

Predictive Blacklisting as an Implicit Recommendation System

Fabio Soldo, Anh Le, Athina Markopoulou

University of California, Irvine
{fsoldo, anh.le, athina}@uci.edu

Abstract—A widely used defense practice against malicious traffic on the Internet is to maintain blacklists, *i.e.*, lists of prolific attack sources that have generated malicious activity in the past and are considered likely to do so in the future. Traditional blacklisting techniques have typically focused on the prolific attack sources and, more recently, on collaborative blacklisting. In this paper, we study *predictive blacklisting*, *i.e.*, the problem of forecasting attack sources based on past, shared attack logs, and we formulate it as an *implicit recommendation system*. Inspired by the recent Netflix competition, we propose a multilevel prediction model that is tailored specifically for the attack forecasting problem. Our model captures and combines various factors, namely: attacker-victim history (using time-series) and attackers and/or victims interactions (using neighborhood models). We evaluate our combined method on one-month of logs from Dshield.org and we demonstrate that it improves significantly the prediction rate over state-of-the-art methods as well as the robustness against poisoning attacks.

I. INTRODUCTION

A widely used defense practice against malicious traffic on the Internet today is through blacklists: lists of the most prolific attack sources are compiled, shared, and eventually blocked. Examples of computer and network blacklists include IP and DNS blacklists to help block unwanted web content, SPAM producers, and phishing sites. Sites, such as DShield.org [1], process firewall and intrusion detection system (IDS) logs contributed by hundreds of victim networks worldwide, and compile and publish blacklists of the most prolific attack sources reported in these logs.

Blacklists essentially attempt to forecast future malicious sources based on past logs. It is desirable that they are *predictive*, *i.e.*, include many of the malicious sources that will appear in the future and as few false positives as possible. It is also desirable that the blacklist size is short, especially when the blacklist is used online for checking every flow on the fly. Predicting future malicious activity accurately and in a compact way is a difficult problem. Given the wide use of blacklists on the one hand, and the inherent complexity of the problem on the other hand, it is surprising how little has actually been done so far to systematically treat this problem.

The two most common techniques are GWOL and LWOL, according to the terminology of [2]. LWOL stands for “Local Worst Offender List”: security devices deployed on a specific site keep logs of malicious activity, and a blacklist of the most prolific attack sources, in terms of target IPs, is compiled. This local approach, however, fails to predict attack sources that have never previously attacked this site; in this sense, a local blacklist protects the network reactively rather than

proactively. Meanwhile, GWOL stands for “Global Worst Offender List” and refers to blacklists that include top attack sources that generate the highest number of attacks globally, as reported at universally reputable repositories, such as [1], [3]. A problem with this approach is that the most prolific attack sources globally might be irrelevant to some victim networks that do not provide the corresponding vulnerable services.

Recently, Zhang *et al.* [2] proposed a collaborative blacklisting technique called “highly predictive blacklisting”(or HPB). They studied flow logs from DShield.org, defined the victim-to-victim similarity graph, and applied an algorithm resembling the Google’s PageRank algorithm to identify the most relevant attackers for each victim. The HPB approach improved over LWOL and GWOL and is, to the best of our knowledge, the first methodological development in this problem area in a long time.

Our work builds on and improves over [2]. Throughout the paper we use the terms *attack forecasting* and *predictive blacklisting* interchangeably. We formulate the problem using a different methodological framework inspired by the emerging area of *recommendation systems* (RS) [4]–[7]. Based on shared security logs, we study malicious behavior at the IP level, *i.e.*, considering the (attacker IP, victim IP, time) tuple. We predict future malicious activity based on the past, and we construct predictive blacklists specific to each victim. We exploit both temporal (attack trends) and spatial (similarity of attackers and victims) features of malicious behavior. One family of temporal techniques predicts future attacks using the time series of the number of reported attacks. Another family of spatial techniques explores neighborhoods of victims as well as of joint attackers-victims. We analyze 1-month of DShield.org data and evaluate different candidate techniques. We optimize each technique independently and then combine them together. We show that the combined method significantly improves the performance, *i.e.*, increases the predictiveness, or “hit count”, of the blacklists over baseline approaches. Specifically, it improves up to 70% the hit count of the HPB scheme with an average improvement over 57%. Last but not least, the formulation of the problem as an implicit recommendation system opens the possibility to apply powerful methodologies from machine learning to this problem.

The rest of this paper is organized as follows. Section II discusses related work. Section III gives a brief overview of some key features of the DShield.org dataset. Section IV formulates the attack prediction problem in the recommendation systems framework; it also motivates this study by

showing the gap between state-of-the-art approaches and the upper bound (achieved by an offline algorithm.) Section V presents the specific temporal and spatial methods we use for prediction. Section VI evaluates the individual methods and their combination over the `Dshield.org` dataset; the combined method significantly outperforms the current state-of-the-art approach. Section VII concludes and discusses open issues and future work.

II. OUR WORK IN PERSPECTIVE

The two traditional approaches to generate blacklists, LWOL and GWOL, have already been outlined in the introduction. They both select the most prolific attackers based on past activity recorded in logs of a single victim site (in the case of LWOL) or of multiple victim sites (in the case of the GWOL.) The local approach can be implemented by the operator of any network independently. The global approach uses more information that may or may not be relevant to particular victims and requires sharing of logs among multiple victims, in a distributed way or through central repositories.

Beyond the traditional approaches, the state-of-the-art method today is the “highly predictive blacklisting” (HPB), recently proposed by Zhang *et al.* [2]. The main idea was that a victim should predict future attackers based not only on his own logs but also on logs of a few other “similar” victims. Similarity between two victims was defined as the number of their common attackers, based on empirical observations made earlier by Katti *et al.* [8]. A graph that captures the similarity of victims was considered, and an algorithm resembling Google’s PageRank was run on this graph to determine the relevance of attackers for a victim. Predictive blacklisting was essentially posed as a link-analysis problem.

Compared to HPB [2], our work solves the same problem (predictive blacklisting based on shared logs), but we have several differences in methodology and intuition. We make the following contributions. (1) We formulate the problem as an *implicit recommendation system (RS)* [4]; this opens the possibility to apply a new set of powerful techniques from machine learning. Within the RS framework, we combine a number of techniques that capture and predict different behaviors present in our dataset. (2) One set of techniques is *spatial*, *i.e.*, using the notion of similarity of victims and/or attackers. HPB is a spatial case, where similarity is considered only among victims and is defined as the number of common attackers. (2a) We use a different notion of victim to victim similarity which focuses on simultaneous attacks from common sources (attacks performed by the same source at about the same time induce stronger similarity among victims.) (2b) Furthermore, we also define another notion of neighborhood that takes into account blocks of attackers and victims jointly, using a co-clustering algorithm called cross-association (CA) [9]. (3) Another set of techniques use time series to exploit *temporal trends* for prediction; to the best of our knowledge, this axis has not been exploited before for predictive blacklisting.

Our work falls within the category of behavioral analysis, in the sense that inferences are made based on flow logs as opposed to packet payload. However, we are interested in

prediction and not in traffic classification [10] or distinguishing legitimate from malicious traffic [11], [12], *i.e.*, we work with flow logs that have already been classified as malicious by IDS and we focus on prediction.

III. THE DSHIELD DATASET: OVERVIEW AND KEY CHARACTERISTICS

In this study, we used logs from `Dshield.org` to understand the patterns existing in real data and to evaluate our prediction methods in practice. In this section, we briefly describe the dataset and mention some key properties that influenced the design of our prediction methods.

The dataset. `Dshield.org` [1] is a repository of firewall and intrusion detection logs collected at hundreds of different networks all over Internet. The participating networks contribute their logs which include the following fields: time stamp, contributor ID, source IP address, destination IP address, source port number, destination port number, and protocol number. In this paper, we work with the first three fields. One challenge when dealing with large-scale security log sharing systems is the amount of noise and errors in the data. For this reason, we pre-processed our dataset to reduce noise and erroneous log entries, such as those belonging to invalid, non-routable, or unassigned IP addresses. Data from `Dshield.org` have been studied and used by several researchers over the years, such as [2], [8], [13]–[15], to name a few examples.

In this paper, we present results for 1-month of `Dshield.org` logs (October 2008). The pre-processed 1-month dataset consists of about 430M log entries, from ~ 600 contributing networks, with more than 800K unique malicious IP sources every day.

Observations. Fig. 1 showcases some observations from the data that motivated design choices in our prediction. First, Fig. 1(a) offers a visualization of part of the data: it shows the number of logs generated by a portion of the IP space over time. One can visually observe that there are several different activities taking place. Some sources attack consistently and by an order of magnitude higher than other sources (heavy hitters); some attack with moderate-high intensity but only for a few days (spikes); some attack continuously in a certain period and do not appear again; finally, most other sources appear to be stealthy and generate limited activity. The wide variety of dynamics in the same dataset poses a fundamental challenge for any prediction mechanism. Methods, such as GWOL, focusing on heavy hitters will generally fail to detect stealthy activity. Methods focusing on continuous activity will not predict sudden spikes in activity. This motivated us to develop and *combine several complementary prediction techniques* that capture different behaviors.

Second, in Fig. 1(b), we show some information about the temporal behavior. In particular, we consider attack sources that appear at least twice in the logs and study the inter-arrival time between logs for the same attack source. We plot the cumulative distribution function (CDF) of inter-arrival time at three different levels of granularity: IP address, /24 prefix, and source subnet. We observe that for /24 prefixes, 90% of

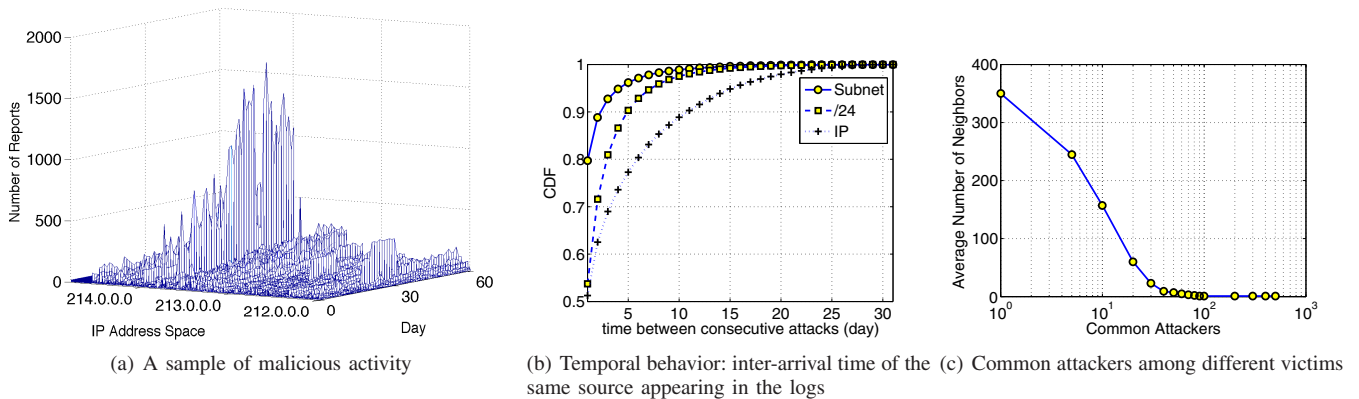


Fig. 1. Some insights from the Dshield.org dataset that motivated some of our design choices.

attacks from the same source happen within a time window of 5 days while the remaining 10% are widely spread over the entire month. Similar trends are true for the other levels of granularity. This implies that attacks have a short memory: if an attacker attacks more than once, then with high probability it will attack again soon. This motivated the *EWMA time series* approach we use for temporal prediction.

Another important aspect influencing the design of our prediction methods is the correlation among attacks seen by different victim networks. We call two victim networks “neighbors” if they share at least a certain number of common attackers. Fig. 1(c) shows the average number of neighbor networks as a function of this number of common attacking IPs for a given day. Most victims share only a few attackers because there are a few source IPs (heavy hitters) that constantly attack most victim networks. However, if we consider a strict definition of neighbors, *i.e.*, sharing a large number of attackers, each victim has a smaller number of neighbor, which is likely to capture a more meaningful type of interaction. This motivated us to consider small neighborhoods (~ 25 nodes) in our spatial prediction methods.

IV. PROBLEM FORMULATION AND FRAMEWORK

Our goal is to predict future malicious IP traffic based on past logs contributed by multiple victims. Predicting malicious IP traffic is an intrinsically difficult problem due to the variety of exploits and attacks taking place at the same time and the limited information available about them.

A. Recommendation Systems vs. Attack Prediction

In this paper, we frame the problem of attack prediction as an implicit recommendation system problem, as depicted in Fig. 2. Recommendation systems aim at inferring unknown user rating about items from known (past) ratings. An example is the Netflix recommendation system, Netflix Cinematech, which aims at predicting unknown user ratings about movies from known ratings, in order to provide movie recommendations to its customers. What makes the prediction possible is that ratings are not given randomly but according to a complex and user-specific rating model, which is not known in advance. The rating matrix is a result of several superimposed processes,

some of which are intuitive, while others need to be unveiled and confirmed through an accurate analysis of the dataset.

Our goal is to predict future attacks leveraging observed past activities. Given a set of attackers and a set of victims, a number r is associated with every (attack source, victim destination, time) triplet according to the logs: r can indicate, for example, the number of time an attacker has been reported to attack a victim over a time period. More generally, we interpret r as the rating, or preference, assigned by an attacker to a victim. There are some important differences from a traditional RS. First, the intensity of an attack may vary over time, thus leading to a time-varying rating matrix. This poses a significant challenge to the direct application of traditional RS techniques that deal with static matrices. Secondly, the rating in this case is implicit, as it is inferred by activity reported in the logs, as opposed to ratings in RS explicitly provided by the users themselves.

In the rest of this section, we first formalize the analogy between recommendation systems and attack prediction. Then, we define upper bounds for prediction and quantify the gap that exists today between the state-of-the-art prediction and what is actually achievable. In subsequent sections, we propose specific methods that bridge this gap.

B. The Recommendation System Problem

1) *Notation:* Let \mathcal{V} be the set of users (customers) and \mathcal{A} be the set of items. A user is allowed to rate items to indicate how much she likes specific items. Let \mathcal{R} be the set of possible ratings, $\mathcal{R} = \{1, 2, \dots, N\}$, where N is typically a small integer. Let $r_{u,i}$ be the rating assigned by user u to item i and R be the entire $|\mathcal{V}|$ -by- $|\mathcal{A}|$ rating matrix.

2) *Problem Formulation:* A recommendation system aims at inferring unknown ratings from known ones, as shown in Fig. 2-left. Let \mathcal{K}_u be the set of items for which the rating $r_{u,i}$'s are known, and $\bar{\mathcal{K}}_u$ be its complement. The goal of RS is to find for every user u , the item, i_u , that has the highest estimated rating. The RS problem can be formalized as [4]:

$$\text{find } i_u = \arg \max_{i' \in \bar{\mathcal{K}}_u} r_{u,i'} \quad \forall u \in \mathcal{V} \quad (1)$$

The recommended item, i_u , for user u , maximizes Eq. (1) and may be different for every user. The solution of Eq. (1)

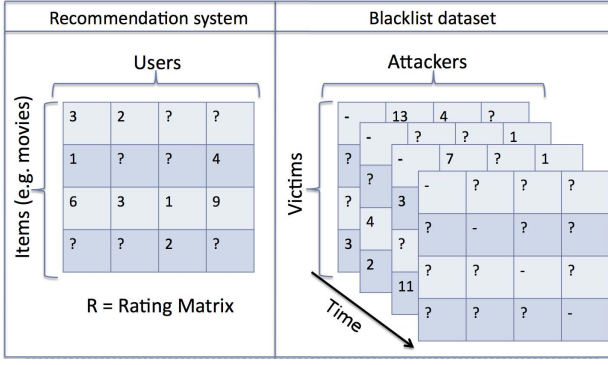


Fig. 2. Analogy between Recommendation Systems (left) and Attack Prediction (right). The former infers unknown user ratings about items from known ones. The latter deals with time varying ratings.

is usually obtained by first estimating the matrix R on the subset \mathcal{K}_u , and then, for every user, selecting the item for which the estimated rating is the highest. In general, if we want to recommend $N \geq 1$ items we need to select the top- N items for which the estimated ratings are the highest.

C. The Attack Prediction Problem

1) *Notation*: We denote with \mathcal{V} the set of victim networks and with \mathcal{A} the set of attackers (*i.e.*, source IP prefixes where attacks are launched from.) Let t indicate the time an attack was performed. Unless otherwise specified, t will indicate the day the attack was reported. T denotes the time window under consideration, so $t = 1, 2, \dots, T$. Moreover, we partition T in two windows of consecutive days: a *training window*, T_{train} , and a *testing window*, T_{test} , to separate training data, used to tune the prediction algorithm, $t \in T_{train}$, from testing data, used to validate the predictions, $t \in T_{test}$.

Similar to the RS problem, we define a 3-dimensional rating matrix B so that per every tuple (a, v, t) , $b_{a,v}(t) = 1$ if an attack from a to v on day t has been reported; 0 otherwise. Finally, we indicate with $\mathcal{A}_v(T)$, the set of attackers that were reported by victim v during the time period T :

$$\mathcal{A}_v(T) = \{a \in \mathcal{A} : \exists t \in T \text{ s.t. } b_{a,v}(t) = 1\}$$

and with $\mathcal{A}(T)$ the total set of attack sources reported in T :

$$\mathcal{A}(T) = \cup_{v \in \mathcal{V}} \mathcal{A}_v(T)$$

2) *Problem Formulation*: For every victim, v , we are interested in determining which attackers are more likely to attack v in the future given past observation of malicious activity. In practice, this translates into providing a blacklist (\mathcal{BL}) of sources that are likely to attack in the future. Given a fixed blacklist size, N , let \mathcal{BL} be any set of N different attackers. The problem of attack prediction can be formalized as follows:

$$\text{find } \mathcal{BL}(v) = \arg \max_{\mathcal{BL} \subset \mathcal{A}} \sum_{t \in T_{test}} \sum_{a \in \mathcal{BL}} b_{a,v}(t) \quad \forall v \in \mathcal{V} \quad (2)$$

The output of the attack prediction problem is a set of blacklists customized for every victim, v , such that each blacklist, $\mathcal{BL}(v)$, contains the top N attackers that are more likely to attack v in the time window T_{test} . The difficulty of

this problem is that for every $t \in T_{test}$, we need to estimate an entire $|\mathcal{A}|$ -by- $|\mathcal{V}|$ matrix before the max operation can be performed, as illustrated in Fig. 2-right. In this sense, this problem is a generalization of the recommendation problem, Eq. (1), where R is now defined on 3-dimensional space, $\mathcal{V} \times \mathcal{A} \times T$, rather than a 2-dimensional space. While the RS problem traditionally estimates missing elements in a matrix, the attack prediction problem estimates matrices in a tensor.

Finally, we observe that for every blacklist \mathcal{BL} , and testing period, T_{test} , the total number of false positive (FP) can be defined as: $FP_{\mathcal{BL}}(T_{test}) = \sum_{t \in T_{test}} (N - \sum_{a \in \mathcal{BL}} b_{a,v}(t))$. Thus, for fixed blacklist length N , solving Problem (2) is equivalent to finding the blacklist that minimizes the number of false positive.

D. Upper Bounds and State-of-the-Art

Given a blacklist of length N , a metric of its predictiveness is the hit count, as defined in [2]: the number of attackers in the blacklist that are correctly predicted, *i.e.*, malicious activity from these sources appears in the logs in the next time slot. A blacklist with a higher hit count is more “predictive.”

A future attacker can be predicted if it already appeared at least once in the logs of some victim networks. Clearly, we cannot accurately predict attackers that have never been reported before. Consequently, we can define two upper bounds on the hit count, a global and a local upper bound, depending on the sets of logs we use to make our prediction.

Definition 1 (Global Upper Bound): Using notations defined above, for every victim v , we define the global upper bound on the hit count of v , $GUB(v)$, as the number of attackers that are both in the training window of any victim and in the testing window of v :

$$GUB(v) = \mathcal{A}(T_{train}) \cap \mathcal{A}_v(T_{test}) \quad (3)$$

This represents the maximum number of attackers of v that are predictable in T_{test} , given observations obtained in T_{train} . This upper bound corresponds to the case that the past logs of all victims are available to make prediction, as it is the case when using central repositories, such as Dshield.org, or when each victim shares information with all other victims.

Definition 2 (Local Upper Bound): For every victim v , we define the local upper bound on the hit count of v , $LUB(v)$, as the number of attackers that are both in the training window and in the testing window of v :

$$LUB(v) = \mathcal{A}_v(T_{train}) \cap \mathcal{A}_v(T_{test}) \quad (4)$$

$LUB(v)$ represents the upper bound on the hit count when each victim, v , has only access to its local security logs, but not to the logs of other victims. This is a very typical case in practice today. Because $\mathcal{A}_v(T_{train}) \subseteq \mathcal{A}(T_{train})$, the following inequality holds trivially: $LUB(v) \leq GUB(v)$.

The next natural question is how far are state-of-the-art methods today from these upper bounds? Existing approaches for creating predictive blacklists include the traditional LWOL and GWOL, as well as the recently proposed HPB [2].

In Fig. 3, we compare the total hit count (for all victims in the system) of different prediction strategies on 1-month of Dshield.org logs. For a fair comparison, we require all

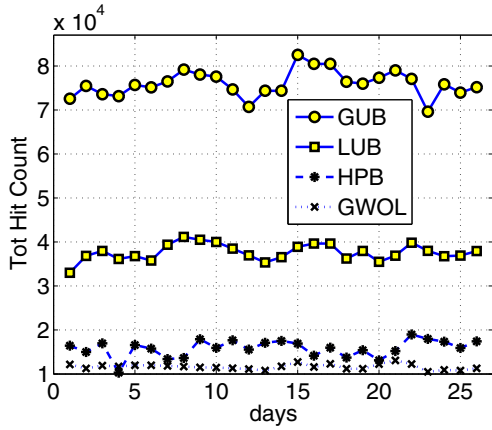


Fig. 3. Comparing different prediction strategies (in terms of total hit-count) on 1-month of Dshield.org logs. Observe that the state-of-the-art HPB improves over baseline, GWOL, but there is still a large performance gap until the upper bounds, LUB and GUB.

methods to use the same blacklist length N . To be consistent with [2], we use $N = 1,000$ and consider that every source in the predictive blacklist is an IP prefix /24. We make two main observations.

First, the state-of-the-art HPB strategy brings benefit over the GWOL strategy. In our dataset, we observed an average improvement in the hit count of about 36% over GWOL, which confirms prior results on older data [2]. However, the gap between HPB and both LUB and GUB is still significant! This shows that there is a large room for improvement in attack prediction, which remains unexplored. This gap motivated us to further investigate the problem in this paper.

The second observation is the large gap between LUB and GUB. This quantifies the improvement in attack prediction when different victim networks share their logs on observed malicious traffic. Collaboration between different networks becomes a crucial factor when dealing with attack prediction because more shared information can potentially reveal correlation between attacks that cannot be discovered otherwise.

V. MODEL OVERVIEW

Motivated by the observations made in Sections III and IV, we develop a multi-level prediction framework to capture the different behaviors and structures observed in the data.

A. Time Series for Attack Prediction

A fundamental difference between forecasting attacks and typical recommendation systems is the way the temporal dynamics affect the ratings. Typically, in recommendation systems, ratings are given at different times, but once given they cannot change. The goal is then to use the known ratings as ground truth and estimate the missing ratings. In contrast, in the attack prediction problem, “ratings” may vary widely over time as they represent the number of attacks (logs) reported in different days. As a consequence, in order to be able to forecast attacks, we must account not only for the time an attack was reported but also for its evolution over time.

Every rating, $b_{a,v}(t)$, is essentially a signal over time. We use a time series approach to model the temporal dynamics

associated with every pair (a, v) . As observed in the data (Fig. 1(b)), multiple attacks from the same source happen within a small time interval from each other, *i.e.*, for the large majority of attacking IP prefixes, the future activity strongly depends on the recent past. Motivated by this observation, we use an Exponential Weighted Moving Average (EWMA) model. We indicate with $r_{a,v}^{TS}(t+1)$ the forecast for $b_{a,v}(t+1)$ given the past observations, $b_{a,v}(t')$, at time $t' \leq t$. $r_{a,v}^{TS}(t+1)$ can be interpreted as a measure of how likely an attacker is to attack again given its past history. We estimate $r_{a,v}^{TS}(t+1)$ as

$$r_{a,v}^{TS}(t+1) = \sum_{t'=1}^t \alpha(1-\alpha)^{t-t'} b_{a,v}(t') \quad (5)$$

where $\alpha \in (0, 1)$ is the smoothing coefficient, and $t' = 1, \dots, t$ indicates the training window, where 1 corresponds to the oldest day considered, and t is the most recent one. Weights assigned to past observations are exponentially scaled so that older observations have smaller weights. This allows to account for spikes in the number of reports, which are frequently observed in our analysis of malicious traffic activities.

B. Neighborhood Model

The strategy described above can model simple temporal dynamics accurately and with low complexity. However, a prediction solely based on time will fail to capture spatial correlations between different attackers and different victims in the IP space. *E.g.*, a persistent attacker that switches its target every day may easily evade this level of prediction. In this section, we show how to capture such “spatial” patterns and use them for prediction. We define two types of neighborhoods: one that captures the similarity of victims (k NN) and another that captures joint attacker-victim similarity (CA).

1) *Victim Neighborhood (k NN)*: One of the most popular approaches in recommendation systems is the use of neighborhood models. Neighborhood models build on the idea that predictions can be made by trusting similar peers. For instance, in a movie recommendation system, a neighborhood model based on user similarity will predict that user John Smith likes Harry Potter, only if users that have shown similar taste to John Smith and have already seen Harry Potter, liked it.

In this context, the definition of similarity plays a fundamental role. There are several different similarity measures proposed in the literature. The most commonly used is the Pearson correlation, which generalizes the notion of cosine distance of vectors with non-zero mean.

In this work, we developed a variation of the Pearson similarity to account for the time the attacks were performed. This is also motivated by [8], which observed that victim networks, that persistently share common attackers, are often attacked at about the same time. For every pair of victims, u, v , we define their similarity, s_{uv} , as

$$s_{uv} = \sum_{t_1 \leq t_2 \in T_{train}} e^{-|t_2-t_1|} \frac{\sum_{a \in \mathcal{A}} b_{a,u}(t_1) \cdot b_{a,v}(t_2)}{\|b_u(t_1)\|_2 \|b_v(t_2)\|_2} \quad (6)$$

where $\|b_u(t_1)\|_2 = \sqrt{\sum_{a \in \mathcal{A}} b_{a,u}^2(t_1)}$. Notice that if u and v report attacks at the same time, s_{uv} reduces to a sum of cosine

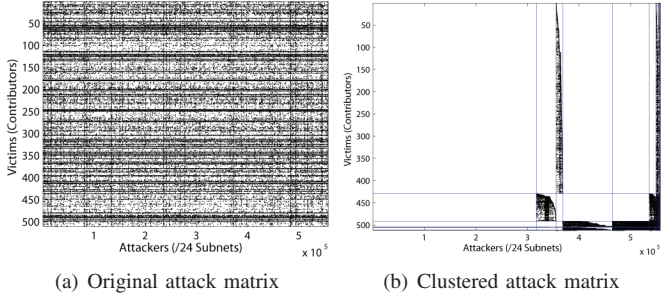


Fig. 4. Result of applying the CA algorithm on 1-day Dshield.org logs. A rectangular block indicates a group of similar sources and victims identified by the CA.

similarities. When u and v report attacks by the same attacker at different times, the smoothing factor, $e^{-|t_2-t_1|}$, accounts for the time interval between the two attacks.

We tried several similarity measures, and we found that the one in Eq. (6) worked best. Attacker activities might vary broadly over time. Eq. (6) models the intuition that victims, that share attacks from the same source in the same time slot, are more similar to each other than victims sharing common attackers but during very different time since they are more likely affected by the same type of attack.

We adapt a k -nearest neighbors (k NN) model to the attack prediction problem. The idea of traditional k NN model is to model missing ratings as a weighted average of known rating given to the same item by similar users:

$$r_{a,v}^{kNN}(t) = \frac{\sum_{u \in N^k(v;a)} s_{uv} r_{a,u}(t)}{\sum_{u \in N^k(v;a)} s_{uv}}, \quad \forall t \in T_{test} \quad (7)$$

where, $r_{a,v}^{kNN}(t)$ is the prediction provided by the k NN model, and $N^k(v;a)$ represents the neighborhood of top k similar victims to v according to the similarity measure, s , for which $r_{a,u}(t)$ is known.

In order to compute $r_{a,v}^{kNN}(t)$, we need two main ingredients: a similarity measure between victims, s , and a set of known rating for the attacker a , $r_{a,u}(t)$. What prevents us from a direct application of Eq. (7) is that none of the ratings, $r_{a,u}(t)$, is known in the testing window. Thus, the neighborhood $N^k(v;a)$ is empty. To overcome this difficulty, we leverage the forecast provided by the time series approach in Eq. (5):

$$r_{a,v}^{kNN}(t) = \frac{\sum_{u \in N^k(v;a)} s_{uv} r_{a,u}^{TS}(t)}{\sum_{u \in N^k(v;a)} s_{uv}}, \quad \forall t \in T_{test} \quad (8)$$

which is a generalization of the k NN model.

2) *Joint Attacker-Victim Neighborhood (CA)*: In addition to the victim neighborhood explored by the k NN model, we also studied the joint neighborhood of attackers and victims. Our intuition is that not only victim similarity but also the similarity among the attackers should be considered when constructing the blacklists. For example, consider botnets, which are the main source of malicious activity on the Internet today: machines in a botnet typically attack the same set of victims. However, the timing of the attacks might differ due to different phases of the attacks [16], [17]: typically a scanning

phase is carried out by a few machines before the attacking phase, which might be carried out by more machines, *e.g.*, in an instance of distributed denial-of-service (DDoS) attack. Therefore, knowing the similarity among the machines of a botnet, even if only a few of them are detected by a victim's IDS, enables the victim to preemptively put the other "similar" machines of the botnet into his blacklist.

To find similarity among both victims and attackers simultaneously, we apply the cross-associations (CA) algorithm [9] – a fully automatic clustering algorithm that finds row and column groups of sparse binary matrices. In this way, we find blocks of victims (contributors) and attackers (/24 subnets.) Fig. 4 depicts the result of applying the CA on a contributor-subnet matrix of 1-day log data. On average, the CA finds over 100 groups per day (more information about the use of the CA algorithm for analyzing Dshield.org logs can be found in our technical report [18].)

For each group (depicted as a rectangular block in Fig. 4), we calculate its density as the ratio between the occupied area and the total area of the rectangle. Then, we use the density of a group to quantify the strength of correlation among the attackers and victims within the group. Intuitively, a dense group corresponds to an attacker-victim bipartite graph that resembles a complete bipartite graph, thus indicating strong correlation. Finally, we use this density for forecasting: the denser a group is, the more likely its attackers will attack its victims.

More formally, $\tilde{r}_{a,v}^{CA}(t+1) = \rho_{a,v}(t)$, where $\rho_{a,v}(t) \in [0, 1]$ is the density of the group that contains the pair (a, v) at time t . We can further improve this CA-based prediction by capturing the persistence of the attacker and victim block over time. In particular, we apply the EWMA model on the time series of the density to predict the rating. The intuition is that if an attacker shows up in a neighborhood of a victim persistently, he is more likely to attack the victim than other attackers. Formally,

$$r_{a,v}^{CA}(t+1) = \sum_{t'=1}^t \alpha(1-\alpha)^{t-t'} \rho_{a,v}(t') \quad (9)$$

Our empirical study shows that the EWMA-CA prediction can improve the hit count by 25% over the simple CA prediction.

C. Combine Predictors

The combination of different predictors is generally referred to as ensemble learning. The idea of ensemble learning is rooted in the traditional wisdom that "in a multitude of counselors there is safety" [19]. Although the gain of ensemble learning is not fully understood yet, it is generally acknowledged that such an approach is particularly suited in scenarios where a complex system is better explained by the combination of different phenomena, which results in different structures in the data, rather than by a single phenomenon, *e.g.*, see Ch. 13 and 18 of [20]. The diverse dynamics observed in the analysis of malicious traffic motivated us to combine diverse algorithms, such as the time series approach to model temporal trends, the k NN to model victims similarity, and

the CA clustering algorithm to model persistent groups of attackers-victims.

There are different methods to combine predictors. A typical approach is to consider the average of individual predictors. What we found more effective is to (i) use the time series prediction as a base predictor and (ii) weight the neighborhood models with weights proportional to their accuracy. More specifically, for k NN we define

$$w_{a,v}^{kNN} = \frac{\sum_{u \in N(v;a)} s_{uv}}{\sum_{u \in N(v;a)} s_{uv} + \lambda_1}$$

where λ_1 is a parameter that needs to be estimated. The intuition is that we want to rely more on k NN when v has a strong neighborhood of similar victims. When $\sum_{u \in N(v;a)} s_{uv}$ is small, *i.e.*, only a neighborhood of poorly similar victims is available, we prefer instead to rely on other predictors. Similarly, we define a weight for the CA algorithm,

$$w_{a,v}^{CA} = \frac{\sum_{t \in T_{train}} \rho_{a,v}(t)}{\sum_{t \in T_{train}} \rho_{a,v}(t) + \lambda_2}$$

so that, $w_{a,v}^{CA} \simeq 1$ for a pair (a, v) that belongs to dense clusters; $w_{a,v}^{CA} \simeq 0$ when the density is low.

In summary, our rule for combining all methods together and giving a single rating/prediction is the following:

$$\hat{b}_{a,v}(t) = r_{a,v}^{TS}(t) + w_{a,v}^{kNN} r_{a,v}^{kNN}(t) + w_{a,v}^{CA} r_{a,v}^{CA}(t) \quad (10)$$

where $\hat{b}_{a,v}(t)$ is the estimated value of $b_{a,v}(t)$, $\forall t \in T_{test}$.

VI. PERFORMANCE EVALUATION

A. Setup

Dataset. We evaluate the performance of our prediction algorithm using 1-month of real logs of malicious IP sources provided by Dshield.org, as described in Section III.

Metric. The hit count was defined in Section IV-D and represents the number of attackers in the blacklist that are correctly predicted; it is bounded by the blacklist length itself. When the algorithm provides individual victims with their customized blacklist, the total hit count is defined as the sum of the hit counts over all victims.

Parameters. Unless otherwise specified, we use a 5-day training window and 1-day testing window. We motivate these choices in Section VI-C. Parameters α , λ_1 and λ_2 are estimated using leave-one-out cross validation on the training set. Finally, for a fair comparison with prior work [2], [1] each predictive blacklist specifies /24 IP prefixes.

Complexity. The complexity of the combined prediction depends on the complexity of the individual methods. Computing the TS prediction requires $O(T_{train})$ operations for each rating $r_{a,v}^{TS}$. Thus, its overall complexity is $O(T_{train}|\mathcal{A}||\mathcal{V}|)$. The complexity of the k NN model is the computation of the similarity matrix $O(T_{train}|\mathcal{V}||\mathcal{V}|)$ plus the complexity of computing Eq.(7) for every pair (a, v) , that is $O(k|\mathcal{A}||\mathcal{V}|) = O(|\mathcal{A}||\mathcal{V}|)$ since k is a constant. Finally, the CA clustering is a heuristic algorithm with a complexity empirically observed to be bounded by $O(|\mathcal{A}||\mathcal{V}|)$ [9]. In practice, $|\mathcal{V}|$ is orders

of magnitude smaller than $|\mathcal{A}|$, thus the overall asymptotical complexity is bounded by $O(T_{train}|\mathcal{A}||\mathcal{V}|)$, that is, it increases linearly with the size of the data set, R . In our experiments, we could compute predictive blacklists for all contributors in ~ 20 minutes with a 2.8 GHz processor and 32 GBs of RAM.

B. Performance Evaluation and Comparison of Methods

We group prediction schemes into two categories depending on whether they use local or global information. In the local category, there are the time series (TS) and LWOL, since they both use only local logs available at each network. In the global category belong the neighborhood models, such as k NN and EWMA-CA, as well as GWOL, since they use logs collected and shared among multiple networks.

In Fig. 5(a), we plot and compare the total hit count of local schemes, namely TS and LWOL. Their performance oscillates based on the specific (training and testing) data available on different days. However, we can see that the TS approach consistently outperforms LWOL over all days. This is expected since the TS has greater flexibility than LWOL to model temporal trends.

In Fig. 5(b), we compare the hit count of global schemes that use information from different networks, namely GWOL, HPB, EWMA-CA, and k NN. We implemented the relevance propagation used in HPB with parameter 0.5. As noted in [2], the average improvement of HPB over GWOL is $\sim 36\%$. The EWMA-CA algorithm has on average the same performance as HPB. However, (i) its performance is more consistent over time than HPB and (ii) the two methods capture different concepts of neighborhood (victim neighborhood in HPB vs. joint victim-attacker neighborhood in EWMA-CA.) Thus, they potentially capture different set of attackers, which explains the difference in performance. Finally, we plot both prediction models for k NN: “ k NN on Train” in Eq. (7) (where k NN is run on top of the last day’s logs), “ k NN on TS” in Eq. (8) (where k NN is run on top of the TS predictions). We set $k = 25$. k NN schemes outperform the other neighborhood schemes, mainly thanks to the notion of similarity that accounts for simultaneous attacks. Computing k NN on top of the TS prediction results in larger improvement.

In Fig.5(c), we show the total hit count achieved by our proposed combined scheme of Eq. (10), which blends together TS, k NN (on TS) and (EWMA) CA and we compare it to the state-of-the-art method (HPB). Our scheme outperforms HPB significantly (up to 70% over the hit count of HPB with an average improvement over 57%) and consistently (in every day of October 2008). We also show the more traditional baseline GWOL that performs even worse than HPB.

We also investigated the reasons behind this improvement in more detail. First, we looked at the set (not only the number) of attackers predicted by each individual method. Each method provides every contributor with a customized blacklist that successfully predicts some attackers. Besides predicting a common set of attackers, the three different methods successfully predict disjoint sets of attacks of significant sizes. *E.g.*, TS and EWMA-CA successfully predict a common set of 9.9 K attackers; however, EWMA-CA alone

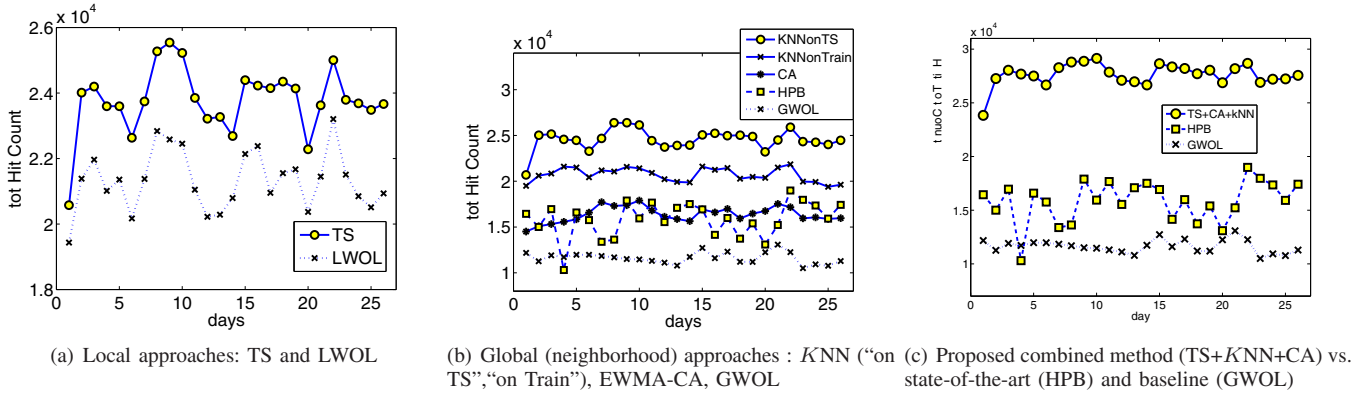


Fig. 5. Evaluating the performance (total hit count) of different individual methods, our proposed combined method (TS+kNN+CA) and baselines methods.

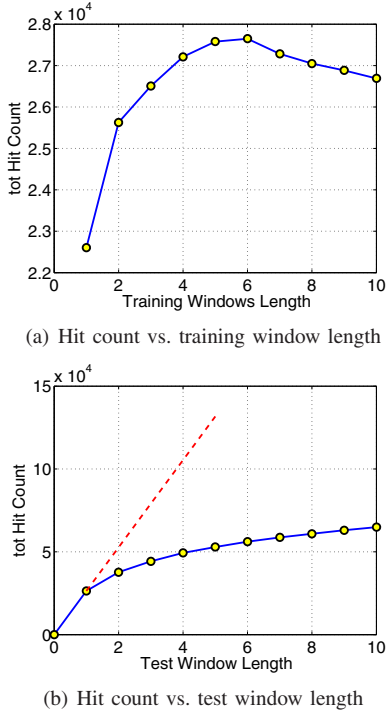


Fig. 6. Tuning the training and testing window of our proposed combined method based on our data. Every point on these plots represents the average total hit count over 7 consecutive days. At the end, we chose a training window of 5 days and a test window of 1 day.

captures an additional 6.1 K attackers that the TS alone cannot. This motivates the combination of these three prediction schemes so that they can complement each other and explains the hit count improvement when combining them. Second, adding new schemes in the combination improves the hit count but has diminishing returns, as it is also the case in traditional recommendation systems [5], [20]. In particular, adding EWMA-CA to TS results in a 12% average hit count improvement; adding k NN to the combination TS + EWMA-CA results in only 6% average improvement. This suggests that incorporating additional neighborhood schemes into the equation would likely give modest improvement.

C. Training and Testing Windows

Throughout the paper we used training and testing windows of 5 and 1 days respectively. Fig. 6 shows the performance of

our prediction scheme as a function of the length of these windows and justifies these choices.

We observe that when the training window is too short, the benefit of the time series model is limited by the few available observations. When the training window is too long, it introduces correlation between remote past and recent activities, which was not the case in our data analysis (e.g., Fig. 1(b)). Fig.6(a) clearly shows this trade-off. The performance of our prediction algorithm first increases with the training windows then it decreases when the windows is more than 6-day long. In fact, the curve empirically shows that our scheme achieves the optimal performance when trains on data of 5–6 days.

In Fig. 6(b), we plot the hit count as a function of the length of the testing window. Here, we make two main observations: (1) by increasing the testing window from 1 to 10, the hit count is more than doubled; and (2) this improvement, although quite significant at first, is much smaller than the hit count we would have by running the prediction from scratch every day (dashed line). We also looked at the ratio of the hit count over the upper bound for prediction (omitted for lack of space) and we found that this relative performance metric decreases with the testing window. This indicates that a short testing window is preferable, or in other words, prediction should be trained/refreshed often.

D. Robustness against Pollution of the Training Data

Large-scale repositories that collect firewall and IDS logs from different networks (contributors), such as Dshield.org, are naturally prone to a certain number of false alerts, as the repository has no control over the contributed logs. False alerts may be either due to errors in the configuration of the IDS of a contributor (pollution) or due to a malicious contributor trying to mislead our prediction (poisoning). It turns out that using a combination of diverse prediction methods increases the robustness against both problems.

Pollution. To quantify how random false positives affect the prediction accuracy of our combined method, we artificially generated fake reports, which are distributed over all contributing networks proportionally to the number of real reports they submitted. We vary the amount of total fake reports generated (noise) from 1% to 15%. Fig.7 shows the results. We observe that the hit count decreases slower than the pollution rate,

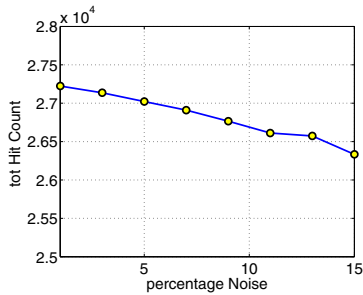


Fig. 7. Robustness of the combined method in the presence of pollution of the training data. The total hit count decreases much slower than the random noise (% of the total number of reports).

e.g., by less than 4% when the pollution rate is 15%. This can be explained as follows. False alerts generated at different networks are unlikely to affect neighborhood models because they usually correspond to different sources reported by different contributing networks, which does not introduce correlation between victims. In order to introduce such correlation, fake reports should have not only the same source but also a similar time stamp to affect the proposed k NN model. Finally, if a source is falsely reported over several days by the same victim, this can affect only the blacklist customized for that specific victim since the time series prediction is specifically computed for each victim network.

Poisoning. Evading our combined prediction is difficult for an attacker and comes at the cost of limiting the attack impact because the attacker must avoid both the time series and the two neighborhood-based predictors. The attacker might limit traffic towards a target network; however, even activities that have low intensity but are persistent over time will be revealed by the time series model. The attacker might also attack different networks for a short time; however, this behavior will be captured by the neighborhood-based models, which focus precisely on this type of behaviors.

VII. SUMMARY AND FUTURE WORK

In this paper, we studied the problem of predicting future malicious activity (through “predictive blacklists”) given past observations. We framed the problem as an implicit recommendation system, which paves the way to the application of powerful machine learning methods. Within this framework, we also proposed a specific prediction method, which is a linear blend of three different algorithms: a time series model to account for the temporal dynamics and two neighborhood-based models. The first neighborhood model is an adaptation of k NN model for attack prediction and focuses on capturing similarity between victims being attacked by the same sources, preferably at the same time. The second is a co-clustering algorithm that automatically discovers a group of attackers that attacks a group of victims at the same time.

We analyzed a real dataset of 1-month logs from Dshield.org, consisting of 100s of millions network security logs contributed by 100s of different networks. We evaluated our proposed algorithms over this dataset and showed significant improvement over the state-of-the-art attack prediction methods. Our combined method improves significantly

not only the prediction accuracy but also the robustness against pollution/poisoning of the dataset.

Despite our performance improvement and methodological development over the state-of-the-art, we believe that this work only scratches the surface of the complicated attack prediction problem. Our analysis shows that even larger improvements can potentially be obtained as there is still a gap between our method and the upper bound. There are several directions for future work: (a) incorporate the effect of other fields of the dataset (such as destination port ID) into our prediction model; (b) add new algorithms in our combination that capture different effects (*e.g.*, latent factor models could capture global behavior); (c) build a prototype.

ACKNOWLEDGEMENTS

We are grateful to P. Barford and M. Blodgett at the University of Wisconsin, Madison, for providing the Dshield dataset. We would also like to thank Michalis Faloutsos for interesting discussions and for bringing the cross-association method to our attention, as well as D. Chakrabarti and C. Faloutsos for making their code publicly available. Our work has been supported by NSF CyberTrust grant 0831530.

REFERENCES

- [1] Dshield dataset, <http://www.dshield.org/>.
- [2] J. Zhang, P. Porras, and J. Ullrich, “Highly predictive blacklisting,” in *Proc. of USENIX Security '08* (Best Paper award), San Jose, CA, USA, Jul. 2008, pp. 107–122.
- [3] SANS Internet Storm Center, <http://isc.sans.org/top10.html>.
- [4] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.
- [5] Netflix Prize, <http://www.netflixprize.com/>.
- [6] G. Linden, B. Smith, and J. York, “Amazon recommendations: Item-to-item collaborative filtering,” *IEEE Internet Computing*, Feb 2003.
- [7] “Google news,” <http://news.google.com/>.
- [8] S. Katti, B. Krishnamurthy, and D. Katabi, “Collaborating against common enemies,” in *Proc. of ACM IMC '05*, Oct. 2005.
- [9] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos, “Fully automatic cross-associations,” in *Proc. of ACM KDD '04*, Seattle, WA, USA, Aug. 2004, pp. 79–88.
- [10] T. Karagiannis, D. Papagiannaki, and M. Faloutsos, “BlinC: Multilevel traffic classification in the dark,” in *ACM SIGCOMM*, Aug 2005.
- [11] A. Ramachandran, N. Feamster, and S. Vempala, “Filtering spam with behavioral blacklisting,” in *ACM CCS*, Alexandria, VA, Oct 2007.
- [12] S. Hao, N. Feamster, A. Gray, N. Syed, and S. Krasser, “Detecting spammers with snare: Spatio-temporal network-level automated reputation engine,” in *18th USENIX Security*, Montreal, Aug 2009.
- [13] P. Barford, R. Nowak, R. Willett, and V. Yegneswaran, “Toward a model for sources of internet background radiation,” in *PAM*, Mar 2006.
- [14] P. B. Z. Chen, C. Ji, “Spatial-temporal characteristics of internet malicious sources,” in *IEEE INFOCOM Mini-Conference*, Apr 2008.
- [15] F. Soldo, A. Markopoulou, and K. Argyraki, “Optimal filtering of source address prefixes: Models and algorithms,” in *INFOCOM '09*. Rio de Janeiro, Brazil: IEEE, Apr. 2009.
- [16] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, “A multifaceted approach to understanding the botnet phenomenon,” in *Proc. of ACM IMC '06*, Rio de Janeiro, Brazil, Oct. 2006, pp. 41–52.
- [17] E. Cooke, F. Jahanian, and D. McPherson, “The zombie roundup: Understanding, detecting, and disrupting botnets,” in *Proc. of USENIX SRUTI '05*, Cambridge, MA, USA, Jul. 2005, pp. 6–6.
- [18] A. Le, “Technical report: Analyzing dshield logs using fully automatic cross-associations,” <http://www.ics.uci.edu/~anhml/publications.html>.
- [19] J. Elder, “Fusing the results of diverse algorithms,” in *Proceeding of the 3rd International Conference on Multi-strategy Learning*, 1996.
- [20] R. Nisbet, J. Elder, and G. Miner, *Handbook of statistical learning and data mining applications*. Elsevier, 2009.