim similarity not only depends on the number of common attackers, but also on the time interval in which the malicious activities are reported. In this case, they are more likely influenced by the same type of attacks. Intuitively, Eq. (6) combines LWOL ($h(T_{a,vi}(\tau, w))$) and GWOL (Confidence($v_i \Rightarrow v$)) approaches. That is, Eq. (6) proactively incorporates those attackers even when they have not been seen previously in the victim network, but are determined to be relevant to the victim network through its neighbors.

4.3 Attacker Correlation

Consider a *DDoS* attack launched by a botmaster to attack a victim, the botmaster dedicates a subset of bots under its control to scan and find vulnerability of the target. Later, the botmaster may command some previously-unseen bots (IPs) to execute the real attack [25, 26]. Therefore, identifying similarity among attackers, even just discovering a few of them, privileges the victim to preemptively include other similar attackers into his blacklist.

Another scenario is that a newly joined victim v' that reported a few malicious activities does not have similar victims in the neighborhood model. The new user problem (a.k.a *cold start*) is one of main difficulties faced by recommendation systems. Since there are not yet enough ratings, the system fails to identify neighbors of the given user. There are a number of solutions discussed in [23]. One effective solution is to use a hybrid technique of combining user-based (*i.e.*, victim similarity) and item-based (*i.e.*, at-

tacker correlation) predictions. Therefore, leveraging attacker correlation not only includes similar attackers into the blacklist preemptively, but also solves the cold start problem.

Observation. In analysis of the *Dshield.org* dataset, Soldo [22] found that a bulk of attackers only harm one victim. Therefore, it is beneficial to explore correlation among attackers.

Soldo studied temporal dynamics of malicious source IPs within one month and observed that only 13% of the IPs are reported as malicious within two consecutive days. This number decreases rapidly to 8% for 3 consecutive days. Overall, only 4% of source IPs are continuously reported as malicious for the whole month. Furthermore, bots are programmed to expand their population by transferring an infected code to other machines. Shin *et al.* [27] showed that it is more likely for a bot to infect another machine within the same subnet.

The distribution of (attacker, victim) pairs over a time period shows the following issues. Firstly, attackers are not evenly distributed in network classes. For instance, fewer attackers are from network class A, which mainly belongs to governmental organizations and large IT companies. Secondly, many new attackers are within previously seen subnets. This is likely due to DHCP configuration that assigns a new IP for the same machine at different times.

In order to capture a higher level of abstraction for attacker correlation, we inspect malicious activities from the subnet level, instead of individual IPs. We tag risky subnets and measure the critical level of a given subnet by malicious activities reported in the dataset.

Design. The key idea of attacker correlation is to estimate the likelihood of a possible attack from similar attackers to those that are reported by victim v. There are two types of victims as presented below:

If victim v is an active contributor of *Dshield.org*, then given a set of attackers $A_v = \{a_1, a_2, \ldots, a_l\}$ reported by victim v, we define two tiers of correlated subnets. The first tier is determined directly from A as $S^{(1)} = \{s_1, s_2, ..., s_l\}$, where s_i is the subnet of a_i. The second tier consists of all relevant subnets that are similar to those in the first tier, *i.e.*

$$S_v^{(2)} = \{s \mid Confidence(s_i \Rightarrow s) > \min\_confidence\}_{\forall s_i \in S_v^{(1)}}. \tag{7}$$

Let $S_v = S^{(1)} \cup S^{(2)}$ be the set of all such direct and indirect subnets.

Otherwise, if victim v is a newly join victim (*i.e.*, $S_v = \phi$), we investigate the attacker correlation over all attackers in A_{train} by setting $S_v = \{s|s = \text{subnet}(a), a \in A_{train}\}$. That is, in the absence of security logs, the attacker correlation model is a GWOL technique that incorporates highly prolific and global attackers. As a result, our algorithm tackles the *cold start* problem by predicting potential global attackers using the attacker correlation model.

Upon receiving a request to estimate the probability that the given attacker a (with s = subnet(a)) harms victim v, if $s \notin S_v$, the attacker correlation algorithm halts and outputs zero. Otherwise, we inspect the correlated subnets $S_v = \{s_j|\text{Confidence}(s_j \Rightarrow s) > \textit{min-confidence}\}$ for a possible threat. The correlation between subnets is formulated as a conditional probability:

$$P(s \mid s_j) = Confidence(s_j \Rightarrow s), j = 1, 2, .., m. \tag{8}$$

Let $P_{sub}(s_i)$ and $P_{att}(a)$ denote the threat levels of subnet s_i and attacker a, respectively. $P_{sub}(s_i)$ is defined as the number of malicious activities originated from subnet s_i over all malicious activities reported in T_{train}. Similarly, $P_{att}(a)$ is the number of malicious activities originated from attacker a over all malicious activities reported in T_{train}. $P_{att}(a)$ discards those IPs that are used in less prolific networks such as class A.

We calculate the attacker correlation influence of attacker a as:

$$C_{a,v}(\Delta_{\tau+1}) = \begin{cases} P_{att}(a) + \dfrac{\sum_{s_i \in S_v} Confidence(s_i \Rightarrow s) \cdot P_{sub}(s_i)}{\sum_{s_i \in S_v} Confidence(s_i \Rightarrow s)}, & \text{if}(subnet(a) \in S_v) \\ 0, & \text{otherwise} \end{cases}. \tag{9}$$

Notice that attacker correlation model locates the potential attackers only based on the attacker activities and the values are not restricted for a specific victim.

4.4 Combining the Predictors

There are many ways to combine predictors, such as averaging. Our empirical analysis leads us to use weighted predictor, similar to the one in [4]. We embed weights to estimate the prediction Eq. (1) as

$$P_{a,v}(\Delta_{\tau+1}) = h(T_{a,v}(\tau, w)) + W^{ns}_{a,v} \cdot N_{a,v}(\Delta_{\tau+1}) + W^{ca}_{a,v} \cdot C_{a,v}(\Delta_{\tau+1}) \tag{10}$$

where the weight of victim similarity is

$$W^{ns}_{a,v} = \frac{N_{a,v}(\Delta_{\tau+1})}{N_{a,v}(\Delta_{\tau+1}) + \lambda_1}. \tag{11}$$

λ_1 is a parameter in range [0, 1]. $N_{a,v}(\Delta_{\tau+1})$ is larger for a victim v with more similar neighbors. The more similar neighbors the victim v has, the larger $W^{ns}_{a,v}$ is. This increases the effect of the neighborhood model by assigning higher importance for $N_{a,v}(\Delta_{\tau+1})$ in Eq. (10).

Likewise, we define the weight for attacker correlation as

$$W^{ca}_{a,v} = \frac{C_{a,v}(\Delta_{\tau+1})}{C_{a,v}(\Delta_{\tau+1}) + \lambda_2}. \tag{9}$$

Where $0 \leq \lambda_2 \leq 1$. The stronger attacker relation the attacker a has, the larger $W_{a,v}{}^{ca}$ is.

4.5 Compiling Blacklists

For the final blacklist, we include more severe attackers which not only have been reported to be prolific, but also have been discovered to be more relevant to the given victim. We use the predictor Eq. (10) to generate an ordered candidate list and pick top N ranked attackers.

5. PERFORMANCE EVALUATION

5.1 The Dataset

Dshield.org is a repository of firewalls and NIDS logs collected from a large number of contributors all over the Internet. Every time an alarm is raised by a contributor's network, the contributor submits a log to the *Dshield.org* repository. The log contains Contributor ID, target port, source IP, source port, Protocol ID, and time stamp, with source IP referring to an attacker and Contributor ID referring to a victim.

Two datasets DS_1 and DS_2 from *Dshield.org* are involved in evaluation of our algorithm. Dataset DS_1 consists of one month data (October 2008) that was used in the experimental evaluation of SLM [4]. DS_1 contains 500M logs from 500K distinct contributors. The logs consist of more than 16M malicious source IPs. The second dataset DS_2, one month (October 2016), is used for verifying the prediction performance of our method for latest attack events. We observe that only 0.008% and 0.001% of victims and attackers within first 5 days of DS_1 also appear in DS_2, respectively. Moreover, only 0.019% of malicious subnets appear in DS_2. Both datasets are different in terms of contributors and attackers.

5.2 Setup

We use sampled datasets DS_1 and DS_2 of real logs of malicious activities from October 2008 and October 2016, respectively. The blacklist length N is bounded, say by 1000. For each victim, we define the *prediction ratio* to be the ratio of the hit count over the total number of attacks on the victim, where *hit count* represents the number of correctly predicted attackers. It outlines the portion of the attackers that have been correctly predicted and the malicious activities of these sources are reported in the T_{test}[5].

The training window T_{train} contains data of some consecutive days. The testing window T_{test} contains the data of the day after the training data. The training and testing window sizes are obtained in Section 5.3.6. We use the training window data to empirically obtain γ, λ_1, λ_2, *min_support*, *min_confidence*, and *min_lift* in Section 5.3.7.

5.3 Performance Evaluation

We arrange the predictors in two groups provided whether they use local or global information. In the local group, there are our predictor function h, Time Series (TS) model in [4], and LWOL, as they use the local logs. We use the LWOL prediction ratio as the ground truth. In the global group, there are our predictor Eq. (10), the SLM method, *Victim Similarity*, *Attacker Correlation* and GWOL as they all use the information shared by different networks.

5.3.1 Prediction performance over DS_1

Fig. 1 compares the prediction ratios of our predictor function h, Time Series (TS) model in [4], and baseline prediction LWOL over $DS1$. The x-axis is the test day, from day 6 to day 11. The first 5 days are for training. The y-axis is the prediction ratio on the given day. The prediction ratios of our predictor function h range between 26% and 63%

[5] A perfect prediction ratio (*i.e.*, equals to 1) indicates all attacks are included.

on the given days depending on the available data on different days. The predictor function h outperform both TS and LWOL over all days. Overall, average prediction ratio of h is 48%, which is twice that of the (TS) model in [4] algorithm.

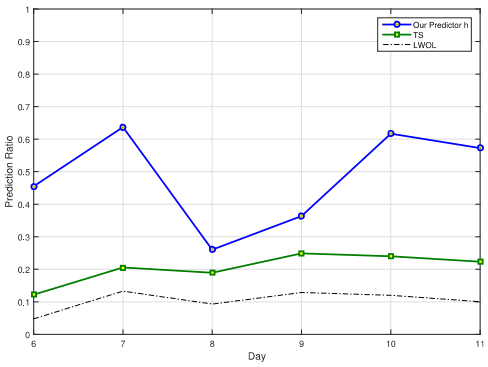


Fig. 1. Prediction ratios of our predictor function h, Time Series (TS) model in [4], and baseline prediction LWOL over DS_1.

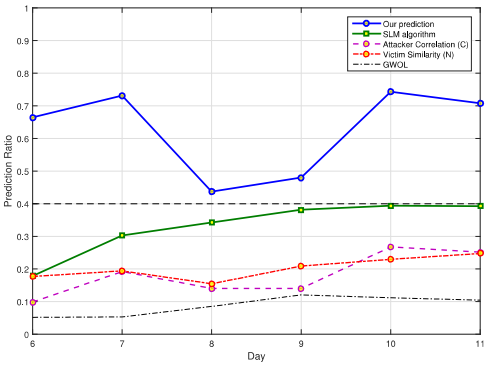


Fig. 2. Prediction ratios of our algorithm, SLM algorithm, neighborhood models, and the baseline prediction GWOL over DS_1.

Due to exponential characteristics of the sigmoid function, predictor h expresses the probability of an attack in an exponential manner. Therefore, our predictor function h converges to 1 faster than the Time Series (TS) model in [4]. Our predictor quickly responds to the rapid change of attackers and makes faster decisions for blacklisting an attacker in its initial attacking phase. On the other hand, when an attacker decides to cease attacking, our predictor function h converges to 0 rapidly and eliminates such an attacker from the blacklist to make space for another severe attacker. h and TS share 20% of predicted common attackers. However, h solely blacklists additional 42% attackers that are not captured by TS.

Fig. 2 shows the prediction ratios of our proposed algorithm Eq. (10), the SLM method[6], *Victim Similarity* (N), *Attacker Correlation* (C), and the baseline prediction (GWOL) over $DS1$. Our predictor has the best performance on all 6 days. Our pre- diction ratios are higher than 70% on days 7, 10 and 11, whereas the prediction ratios of the SLM algorithm are all lower than 40%. Overall, our average prediction ratio is 62%, which is almost twice that of the SLM algorithm. In Fig. 2, we plot both *Victim Similarity* (N) and *Attacker Correlation* (C). Their prediction ratios consistently over-perform GWOL over all days. They capture different concepts of similarity (victims vs. attackers). Hence, they potentially capture different set of attackers, which shows the difference in prediction ratios. For instance, we observe that *Victim Similarity* (N) and *Attacker Correlation* (C) share some attackers range between 3% and 49% on the given day.

Another reason for our better performance is that we discover stronger correlations among victims and attackers by leveraging *lift* to measure the stability of discovered rules and discard the misleading ones. In addition, if a victim is an active contributor of *Dshield.org*, the attacker correlation model discards highly prolific and global attackers that may be irrelevant to the victim by only considering the attackers in direct and indi-

[6] We implemented the SLM algorithm in order to conduct a comparison using the same dataset.

rect subnets. Since we apply the same weight function in combining individual predictors as in SLM, the significant improvement in prediction performance is due to better correlations among victims and attackers and the fast-responding predictor function h.

Our prediction ratios range between 43% and 74% with large variance. This is due to variance among the number of attackers which the networks experience in different testing windows. The prediction ratio starts to decrease as several unseen attackers in the training window appear in day 8 of the testing window. In our sampled dataset, 70% of reported malicious activities on day 8 are newly joined attackers. Our predictor function h fails to capture new attackers since there are no interactions between victims and new attackers. The attacker correlation model also fails since there are no similar attackers to new attackers. Our algorithm cannot properly predict a zero-day attack that has no similarity to the attackers reported in T_{train} (see Section 5.3.3 for more details).

5.3.2 Prediction performance over DS_2

This analysis attempts to verify correctness of the observations (4.1, 4.2, 4.3 and 4.4). We repeat the previous experiments on new malicious activities available in DS_2 using the same parameters as in DS_1.

Fig. 3 compares the prediction ratios of our algorithm and the SLM algorithm. Our prediction ratios are higher than 60% over all days, whereas the prediction ratios of the SLM algorithm are all lower than 56%. The average prediction performance is higher than the SLM algorithm. Overall, our average prediction ratio is 64%, which is 1.3 times higher that of the SLM algorithm.

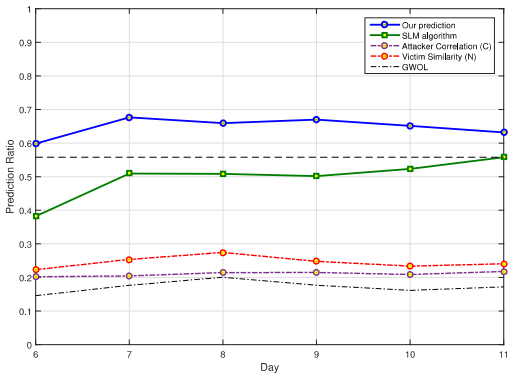
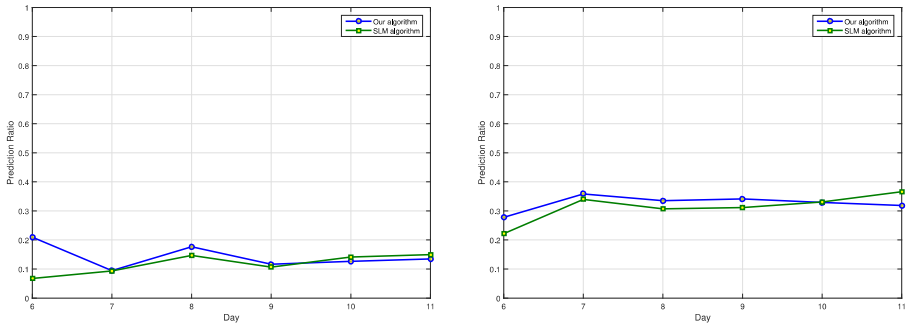


Fig. 3. Prediction ratios of our algorithm, SLM algorithm, neighborhood models, and the baseline prediction GWOL over DS_2.

In DS_2, networks experience fewer unseen attackers in different testing windows than DS_1. For example, on average, only 7% of reported malicious activities are newly joined attackers. This experiment further shows that the SLM algorithm performs better over a stable network as its average prediction ratio increases from 33% (Fig. 2) to 49% (Fig. 3), whereas our average prediction ratio remains stable from 62.7% (Fig. 2) to 64.7% (Fig. 3). This illustrates that our algorithm performs better in both stable and unstable networks with significantly higher average prediction ratios.

5.3.3 Predicting new attacks

Proactive prediction is an important defense strategy for a victim to use against attackers that have not been seen previously in its network. We say that a *new attack* relative to victim v occurs if v has not reported this attacker in the training window. However, such an attacker may not be new for other victims since some victims reported this attacker in the training window. Note that a *new attack* differs from a zero-day attack which introduces an attacker that has never been reported by all victims. In this experiment, Fig. 4 (a) shows that our average proactive prediction ratio is 14% over $DS1$, which is 2.5% higher than that of the SLM algorithm. In Fig. 4 (b), our average proactive prediction ratio is 32% over $DS2$, which is slightly higher than that of the SLM algorithm. For each victim, we only consider the portion of predictions that contain *new attackers*. Since the time-based predictions (predictor function h and TS) are not useful in this prediction, the performance gap demonstrates the effectiveness of our *Victim Similarity* and *Attacker Correlation models.*



(a) DS_1 (b) DS_2

Fig. 4. Proactive prediction ratio results over 6 consecutive days.

5.3.4 Performance consistency

We investigate the stability of predictions by introducing performance consistency that measures the strength of a prediction algorithm over some consecutive testing windows. For each victim, we compare our algorithm with the SLM algorithm on intervals Δ_6, Δ_7, ..., $\Delta_{11} \in T_{test}$ and obtain 6 improvement values. Let $IV(v) = \{IV_6(v), IV_7(v), ..., IV_{11}(v)\}$, where $IV_i(v)$ is the prediction ratio of our algorithm minus that of the SLM algorithm for given blacklists at time interval Δ_i. The consistency index (CI) for each victim is the number of times that our algorithm performs better than the SLM algorithm [4]. Fig. 5 shows the sorted CI values for all victims, where the x-axis is cumulative percentage of victims and the y-axis is CI. Performance prediction of our algorithm over $DS1$ (blue curve) is highly consistent for 16% of victims. They have CI values of 6, meaning that for all time intervals, our algorithm always has more hits than the SLM algorithm. Our algorithm obtains a fairly high consistent index for all victims. The CI values are at least 4, meaning that our algorithm compiles a high quality blacklist not only within one time interval, but also within nearly all time intervals. Performance prediction of our algorithm over $DS2$ (red curve) is extremely consistent for almost 50% of

victims. This illustrates that even when the average prediction performance of the SLM [4] algorithm improves on *DS*2, our method still provides a better blacklist for each victim.

5.3.5 Blacklist length

Fig. 6 shows the performance of our predictor for various blacklist lengths. The prediction ratio increases significantly when the blacklist length increases from 100 to 1000. The prediction ratio improves slightly when the blacklist increases from 1000 to 3000. In fact, a blacklist of length 1000 has average prediction ratio of 62%. While a blacklist of 3 times higher has a prediction ratio of 78%. Therefore, we adopt the blacklist length to be 1000.

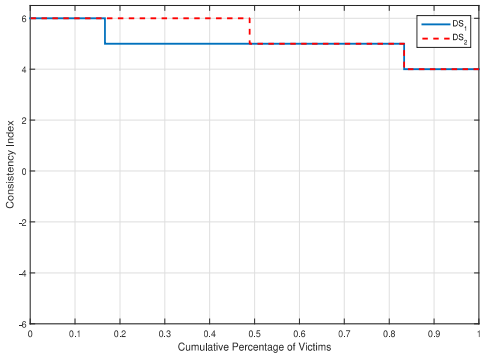


Fig. 5. Cumulative distribution of consistency index across 6 consecutive days.

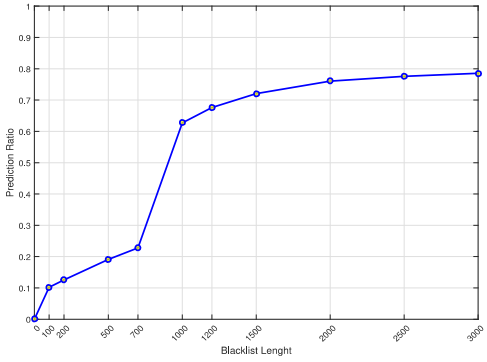


Fig. 6. Prediction ratio as a function of blacklist length N over DS_1.

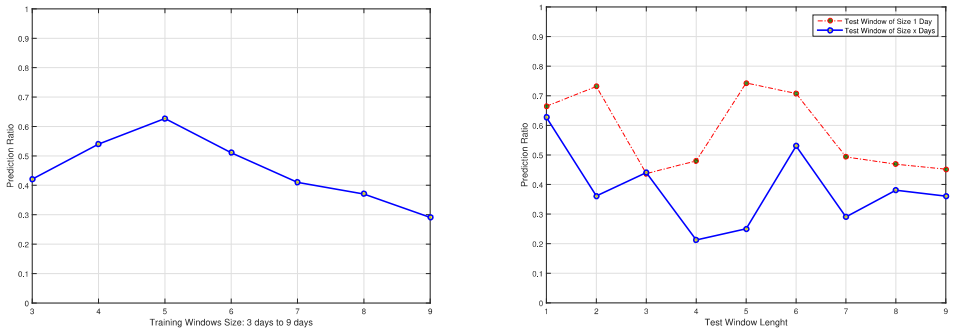
5.3.6 Training and testing windows

We examine how training and testing window sizes affect the performance of our algorithm. The training window size determines the amount of history data as the input to our algorithm. The testing window size shows how often we need to re-compute the blacklist.

Fig. 7 (a) shows the relation between the prediction ratio and the training window size (in days), where the testing window size is fixed to 1 day. When the training window duration is short, the profit of the predictor function h is not much due to only a few available interactions among attackers and victims. If the training window duration is too long, old activities damage prediction performance. The curve in Fig. 7 (a) shows that the best training window duration is 5 days similar to SLM [4].

Fig. 7 (b) shows the prediction ratios as a function of the testing window size, where we fix the training window size to be 5 days. Each point on the blue curve denotes the average prediction ratio of 6 consecutive days. In this experiment, we measure the effect of different testing window durations from 1 to 9 days. When the testing window duration increases, the prediction ratios vary inconsistently. This is due to many unseen attackers in T_{test}, as in Fig. 2. Another reason is that the discovered similarities among victims and attackers are not compatible with a large testing window duration. In Fig. 7 (b) the dashed-line shows the prediction ratios, when the testing window size is fixed to

1 day similar to Fig. 2. It has a higher prediction ratio over other testing window sizes. Thus, a shorter testing window size is preferred.



(a) (b)

Fig. 7. Determining training (a) and testing (b) window sizes over DS_1.

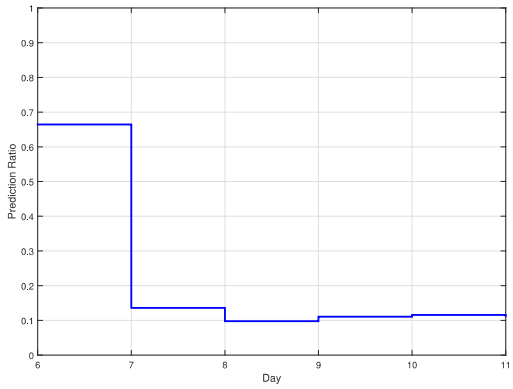


Fig. 8. Determining training and testing window sizes of our algorithm.

In Fig. 8, we use the blacklists compiled on day 6 to predict the rest of testing days. The prediction ratio drops quickly when blacklists are more than one day old. For example, when blacklists are two days old (day 7), the prediction ratio decreases by nearly 60%, from 73% in Fig. 2 to 13%. Therefore, blacklists need to be re-computed frequently.

5.3.7 Parameter selection

We use the training window data from DS_1 to obtain *min_support*, *min_confidence*, *min_lift*, λ_1, and λ_2. Fig. 9 shows the prediction ratios as a function of *min_support* and *min_confidence*, where we fix λ_1 and λ_2. The best accuracy is obtained when *min_support* = 0.9, *min_confidence* = 0.2. More precisely, accuracy continuously increases as long as *min_support* increases. It seems a *min_support* $\geq$ 0.8 provides a better threshold for victim similarity and attacker correlation models. The other choices of *min_support* and *min_confidence*, do not significantly change the prediction ratios.

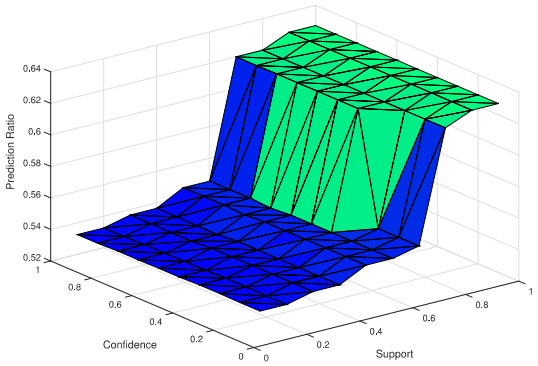


Fig. 9. Prediction ratio as a function of *min_support* and *min_confidence* over DS_1 with *lift* ≥ 1.

Similarly, we empirically search the space of λ_1 and λ_2 to measure the effect of different values over prediction ratios. We observe that the results did not notably change for different choices of λ_1 and λ_2. Thus, we let $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$ contribute equally to both *Victim Similarity* and *Attacker Correlation* models.

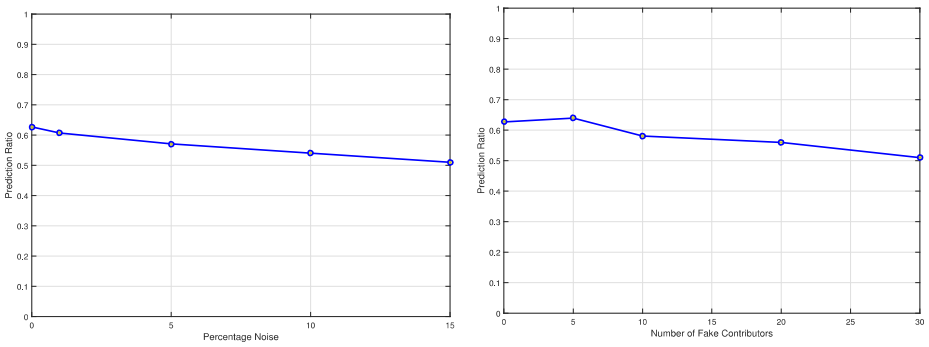
For *min_lift*, we tried the cases where *min_lift* < 1, *min_lift* = 1, and *min_lift* > 1. We observed that *min_lift* > 1 obtains higher prediction ratios as it only considers highly correlation rules.

5.3.8 Robustness

The logs in the *Dshield.org* dataset may contain false positive reports as the repository has no control over contributor's logs. In addition, the reported logs are generated by various types of NIDS with different levels of accuracy. Therefore, the origin of false alerts may be due to errors in contributor's NIDS (pollution), or to a malicious contributor who poisons the dataset by sending fake reports in order to mislead prediction algorithms (poisoning).

Pollution: To study the effect of random false positive reports, we randomly generate noisy logs and have the victims to report them. The victims of reporting fake malicious activities are chosen uniformly and randomly. We examine the noise percentages of 1%, 5%, 10% and 15%. Each point on the blue curve denotes the average prediction ratio of 6 consecutive days. Fig. 10 (a) demonstrates the results with noisy percentages shown on the x-axis and prediction ratios shown on the y-axis. Similar to SLM, the prediction ratio decreases almost linearly as the percentage of fake reports increases. For example, when the noise percentage increases from 0% to 15%, the performance ratio decreases by nearly 11%, from 62% to 51%.

This phenomenon can be explained as follows. Randomly generated noisy reports are unlikely to affect the victim similarity model. They do not produce similarities among victims. According to the victim similarity model in Eq. (6), for a pair of victims to appear in the neighborhood of each other, the noisy reports should not only have the same source, but also be within the same time interval.



(a) (b)

Fig. 10. Robustness of our proposed method with respect to (a) noise percentage of training dataset, and (b) number of fake contributors over DS_1. In both experiments, the prediction ratio decreases linearly as the amount of random noise increases.

Poison: Our method uses the collection of reported malicious activities from trusted8 contributors to compile a customized blacklist for each contributor. Yet, an attacker can masquerade itself as a fake contributor and produce fake reports. A set of colluding contributors could cooperate by submitting the same or overlapping reports. These fake reports have the potential to affect the prediction performance of a true contributor that has some overlap with the fake reports. This also describes an attempt of a botmaster to execute an attack by using the same set of attackers (IPs).

To study the effect of fake reports, we introduce some fake contributors in the training dataset T_{train}. We generate a set of attackers A_{Sybil} and have the fake contributors to report them. In order to differentiate the attackers in A_{Sybil} from other prolific attackers that are likely to belong to the blacklist in any case, we chose some new attackers from legitimate IPs that have never been reported. We set the size of A_{Sybil} equal to the average number of attacks reported by true contributors.

We examine a number of 5, 10, 20 and 30 fake contributors. Fig. 10 (b) demonstrates the results with the number of fake contributors shown on the x-axis and prediction ratios shown on the y-axis. Each point on the blue curve denotes the average prediction ratio of 6 consecutive days. Initially, the average prediction ratio slightly increases as the attackers reported by the set of colluding contributors have some overlap with the attackers reported by true contributors. This indicates that the fake reported activities successfully affect the similarity models and, thus, the set of colluding contributors appear in the neighborhood of true contributors. The average prediction ratio reduces slightly as the number of fake contributors increases. For example, every day on average only 3.4% of IPs has been blacklisted falsely after introducing 30 fake contributors. This indicates that a larger set of colluding contributors who have some overlap with the attackers reported by true contributors have a higher potential to affect the similarity models. However, the set of colluding contributors still loosely affect the victim similarity model since there are enough similar true contributors available that are more related to the target victim than the fake contributors. This demonstrates the robustness of the victim similarity model in Eq. (6) against fake contributors. Generally, while there are enough similar true contributors available, the effect of fake reports diminishes as the

victim similarity model will also contain true contributors. Nevertheless, the reports from fake contributors have less likelihood to affect the predictor function *h*, unless they share the same attackers.

5.3.9 Evading the predictor

It is difficult for persistent attackers to evade our predictor by limiting their activities. They can attack different networks for a short time and pretend to be inactive later. For example, they attack a victim within a few time intervals and become inactive for some consecutive intervals. This behavior deludes the temporal predictor function *h*, which is designed to deal with active attackers. However, the victim similarity and attacker correlation models can capture these attackers.

It remains for future work to apply more sophisticated poisoning attacks such as [28] and verify the effectiveness of Sybil attack detection schemes.

5.3.10 Limitations of our algorithm

Our algorithm makes use of a central repository, such as *Dshield.org*, of shared malicious activities reported by victims all over the Internet. It leverages direct relationships between attackers and victims, victim similarity, and attacker correlation to compile a customized blacklist for each victim. Therefore, victims that regularly share their security logs benefit from high quality blacklists.

Yet, we are unable to deliver a high quality blacklist to a *free-rider* victim. The free-rider problem occurs when a victim chooses not to share its security logs with a central repository. Free-riders take advantage of the central security logs, hoping that other victims would contribute their information. This results in an under-provision of service. The compiled blacklist for such a victim is solely based on the attacker correlation model, which treats the victim as a newly joined and incorporates the globally prolific attackers.

5.4 Complexity Analysis

We analyze the time complexity of our proposed algorithm in training and testing processes. Let $T_{train}|A|$, $T_{train}|S|$, and $T_{train}|V|$ denote the number of attackers, attacker subnets and victims in the training window, respectively. In the training process, the complexities of two phases of our ARM algorithm are

1. Finding all frequent itemsets: Since we only require the correlation between two items, the complexity of the first phase is $O(w \times Cob(T_{train}|V|, 2) \times T_{train}|V|)$, where w is training window size and $Cob(T_{train}|V|, 2)$ is the number of possible combinations of two 2 victims.
2. Generating association rules from the frequent itemsets: Assuming all frequent itemsets satisfy *min_support*, the complexity of phase 2 is at most $O(Cob(T_{train}|V|, 2))$.

Therefore, the overall complexity of discovering victim similarity is $O(Cob(T_{train}|V|, 2) \times T_{train}|V|)$. Similarly, the complexity of determining correlated attackers is $O(Cob(T_{train}|S|, 2) \times T_{train}|S|)$. Since $T_{train}|S|$ is smaller than $T_{train}|V|$ by several orders of magnitude, the complexity of our algorithm in the training phase is $O(T_{train}|V|^3)$.

For testing, computing $h(T_{(a,v)})$ needs $O(w)$ operations for each prediction. The com-

plexity of computing victim similarity, assuming the existence of similarity among all victims, is at most $O(w \times T_{train}|V|)$. The complexity of determining attacker correlations is $O(T_{train}|S|)$. Therefore, the overall complexity of our algorithm in the test phase for predicting an $n \times m$ dimension matrix $\boldsymbol{B}_{b+1}$ is $O(w \times T_{train}|A| \times T_{train}|V|)$, where $n = T_{train}|A|$ and $m = T_{train}|V|$.

In our experiment, we use a commodity server with 2.1 GHz processor and 128 GBs of RAM. The execution time for the training and test phases are about 7 and 2 hours, receptively. We can improve the execution time, in particular, the test time, by employing advanced computing technology, such as cloud computing and GPU computing. We shall explore in this direction in the future.

6. CONCLUSION

We proposed a novel method of investigating accurate and robust prediction for future attacks. We used predictor function h with an exponentially updated ratio to reflect direct and timely relation between an attacker and a victim. In addition, we tackled the victim/attacker similarity problem by the method of association rule mining.

We analyzed the performance of our prediction algorithm against the previous results. Our prediction boasts a significant improvement in both prediction ratios and predicting new attacks. Our significant prediction performance is due to discovering strong victim similarities and attacker correlations in addition to the fast-responding predictor function h. In our victim similarity and attacker correlation models, *lift* plays an important role in highlighting stable rules.

REFERENCES

1. "Dshield dataset," https://www.dshield.org/.
2. J. Zhang, P. Porras, and J. Ullrich, "Highly predictive blacklisting," in *Proceedings of the 17th Conference on Security Symposium*, 2008, pp. 107-122.
3. Z. Chen and C. Ji, "Optimal worm-scanning method using vulnerable-host distributions," *International Journal of Network Security*, Vol. 2, 2007, pp. 71-80.
4. F. Soldo, A. Le, and A. Markopoulou, "Predictive blacklisting as an implicit recommendation system," in *Proceedings of the 29th Conference on Information Communications*, 2010, pp. 1640-1648.
5. A. Ramachandran, N. Feamster, and S. Vempala, "Filtering spam with behavioral blacklisting," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007, pp. 342-351.
6. S. Sinha, M. Bailey, and F. Jahanian, "Improving spam blacklisting through dynamic thresholding and speculative aggregation," in *Proceedings of the 17th Conference on Network and Distributed System Security*, 2010.
7. T. Ouyang, S. Ray, M. Allman, and M. Rabinovich, "A large-scale empirical analysis of email spam detection through network characteristics in a stand-alone enterprise," *Computer Networks*, Vol. 59, 2014, pp. 101-121.
8. N. Chiluka, N. Andrade, J. Pouwelse, and H. Sips, "Social networks meet distributed systems: Towards a robust sybil defense under churn," in *Proceedings of the*

10th ACM Symposium on Information, *Computer and Communications Security*, 2015, pp. 507-518.
9. N. Z. Gong, M. Frank, and P. Mittal, "Sybilbelief: A semi-supervised learning approach for structure-based sybil detection," *Transactions on Information Forensics and Security*, Vol. 9, 2014, pp. 976-987.
10. H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," *IEEE/ACM Transactions on Networking*, Vol. 18, 2010, pp. 885-898.
11. A. Mislove, A. Post, P. Druschel, and K. P. Gummadi, "Ostra: Leveraging trust to thwart unwanted communication," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, 2008, pp. 15-30.
12. H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: Defending against sybil attacks via social networks," *IEEE/ACM Transactions on Networking*, Vol. 16, 2008, pp. 576-589.
13. G. Danezis and P. Mittal, "Sybilinfer: Detecting sybil nodes using social networks," in *Proceedings of the 16th Conference on Network and Distributed System Security*, 2009.
14. J. R. Douceur, "The sybil attack," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, 2002, pp. 251-260.
15. C. Peng, M. Xu, S. Xu, and T. Hu, "Modeling and predicting extreme cyber attack rates via marked point processes," *Journal of Applied Statistics*, 2016, pp. 1-30.
16. Z. Zhan, M. Xu, and S. Xu, "Predicting cyber attack rates with extreme values," *Transactions on Information Forensics and Security*, Vol. 10, 2015, pp. 1666-1677.
17. Z. Durumeric, M. Bailey, and J. A. Halderman, "An internet-wide view of internet-wide scanning," in *Proceedings of the 23rd USENIX Conference on Security Symposium*, 2014, pp. 65-78.
18. E. Wustrow, M. Karir, M. Bailey, F. Jahanian, and G. Huston, "Internet background radiation revisited," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, 2010, pp. 62-74.
19. E. Glatz and X. Dimitropoulos, "Classifying interet one-way traffic," in *Proceedings of ACM Conference on Internet Measurement Conference*, 2012, pp. 37-50.
20. J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed., Morgan Kaufmann Publishers Inc., CA, 2011.
21. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, 1994, pp. 487-499.
22. F. Soldo, "Predicting future attacks data analysis of dshield data set," Technical Report, http://www.ece. uci.edu/ athina/PAPERS/dshield-analysis-tr.pdf.
23. J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez, "Recommender systems survey," *Knowledge-Based Systems*, Vol. 46, 2013, pp. 109-132.
24. S. Katti, B. Krishnamurthy, and D. Katabi, "Collaborating against common enemies," in *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, 2005, p. 34.
25. M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon, " in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, 2006, pp. 41-52.

26. E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," in *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet Workshop*, 2005, p. 6.
27. S. Shin and G. Gu, "Conficker and beyond: A large-scale empirical study," in *Proceedings of the 26th Annual Computer Security Applications Conference*, 2010, pp. 151-160.
28. C. Liu, P. Gao, M. Wright, and P. Mittal, "Exploiting temporal dynamics in sybil defenses, " in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 805-816.

Amir Rezapour is currently a Ph.D. candidate in Computer Science Department at National Chiao Tung University, Taiwan. He received his M.S. degree in Computer Science from National Tsing Hua University, Taiwan, in 2013. His research interests are in the area of cryptography and network security.

Wen-Guey Tzeng (曾文貴) received his BS degree in Computer Science and Information Engineering from National Taiwan University, Taiwan, 1985; and MS and Ph.D. degrees in Computer Science from the State University of New York at Stony Brook, USA, in 1987 and 1991, respectively. He joined the Department of Computer Science, National Chiao Tung University, Taiwan, in 1991. His current research interests include security data analytics, cryptology, information security and network security.