

Blacklisting Recommendation System: Using Spatio-Temporal Patterns to Predict Future Attacks

Fabio Soldo, *IEEE Student Member*, Anh Le, *IEEE Student Member*, and Athina Markopoulou, *IEEE Member*

Abstract—In this paper, we study the problem of forecasting attack sources based on past attack logs from several contributors. We formulate this problem as an implicit recommendation system, and we propose a multi-level prediction model to solve it. Our model evaluates and combines various factors, namely: (i) attacker-victim history using time-series, (ii) attackers and/or victims interactions using neighborhood models and (iii) global patterns using singular value decomposition. We evaluate our combined method, referred to as Blacklisting Recommendation System (or BRS), on one month of logs from Dshield, and we demonstrate that it improves significantly the prediction rate over state-of-the-art methods as well as the robustness against poisoning attacks. Along the way, we analyze the Dshield dataset, and we reveal dominant patterns of malicious traffic.

Index Terms—Prediction Algorithms, Network Security, Internet, Machine Learning

I. INTRODUCTION

NETWORK operators routinely maintain and process security logs, *e.g.*, produced by firewall and intrusion detection systems, in order to monitor and prevent malicious traffic from entering, or being generated inside, their networks. Such security logs provide a significant part of the digital evidence that is necessary to perform forensics analysis.

Based on this data, blacklists with the most prolific attack sources are compiled and used to take action against the blacklisted traffic. For example, traffic originating from a blacklisted source may be logged, further analyzed, or even dropped at the edge of a network. Examples of blacklists include IP and DNS blacklists that help to block unwanted web content, spam producers, and phishing sites. Sites, such as DShield.org [1], process firewall and intrusion detection system logs contributed by hundreds of victim networks worldwide and publish blacklists of attack sources based on those logs.

Blacklists essentially attempt to forecast future malicious sources based on past logs. It is desirable that they are predictive, *i.e.*, that they 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 length 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.

Manuscript received 1 August 2010; revised 23 January 2011. This work was supported by NSF CyberTrust grant 0831530.

F. Soldo and A. Markopoulou are with the Electrical Engineering and Computer Science Department, University of California, Irvine, CA 92697 USA (e-mail: fsoldo@uci.edu; athina@uci.edu).

A. Le is with the Information and Computer Science Department, University of California, Irvine, CA 92697 USA (e-mail: anh.le@uci.edu).

Digital Object Identifier 10.1109/JSAC.2011.110808.

We consider a set of networks that (i) log malicious activity at their edge and (ii) share flow-level information from their logs with each other or with a central repository such as Dshield. The purpose of this collaboration is to enable better analysis of the shared logs and eventually better prediction of future attacks. Given these past security logs from many contributing networks, our goal is to analyze and predict malicious behavior at the IP level. More specifically, we analyze the following information about each logged flow: (i) the attack IP address (source of malicious flow), (ii) the victim network (destination of malicious flow) and (iii) the time (when the malicious flow was logged). We predict future malicious activity based on the past, and we construct a customized predictive blacklist for each victim network. To achieve this goal, we take the following steps.

First, we formulate the problem using a methodological framework inspired by recommendation systems [2]–[5]. More specifically, we frame the problem of predictive blacklisting (*i.e.*, predicting which networks will be targeted by which attacker based on historical data) as an implicit recommendation system (*i.e.*, which items will be chosen by a user given the history of past user's actions). This novel formulation opens the possibility to apply powerful methodologies from machine learning to this problem.

Second, within the recommendation system framework, we evaluate, compare, and combine several candidate techniques, namely: time-series analysis (to capture history and temporal trends in the logs), two neighborhood models (to capture similarity of attackers and/or victims) and singular value decomposition (to capture global spatial patterns). The end-result is a combined method, which we call *Blacklisting Recommendation System* or *BRS*. BRS predicts the likelihood of a particular attack source attacking a particular victim network. A customized blacklist per victim network can then be constructed by selecting the most likely attack sources to attack that particular victim network.

Third, we thoroughly analyze one month of logs from Dshield, and we reveal dominant spatio-temporal patterns of malicious traffic. Apart from being informative on their own merit, these patterns also guide the design of our blacklisting recommendation system. We evaluate BRS over the Dshield dataset, and we demonstrate that it improves (i) the predictiveness of the blacklists, by up to 70% over state-of-the-art approaches [6], and (ii) the resilience to noise and poisoning attacks.

The rest of the paper is organized as follows. Section II discusses related work and puts our work in perspective. Section III presents the key spatio-temporal patterns we identified in the Dshield dataset. Section IV formulates our problem

in the recommendation system framework. It also motivates this study by showing the gap between current state-of-the-art approaches and the upper bound (achieved by an offline algorithm). Section V presents the specific temporal (time-series) and spatial (two neighborhood models and a singular value decomposition) methods that we use to capture various patterns in the data and eventually to perform prediction. Section VI evaluates the individual methods and their combination over the *Dshield* dataset and demonstrates that the combined method (BRS) significantly outperforms the current state-of-the-art approach. Section VII outlines future directions and Section VIII concludes the paper.

II. RELATED WORK

Closely Related Blacklisting Techniques. The two traditional approaches to generate blacklists are GWOL and LWOL, according to the terminology of [6]. LWOL stands for “Local Worst Offender List”: security devices deployed on a specific site log malicious activity, and a blacklist of the most prolific attack sources is compiled. This 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. 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], [7]. A problem with this approach is that the globally most prolific attack sources might be irrelevant to some specific victim networks.

Beyond the traditional approaches, Zhang *et al.* proposed “Highly Predictive Blacklisting” (HPB), [6]. The main idea of HPB is that a victim should predict future attackers based not only on her own logs but also on logs of a few other “similar” victims. In [6], 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. The HPB scheme is implemented in *Dshield* and is provided as a service to contributors: operators contribute daily logs from their network to *Dshield* and get back in return a customized blacklist that helps them to better defend against future attacks to their networks.

Compared to HPB [6], our work solves the same problem (predictive blacklisting based on shared logs), but we have several differences in methodology and intuition. More specifically, we make the following contributions compared to [6]. (1) We formulate the problem as an implicit recommendation system [2]; this opens the possibility to apply a new set of powerful techniques from machine learning. Within the recommendation system framework, we combine a number of techniques that capture and predict different behaviors present in our dataset. (2) One set of techniques exploits *local spatial correlation*, *i.e.*, identifies similar victims and/or attackers. HPB is a special 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 similarity that takes into account blocks of attackers and victims jointly, using a co-clustering algorithm called cross-association (CA) [9]. (3) We also investigate the use of singular value decomposition to identify *global spatial patterns*. (4) Another set of techniques use time series to exploit *temporal trends* for prediction; we also incorporate the time aspect at various parts of the model. This includes LWOL as a special case, where the past consists of a single time period.

Other Work on Blacklisting. Blacklists are widely used to deal with several types of malicious activity. For example, IP and DNS blacklists help to block unwanted web content, spam producers, and phishing sites. Below we mention some examples.

In the context of spam, Spamhaus is a project that tracks spammers on the Internet [10]. It provides dependable realtime anti-spam protection by publishing blacklists of spamming sources and works with law enforcement to identify and pursue spammers. One of the widely used blacklists compiled by Spamhaus is the Exploits Block List (XBL). XBL is a list of IP addresses of illegal 3rd party exploits, including open proxies (HTTP, socks, AnalogX, wingate, etc), worms and viruses with built-in spam engines, and other types of trojan-horse exploits. XBL incorporates data from two highly-trusted DNS blacklist sources, with tweaks by Spamhaus to maximize the data efficiency and minimize false positives.

In the context of defense against phishing attacks, Prakash *et al.* [11] proposed PhishNet, a simple algorithm to predict future malicious URLs. In this work, based on the observation that attackers often make only slight modifications to old phishing URLs to generate new ones, the authors enumerate combinations and variations of known phishing sites to create new ones. In [12], [13], the authors proposed the use of online learning to extract features of known phishing URLs, build a classification model for phishing URLs, and use this model to predict new malicious URLs.

Several researchers have looked at the effectiveness of various types of blacklists. In [14], Jung and Sit study spam traffic going to an institution network and find that 80% of the spam sources appear in at least one of seven popular blacklists at the time. In [15], Ramachandran *et al.* propose counter-intelligence techniques to detect botmasters based on DNS blacklists. In particular, the authors exploit the fact that botmasters themselves perform DNS blacklist lookups to determine whether their spamming bots are blacklisted to identify potential bots. In [16], Sinha *et al.* evaluate four popular reputation-based blacklists on an academic network with 7,000 hosts. The authors find that the false negative rates are high: ranging from 35% to 98%, and that the false positive rates are lower but some rates are still significant: ranging from 0.2% to 9.5%.

Prediction vs. Detection. Our work falls within the category of behavioral analysis, in the sense that inferences are made based on flow logs and traffic patterns at the network layer, rather than on deep packet inspection and packet payload. This work focuses on prediction and not

on traffic classification [17] or detection (*i.e.*, distinguishing legitimate from malicious traffic such as [18], [19]). In other words, we work with flow logs that have already been labeled as malicious by a different module, such as an intrusion and detection system, and are given as input to our prediction algorithms. However, as discussed in Section VI-D, BRS is robust to noise in the labeled data.

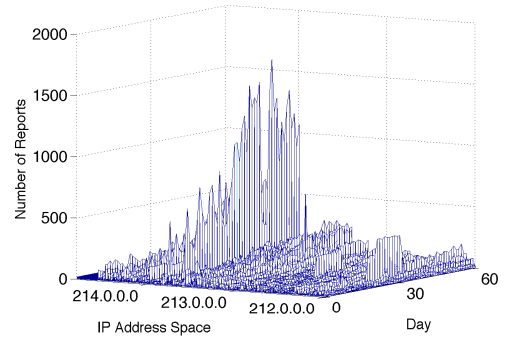
Dshield Data. Our data analysis and performance evaluation is based on data from the Dshield repository. An advantage of using this dataset is that it is a rich dataset from a diverse set of contributors. Furthermore, Dshield data are available to the research community and have been used by many researchers in this area, including [6], [8], [20], [21], thus making Dshield an appropriate baseline for comparison. Indeed, some of the observations we make during our data analysis were consistent with the findings of previous studies. For example, the fact that victims share “common enemies” was first observed in [8] and used in [6]; or the short term memory of attackers was also observed in [20], [21].

On the other hand, the Dshield dataset has its limitations. First, we have no control over the configuration of the detection systems that generate the logs contributed to the repository. Therefore, it is possible that there are false positives in the datasets. Although BRS does not address the detection part and uses the Dshield data as input, we partially address this problem by pre-processing the logs (as explained later) and by combining various techniques (which make BRS robust to noise). Second, we have access only to flow logs, not to the detailed alerts raised by the intrusion and detection systems, which could provide more information and context.

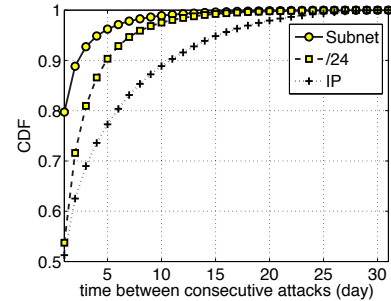
Recommendation Systems. Our problem formulation is inspired by recommendation systems [2], which currently find applications on e-commerce web sites, such as, Netflix [3] and Amazon [4], as well as on other areas such as Google News [5]. An explicit recommendation system uses the explicit rating provided by users on users to represent their interests. The problem of attack prediction is best modeled as an implicit recommendation system, where ratings are inferred (not given explicitly) by observing malicious activity, and recommendations are provided to victims about what addresses to block in the future.

One important complication in our case is that malicious activity varies over time. Accounting for temporal dynamics is currently an active research area in the recommendation system community [22] and the traditional recommendation system techniques are not directly applicable. We had to come up with novel ways in order to take time into account in the prediction methods.

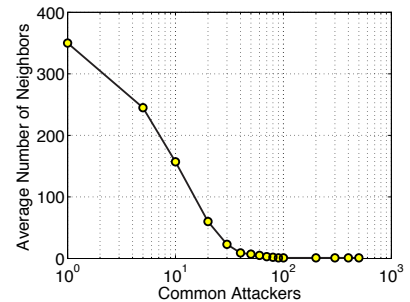
Another set of related techniques are those developed in compressive sensing [23]. Compressive sensing allows to reconstruct a signal from few samples. When applied to matrices these techniques allows to reconstruct the missing entries when $O(n^{1.2} \log n)$ entries of an $n \times n$ matrix are observed [24]. These results are very powerful; however, they require that a certain number of entries for each row and column are observed. Our problem does not satisfy these assumptions as we are trying to predict future events, where no information is already observed, and thus falls into a different category of problems.



(a) A sample of malicious activity



(b) Temporal behavior: inter-arrival time of the same source appearing in the logs



(c) Common attackers among different victims

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

Our Prior Work. An earlier version of this work appeared in [25]. Compared to [25], new materials include: (i) an extended evaluation of the robustness of our techniques to poisoning attacks and colluding attackers; (ii) the analysis of the global spatial patterns via singular value decomposition (SVD); (iii) a more extensive data analysis.

III. THE DSHIELD DATASET: OVERVIEW AND KEY CHARACTERISTICS

In this study, we used logs from Dshield 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 [1] is a repository of firewall and intrusion detection logs collected at hundreds of different networks all over the Internet. The participating networks contribute their logs, which are then converted into a common

format that includes 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 preprocessed our dataset to reduce noise and erroneous log entries, such as, those belonging to invalid, non-routable, or unassigned IP addresses. Data from `Dshield` have been studied and used by several researchers over the years, such as [6], [8], [20], [21], [26] to name a few examples, and the main findings (e.g. clustering of malicious sources and short-lived IP addresses in the logs) were consistent over time and with our own analysis.

We analyzed 6 months of `Dshield` logs, from May to October 2008. In this paper, we present results for 1-month only (October 2008); the results were quite similar for the other months. The pre-processed 1-month dataset consists of about 430,000,000 log entries, from ~ 600 contributing networks, with more than 800,000 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.

Secondly, 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 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. 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 *moving average 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. Let us 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 neighbors, 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.

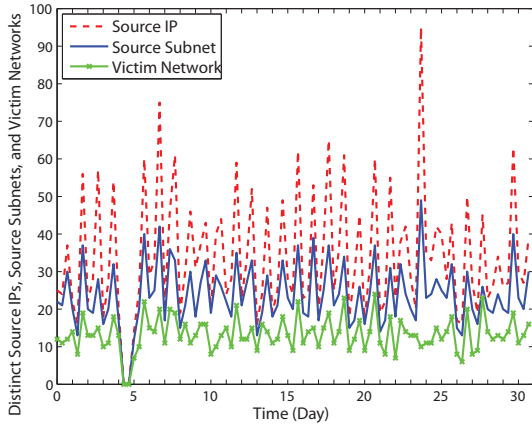
Finally, we focused on mining persistent coordinated enemies from the `Dshield` logs and provide sufficient accompanying evidence showing that the flow-level patterns we identify are not a coincidence. To this end, we take the following three steps. First, we view the activities from source hosts to victim hosts as a binary matrix. We use this matrix as input to the cross-association (CA) algorithm [9] for every day of the month. From results of the CA algorithm, we identify several dense clusters of coordinated attackers: attackers in the same cluster attack the same set of victim hosts (Sec. V. B2). Second, we analyzed the temporal persistence of each dense cluster by measuring how similar it is to another dense cluster in a different day throughout the whole month. Third, we examined attack activities, as indicated by ports, protocols, and time, to find additional evidence of coordination among attack sources in the same group.

Fig. 2 plots the attack activities by the most persistent dense cluster. Fig. 2(a) depicts activities associated with port 445. Port 445 is a well-known port (Windows Directory Service port), and received the most number of attacks throughout the month. The plot shows the number of distinct source IPs, source subnets, as well as victim networks over time. In total, there are 158 source subnets, distributed across 32 different countries, which attack 263 victim networks. Although both the attackers and victims are distributed across different time zones, there is a clear diurnal pattern of their attack activities: the number of attackers starts out small at the beginning of the day, grows to a peak around the midday, and decreases by the end of the day during the whole month. Similar diurnal patterns also appear in our analysis of activities associated with other active well-known ports, such as, ports 139 and 1433. Fig. 2(b) plots attack activities associated with port 59812, an undocumented port. There is a large number of well-spread attackers associated with this attack: 179 source subnets, from 35 different countries; moreover, they attack a very small number of victim networks: 6 networks, from 3 different countries of 3 different continents: US, Europe, and Asia, in a very short amount of time. We also notice that there is a short 10-minute interval of day 27 during which this undocumented port of two different victims networks are exploited at the same time.

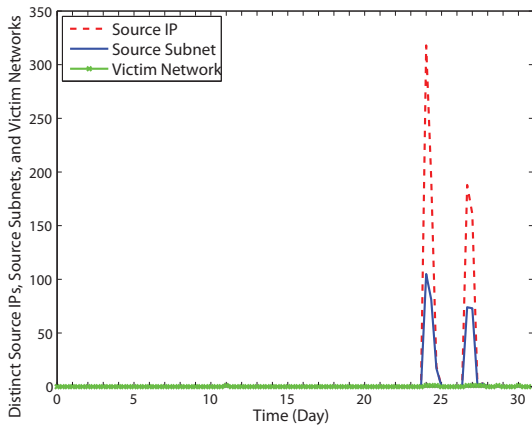
In summary, we analyze attack activities associated with both well-known and undocumented ports using ports, protocols, and time. We find strong evidence of persistent coordination of the candidate attack clusters, which are identified earlier by CA algorithm. This further supports our findings of coordinated attackers.

IV. PROBLEM FORMULATION AND FRAMEWORK

Our goal is to predict future malicious IP traffic based on past logs contributed by multiple victims. Predicting malicious



(a) Most Active Well-Known Port - 445



(b) Active Undocumented Port - 59812

Fig. 2. Plot (a) and (b) show attack activities associated with port 445 and 59812 of subnets of the most persistent cluster, respectively. Port 445 is the most active well-known port while port 59812 is an active but undocumented port. Plot (a) shows clear diurnal pattern of activities. Plot (b) shows several high spikes that involve up to 100 subnets targeting port 59812 of 6 victim networks, sometimes 2 different networks within 10 minutes, which also indicates persistent coordination.

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. 3. Recommendation systems aim at inferring unknown user rating about items from known (past) ratings. An example is the Netflix recommendation system, Netflix Cinematch [3], which aims at predicting unknown user ratings about movies from known ratings, in order to provide movie recommendations to its customers. Other examples of real deployment of recommendation systems include the Amazon recommendations [4], Google news personalization [5], TiVo, GroupLens, to mention a few. 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

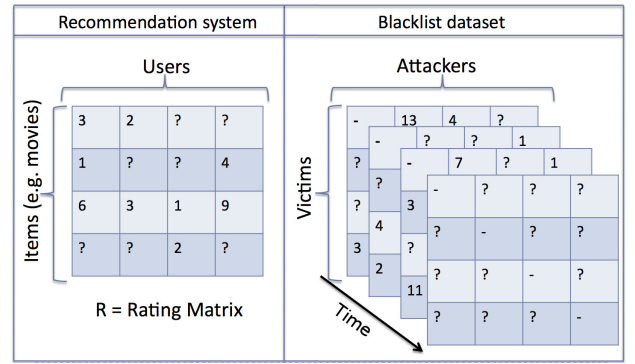


Fig. 3. Analogy between Recommendation Systems (left) and Attack Prediction (right). The former infers unknown user ratings about items from known ones. The latter deals with time varying ratings.

superimposed processes, some of which are intuitive, while others need to be unveiled and confirmed through an accurate analysis of the dataset. For instance, a user can give on average a higher score to drama movies than to comedy, or vice versa, according to her preferences; or he can rate differently on weekdays than on weekends, which might reflect his state of mind [22].

We also aim at predicting 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: higher r indicates that an attacker intensifies its efforts to attack a specific victim, thus implying a higher preference. There are some important differences from traditional recommendation systems. 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 recommendation system 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 recommendation system 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{U} be the set of users (customers) and \mathcal{I} be the set of items. In practical applications, these two sets are usually very large, including up to hundreds of thousands of elements. 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 $|\mathcal{U}|$ -by- $|\mathcal{I}|$ rating matrix.

2) *Problem Formulation*: A recommendation system aims at inferring unknown ratings from known ones, as shown in

Fig. 3-left. Ultimately, the goal of recommendation systems is to find for every user u , the item, i_u , that has the highest rating [2]: $i_u = \arg \max_{i' \in \mathcal{I}} r_{ui'}$, $\forall u \in \mathcal{U}$. What makes the recommendation system problem difficult is that the values $r_{u,i}$'s are unknown. We only know a limited subset of ratings because a user generally does not rate all available items but only a subset of them, typically orders of magnitude smaller than the total number of available items.

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 recommendation system problem can be formalized as follows:

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

The recommended item, i_u , for user u , maximizes Eq. (1) and may be different for every user. The solution of Eq. (1) is usually obtained by first estimating the matrix R on the subset $\bar{\mathcal{K}}_u$, 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*: Let \mathcal{A} be the set of attackers (*i.e.*, source IP prefixes where attacks are launched from) and \mathcal{V} be the set of victim networks. Let t indicate the time an attack was performed according to its log. 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 recommendation system problem, we define a 3-dimensional rating matrix R so that per every tuple (a, v, t) , $r_{a,v}(t)$ represents the number of attacks reported on day t from $a \in \mathcal{A}$ to $v \in \mathcal{V}$. We denote with B the binary matrix that indicates whether or not an attack occurred: $b_{a,v}(t) = 1$ if $r_{a,v}(t) > 0$, and $b_{a,v}(t) = 0$ 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 $\tilde{\mathcal{BL}}$ be any set of N different attackers. The problem of attack prediction can be formalized as follows:

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

The output of the attack prediction problem is a set of blacklists, $\mathcal{BL}(v) = \{a_1^v, a_2^v, \dots, a_N^v\} \subset \mathcal{A}$, 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 an entire $|\mathcal{A}|$ -by- $|\mathcal{V}|$ matrix to be estimated before the max operation can be performed, as illustrated in Fig. 3-right. In this sense, this problem is a generalization of the recommendation problem, Eq. (1), where R is now defined on 3-dimensional space, $\mathcal{A} \times \mathcal{V} \times T$, rather than a 2-dimensional space. While the recommendation system problem traditionally estimates missing elements in a matrix, the attack prediction problem estimates matrixes in a tensor.

Finally, we observe that per 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 [6]: 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 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. Under this assumption, 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**, 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 , only has access to its local security logs, but not to the logs of other victims. This is a very typical

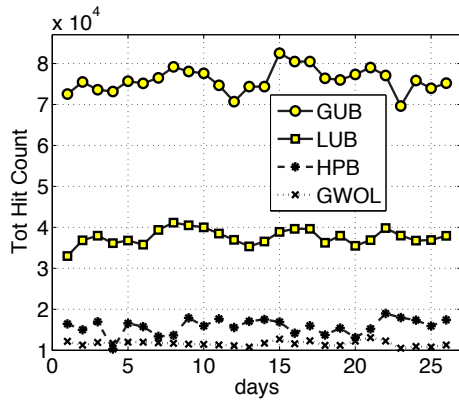


Fig. 4. Comparing different prediction strategies (in terms of total hit-count) on 1-month of Dshield logs. Observe that the state-of-the-art HPB improves over baseline, GWOL, but there is still a large performance gap until the upper bounds, LUB and GUB.

case in practice today. Because $A_v(T_{train}) \subseteq A(T_{train})$, the following inequality holds trivially: $LUB(v) \leq GUB(v)$. At the end of this section, we will quantify this gap on a real dataset of malicious sources.

State-of-the-art. 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 state-of-the-art HPB [6]. These methods have been described in detail in the previous Section II.

Room for improvement. In Fig. 4, we compare the total hit count (for all victims in the system) of different prediction strategies on 1-month of Dshield logs. For a fair comparison, we require all methods to use the same blacklist length N . To be consistent with [6], we set $N = 1,000$ 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 [6]. 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 this 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 dy-

namics affect the ratings. In recommendation systems, ratings are given at different times, but once given they cannot be changed. The goal is to use the known ratings as ground truth and estimate the missing ratings based on them. In contrast, in the attack prediction problem, “ratings” vary rapidly 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 $r_{a,v}(t)$ 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 short 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, which predicts future values based on past values weighted with exponentially decreasing weights toward older values.

We indicate with $\hat{r}_{a,v}^{TS}(t+1)$ the predicted value of $r_{a,v}(t+1)$ given the past observations, $r_{a,v}(t')$, at time $t' \leq t$. We estimate $\hat{r}_{a,v}^{TS}(t+1)$ as

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

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.

We note that Eq. (5) include as a special case the LWOL when the training window has only one day, $t = 1$. In this case, the EWMA model gives the highest weight to attackers that were most prolific in the last day, as in the LWOL strategy. However, the general EWMA model has higher flexibility that can model temporal trends. 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.

We also observed that an improved prediction accuracy can be obtained when applying the same EWMA model to the binary version of R, B . This is because attackers that performed a large number of attacks in the recent past are not necessarily the most likely to attack also in the future. An attacker can stop its activities at any time. Moreover, the number of reports is also very sensitive to the specific configuration of the victim’s intrusion and detection system, and thus it is prone to configuration errors. The group of attackers that is more likely to keep on attacking is the one that was continuously reported as malicious for a large number of days independently from the number of reports. Therefore, an improved forecast can be obtained based on B :

$$b_{a,v}^{TS}(t+1) = \sum_{t'=1}^t \alpha_{a,v}(1-\alpha_{a,v})^{t-t'} b_{a,v}(t'), \quad (6)$$

where $\alpha_{a,v}$ is the smoothing coefficient trained for each specific pair (attacker, victim), and $b_{a,v}^{TS}(t+1)$ indicates the forecast for $b_{a,v}(t+1)$ and can be interpreted as a measure of how likely an attacker is to attack again given its past history.

In the rest of this paper, we will use the improved forecast based on B when we mention the Time Series (TS) method.

B. Neighborhood Model

The strategies 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. A commonly used metric is the Pearson correlation, which generalizes the notion of cosine distance of vectors with non-zero mean. Formally, given two n -dimensional vectors, x, y , with mean values, m_x, m_y , respectively, the Pearson correlation of x and y is defined as

$$s_{xy} = \frac{\sum_{i=1}^n (x_i - m_x)(y_i - m_y)}{\sqrt{\sum_{i=1}^n (x_i - m_x)^2} \sqrt{\sum_{i=1}^n (y_i - m_y)^2}}. \quad (7)$$

We observe that for zero mean vectors, this reduces to $s_{xy} = \frac{x \cdot y}{\|x\|_2 \|y\|_2} = \cos(x, y)$.

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

$$s_{uv} = \sum_{t_1 \leq t_2 \in T_{train}} e^{-|t_2 - t_1|} \frac{\sum_{a \in \mathcal{A}} b_{a,u}(t_1) \cdot b_{a,v}(t_2)}{\|b_u(t_1)\|_2 \|b_v(t_2)\|_2}, \quad (8)$$

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 similarities. In other words, our spatio-temporal notion of similarity includes the spatial notion of similarity as special case, for $t_1 = t_2$.

Eq. (8) captures our intuition that victims attacked by the same sources simultaneously or within a short time interval, are more similar to each other than victims attacked by common sources but at different times. In the former case, they are more likely affected by the same type of attack. In the latter case, the same attack source might have engaged in different type of attacks at different times and as a result the victim networks might not have the same characteristics and vulnerabilities. Furthermore, malicious flows originating

by the same source almost at the same time, can also be the result of the victim hosts being consecutive in the hit list of a scanning or other bot. Thus, giving higher importance to attacks occurring in the same time slot captures a stronger correlation among victims than just using the number of common attackers. When two victims, u and v , report attacks by the same attacker at different times, the smoothing factor, $e^{-|t_2 - t_1|}$, accounts for the interval time between the two attacks and discounts their similarity, s_{uv} , accordingly.

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:

$$b_{a,v}^{kNN}(t) = \frac{\sum_{u \in N^k(v;a)} s_{uv} b_{a,u}(t)}{\sum_{u \in N^k(v;a)} s_{uv}}, \quad \forall t \in T_{test} \quad (9)$$

where, $b_{a,v}^{kNN}(t)$ 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 $b_{a,u}(t)$ is known.

In order to compute $b_{a,v}^{kNN}(t)$, we need two main ingredients: a similarity measure between victims, s , and a set of known rating for the attacker a , $b_{a,u}(t)$. What prevents us from a direct application of Eq. (9) is that none of the ratings, $b_{a,u}(t)$, 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. (6):

$$b_{a,v}^{kNN}(t) = \frac{\sum_{u \in N^k(v;a)} s_{uv} b_{a,u}^{TS}(t)}{\sum_{u \in N^k(v;a)} s_{uv}}, \quad \forall t \in T_{test}, \quad (10)$$

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 [27], [28]: 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 intrusion and detection system, enables the victim to preemptively put the other “similar” machines of the botnet into his blacklist.

To find similarities among both victims and attackers simultaneously, we apply cross-association (CA) algorithm [9] – a fully automatic clustering algorithm that finds row and column clusters of sparse binary matrices. In this way, we find clusters of both similar victims (contributors) and attackers (/24 subnets.) Fig. 5 depicts the result of applying the CA on a contributor-subnet matrix of 1-day log data. On average, the CA finds over 100 clusters per day.

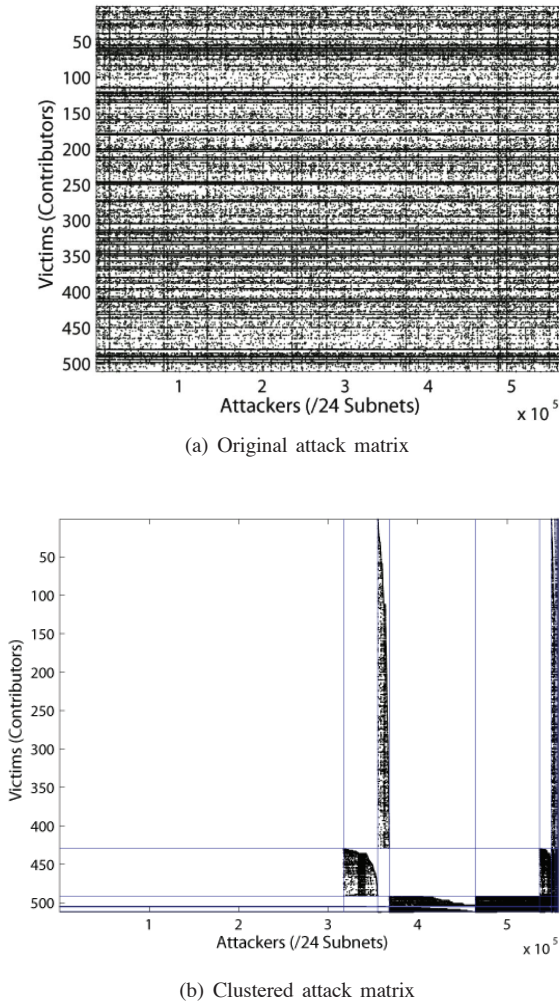


Fig. 5. Result of applying the CA algorithm on 1-day Dshield.org logs. A rectangular block indicates a cluster of similar sources and victims identified by the CA.

For each cluster (depicted as a rectangular block in Fig. 5), 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 cluster to quantify the strength of correlation among the attackers and victims within the cluster. Intuitively, a dense cluster corresponds to an attacker-victim bipartite graph that resembles a complete bipartite graph, thus indicating strong correlation. Finally, we use this density for forecast: the denser a cluster is, the more likely its attackers will attack its victims.

More formally, $\tilde{b}_{a,v}^{CA}(t+1) = \rho_{a,v}(t)$, where $\rho_{a,v}(t) \in [0, 1]$ is the density of the cluster 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 correlation 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 again the victim than other attackers. Formally,

$$b_{a,v}^{CA}(t+1) = \sum_{t'=1}^t \alpha(1-\alpha)^{t-t'} \rho_{a,v}(t'). \quad (11)$$

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

C. Latent Factors Model

In the previous section, we investigate local interaction between different attack traces: each attack is interpreted as a time signal and correlated with other signals based on their activities. A different and complementary approach is to build a global model for the entire attack matrix. Let $B_{|a|,|v|} = P_{|a|,|k|} Q_{|k|,|v|}$ be the Singular Value Decomposition (SVD) of B [29]. Each entry of B can be determined as the result of a dot product:

$$b_{a,v}^{SVD} = \bar{p}_a \cdot \bar{q}_v^T = \sum_{k=1}^{|k|} p_{a,k} q_{k,v} \quad (12)$$

where \bar{p}_a, \bar{q}_v are the singular vectors associated with the space of attackers, and victims respectively. This has the nice interpretation of mapping the space of attackers and space of victims, which are naturally different and cannot be compared directly, to a common “latent” space where a direct comparison is possible.

The latent space corresponds mathematically to the space spanned by the singular vectors of B . Moreover, for every fixed $k \leq \text{rank}(B)$, the truncated SVD of B is guaranteed to provide the best rank- k approximation of B ¹. The motivation to use this method is that most attacks do not happen at random but they are determined by a few factors, such as, vulnerabilities at the target host, OS, type of services, that are not always directly measurable through network-level traces. These factors characterize both the attackers (*e.g.*, how likely they are to launch a specific type of attack), and the victims (*e.g.*, how vulnerable they are to that specific exploit.) Thus, the likelihood of each attack can then be regarded as the dot product of these two quantities.

Similar techniques has been successfully applied in different areas, such as, collaborative filtering, e-commerce, and advertising [4], [22]. When applying SVD models to the attack prediction problem, we face issues that are specific to this application: dealing with IP addresses (or subnets), the size of the attack matrix is generally much larger than in other applications [3]; we do not have a static matrix but instead a sequence of them observed at different times (the time dimension is a critical factor in this analysis); and the entries we are trying to predict are not missing uniformly at random as in other works [24] but represent *all* future entries.

D. Combine Predictors

The combination of different predictors (either obtained from different methods, or with the same method trained on different subset of the data) 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” [30]. 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

¹more precisely, the rank- k approximation of B that minimizes the Frobenius norm of the error.

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 [31]. 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, the CA clustering algorithm to model persistent clusters of attackers-victims, and the SVD model to capture latent factors.

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 weigh the neighborhood models with weights proportional to their local 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:

$$\hat{b}_{a,v}(t) = b_{a,v}^{TS}(t) + w_{a,v}^{kNN} b_{a,v}^{kNN}(t) + w_{a,v}^{CA} b_{a,v}^{CA}(t) + b_{a,v}^{SVD}(t) \quad (13)$$

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

Data set. We evaluate the performance of our prediction algorithm using one-month of real logs on malicious IP sources provided by Dshield, as described in Section III.

Metrics. We use two different metrics to evaluate the predictiveness of the blacklisting methods: the total hit count and the prediction rate. The *hit count* was defined in Section IV-D and it 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 contributors. The *prediction rate* is the ratio between the hit count and the global upper bound for each contributor separately. Thus, for each contributor, the prediction rate is a number in $[0, 1]$, which represents the fraction of attackers correctly predicted out of the global upper bound of the contributor, which is the maximum number of attackers that can be predicted based on past logs. A prediction rate equal to 1 indicates perfect prediction.

Parameters. We call the time period (in the recent past) over which logs are analyzed in order to produce the blacklist

the training window. We call the time period (in the near future) for which the forecast is valid *the testing window*. 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 [6], each predictive blacklist specifies /24 IP prefixes, which is also often the case in practice [1]. However, we note that the methodology described here applies to any granularity of IP address/prefix considered.

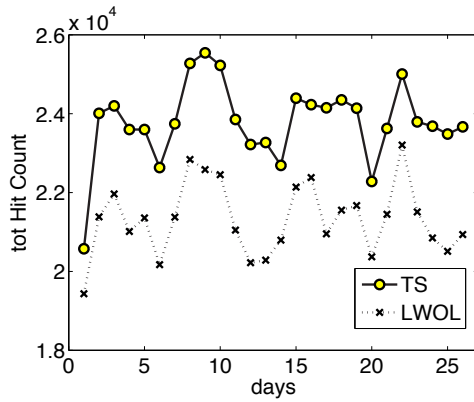
Complexity. The complexity of the combined prediction depends on the complexity of the individual methods. Computing the TS prediction requires $O(T_{train})$ operations for each rating $r_{a,v}^{TS}$. Thus, its overall complexity is $O(T_{train}|\mathcal{A}||\mathcal{V}|)$. The complexity of the k NN model is the computation of the similarity matrix $O(T_{train}|\mathcal{V}||\mathcal{V}|)$ plus the complexity of computing Eq.(9) for every pair (a,v) , that is $O(k|\mathcal{A}||\mathcal{V}|) = O(|\mathcal{A}||\mathcal{V}|)$ since k is a constant. 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 . Finally, a full SVD decomposition is computed in $O(|\mathcal{A}||\mathcal{V}|^2)$. This is further reduced using iterative algorithms that computes only the top k singular vectors of interest. 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

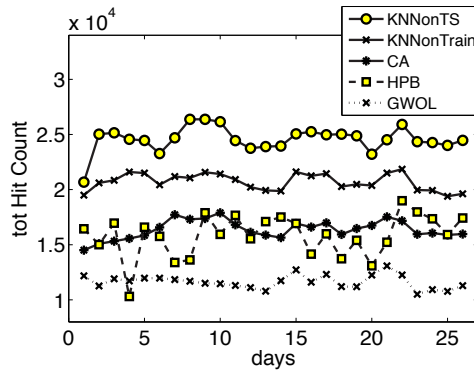
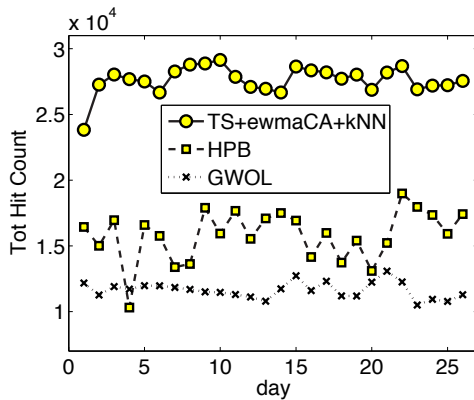
We group prediction schemes in two categories depending on whether they use local or global information. In the local category, there are the time series, TS, and LWOL, since they both use only local logs available at each network. In the global category belong the neighborhood models, such as k NN,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

In Fig. 6(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 includes LWOL as a special case when the training window is equal to a single time slot.

In Fig. 6(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 [6], 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 across 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. (9) (where k NN is run on top of the last day’s logs), “ k NN on TS” in Eq.



(a) Local approaches: TS and LWOL

(b) Global (neighborhood) approaches : k NN (“on TS”, “on Train”), EWMA-CA, GWOL

(c) Proposed combined method (TS+KNN+CA) vs. state-of-the-art (HPB) and baseline (GWOL)

Fig. 6. Evaluating the performance (total hit count) of different individual methods, our proposed combined method (TS+KNN+CA) and baselines methods.

(10) (where k NN is run on top of the TS predictions). We set $k = 25$. k NN schemes outperforms 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 further improvement.

We applied the SVD model of Eq.(12) to the matrix obtained as the combination of B at different times and TS . To this end, we used a similar technique as in [32]: each instance of the matrix B (and TS) is reshaped in a single column vector by concatenating the different columns. Each vector is then associated with a different time. The final matrix is obtained by placing these vectors in increasing order $[\mathcal{B}(t_1), \mathcal{B}(t_2), \dots, TS]$,

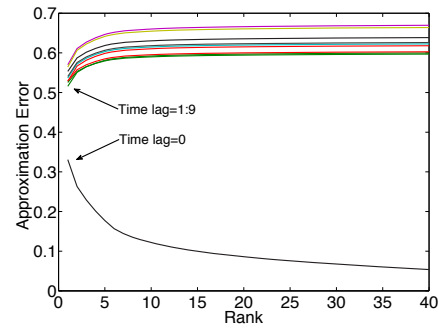


Fig. 7. SVD approximation error for different values of rank, k of the truncated approximation. When we compute the error between the SVD approximation and the real dataset of the same day (time lag = 0), we observe that as expected the error decreases as k increases. However, as soon as the time lag is greater or equal than 1 (day), the approximation error tend to increase with k . This suggest that it is not possible to identify a low-rank approximation for the entire dataset that is persistent over time, *i.e.* has a small approximation error across different days.

where $\mathcal{B}(t_i)$ indicates the matrix $B(t_i)$ reshaped in a column vector.

Combining the SVD prediction to the time series improved the prediction accuracy. However, this gain was still smaller than the relative gain provided by the neighborhood models. Moreover, when adding the SVD predictions on the top of the combination of time series and neighborhood models, we did not observe a significant improvement. This is probably due to the fact that attacks change rapidly over time and thus the SVD decomposition fails in capturing a stable low-rank approximation. To verify this intuition, in Fig. 7 we plot the approximation error between the SVD decomposition for different values of rank k and across different days. The error is defined as the ratio: $\frac{\|B(t_i) - B(t_j)_k\|_F}{\|B(t_i)\|_F}$, where $\|\cdot\|_F$ is the Frobenius norm, and $B(t_j)_k$ is the SVD approximation of $B(t_j)$ of rank k , and $|i - j|$ is the time lag. When the time lag is greater or equal than 1 the approximation error is smaller for small values of k . We know that, for larger values of k , $B(t_j)_k$ tends to better approximate $B(t_j)$. However, the fact that the approximation error increases with k when the time lag ≥ 1 shows further evidence that the short-term temporal dynamics significantly affect the entire dataset, *i.e.*, most of the attacks are short-term, and this results in observing very different datasets at different times.

For these reasons, we do not include SVD in the combined predictor, as most of the benefit comes from other temporal and local neighborhood methods, which are also lower complexity.

In Fig.6(c), we show the total hit count achieved by our proposed combined scheme of Eq. (13), which blends together TS, k NN (on TS) and (EWMA) CA and we compare it to the state-of-the-art method (HPB). This figure shows essentially the main result of this paper. 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 size. *E.g.*, TS and EWMA-CA successfully predict a common set of 9.9 K attackers; however, EWMA-CA alone 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 [3], [31]. 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.

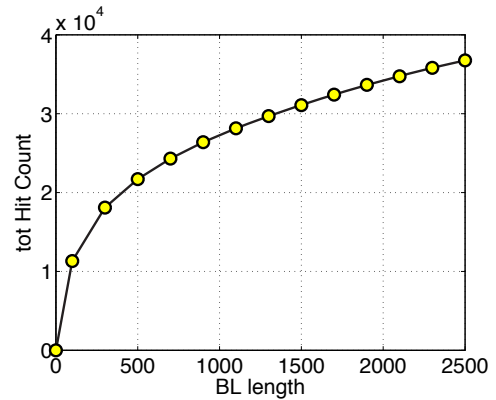
In Fig. 8 we analyze the performance of our algorithm as a function of the blacklist length. As we expect both the hit count and the prediction rate increases with the blacklist length. The largest relative increase occurs with a blacklist of length 3–500. In fact, a blacklist of length 500 has on average a prediction rate of about 50%. While a blacklist 5 times longer, corresponding to 2500 entries, has a prediction rate of about 59%. This suggests that the best length for our predictive blacklist is about 500 entries.

C. Training and Testing Windows

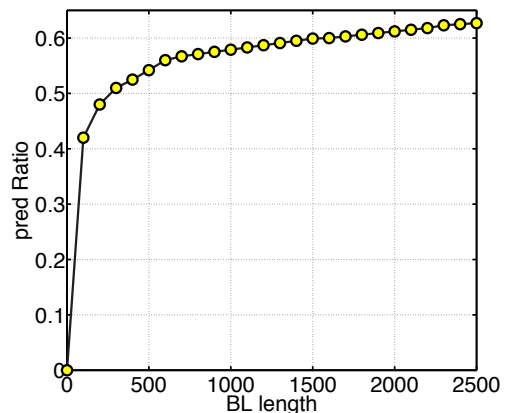
Throughout the paper we used training and testing windows of 5 and 1 days respectively. Fig. 9 shows the performance of our prediction scheme as a function of the length of these windows and justifies these choices.

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. 9(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 5-6-day data.

In Fig. 9(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.



(a) Hit count as a function of the blacklist length



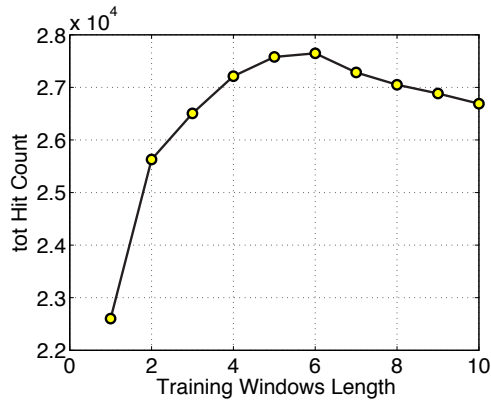
(b) Prediction ratio as a function of the blacklist length

Fig. 8. As the blacklist length increases so does the hit count. However, the largest relative increase occurs when the blacklist length is 500.

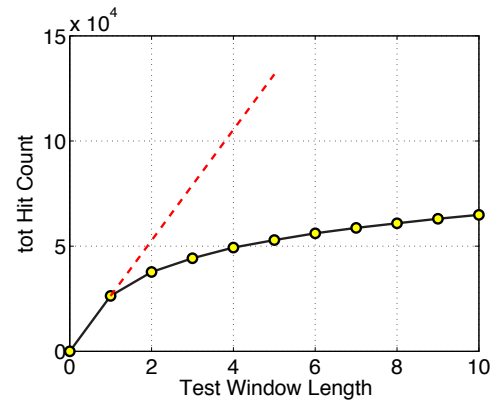
D. Robustness against Pollution of the Training Data

Large-scale repositories that collect firewall and intrusion and detection system logs from different networks (contributors), such as Dshield, are naturally prone to include 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 a contributor's intrusion and detection system (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.

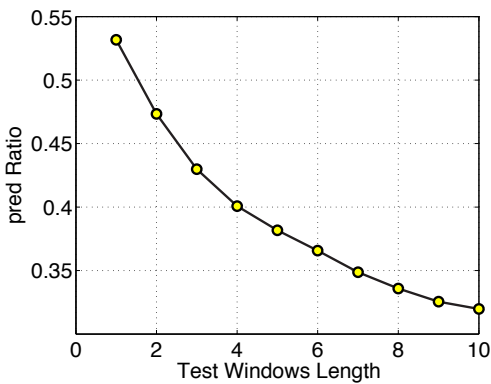
Noisy entries. To quantify how random false positives affect the prediction accuracy of our combined method, we artificially generated fake reports, which are distributed over all contributing networks proportionally to the number of real reports they submitted. We vary the amount of total fake reports generated (noise) from 1% to 15%. Fig. 10(a) shows the results. We observe that the hit count decreases slower than the number of noisy entries, *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



(a) Hit count vs. training window length



(b) Hit count vs. test window length

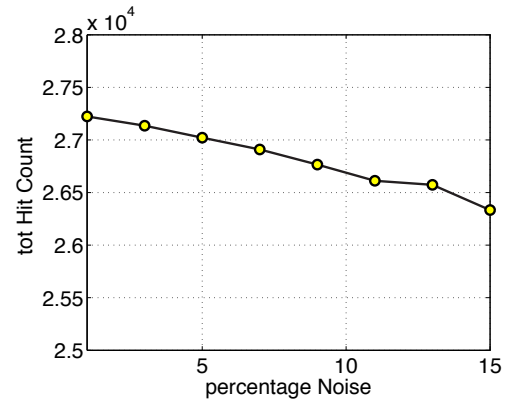


(c) Prediction rate vs. test window length

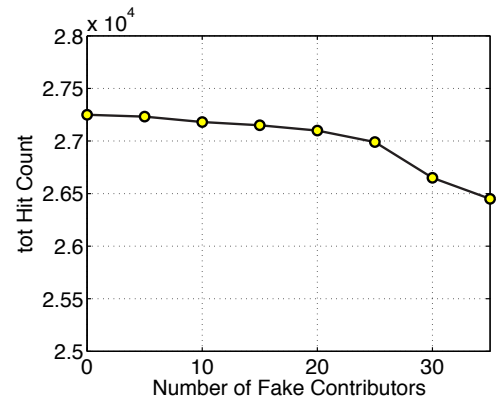
Fig. 9. Tuning the training and testing window of our proposed combined method based on our data. Every point on these plots represents the average total hit count over 7 consecutive days. At the end, we chose a training window of 5 days and a test window of 1 day.

the same source but also a similar time stamp to affect the k NN model presented. 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.

Colluding attackers. The system proposed collects security logs from various trusted sources and use them to provide an estimation on which IPs should be are more likely to attack each contributing network. An attacker may try to bias the prediction mechanism by posing itself as a fake contributor



(a) Noisy entries.



(b) Colluding contributors.

Fig. 10. Robustness of the combined method in the presence of pollution of the training data generate by random noise (a), and by colluding contributors (b). In both scenarios, the total hit count decreases much slower than the amount of pollution injected in the training set.

(victim network) and providing artificially generated data. These fake reports affect the similarity measures, and thus have the potential to affect the prediction for all other contributors in the system. If the fake reports were randomly distributed over all possible attackers, this would affect the neighborhood models to a similar extent they are affected by noisy entries. However, a set a colluding contributors could coordinate to fake the system into believing that a group of attackers are part of the same botnet by submitting the *same* or highly overlapping reports. In this way, a true contributor that happens to have some overlap with the fake report could be induced to believe that the not overlapping part of the fake report is also likely to attack in the future.

To simulate this attack, we introduce an increasing number of fake contributors in the training dataset, B . Each fake contributor reports that it has been attacked by the same set of attackers \mathcal{A}_f . The size of \mathcal{A}_f is chosen equal to the average number of attackers reported by each (true) contributor. As shown in Fig. 10(b), the combined predictor is significantly robust to this type of coordinate attack. This happens for several reasons. This attack affects exclusively the neighborhood models and not the time series prediction. Moreover, to significantly mislead the neighborhood predictors, the fake contributors should be “more similar” to the target victims than all other (true) contributors. In fact, as long as there

are true contributors that share common attackers with the target victims, the neighborhood models will account also for their (true) contribution and this will have the effect to partly counterbalancing the effect of fake reports.

Evading the system. Evading our combined prediction is difficult for an attacker and comes at the cost of limiting the attack impact. Indeed, an attacker must avoid both the time series prediction and the two neighborhood-based methods. To mislead the time series, an attacker can limit traffic towards the target network. In fact, even activities that have low intensity but are persistent over time will be revealed by the time series model. This is beneficial on its own. A different attack strategy to evade the time series prediction would be to attack different networks for a short time. However, this is precisely the behavior that will be captured by the neighborhood-based models, which focus on analyzing and correlating malicious activities reported by different networks.

VII. APPLICATIONS AND FUTURE DIRECTIONS

We envision that BRS can be used as an improved black-listing technique or as side-information in forensics analysis. For example, BRS could be used as an improvement of HPB, which is currently provided as a service by `Dshield.org`. The same functionality could also be implemented in a distributed way among collaborating participants. If used to provide side-information, BRS can recommend a list of addresses that are most likely involved in malicious activity. More resources (*e.g.*, deep packet inspection) could then be selectively allocated to closely monitor these hosts. Alternatively, the network administrators of these hosts could be contacted and warned so that they can take action and fix their systems. BRS could also complement an intrusion detection system by providing context and helping to narrow the error margins in the classification of malicious flows.

There are several ways one can apply and extend our framework. Our recommendation system computes a rating matrix, whose element indicates the probability that an attacker will target a specific victim in the near future. In this paper, this matrix was used to create per-victim customized blacklists, which consist of attack sources associated with the highest ratings for that particular victim. One can use the same rating matrix differently, *e.g.*, to predict which victim networks are most likely to be attacked by a source, or to look at different granularity of attack or victim addresses (*e.g.*, other or variable prefix lengths). Extending the framework to exploit additional fields in the data, *e.g.*, port numbers of logged flows, will increase the dimension of the problem but has also the potential to improve prediction. Furthermore, it would be interesting to look more into the temporal dynamics of malicious activity, *e.g.*, to understand whether victims turn into attackers over time.

VIII. CONCLUSION

In this paper, we studied the problem of predicting future malicious activity given past observations, available through a shared repository of logs from different victims/contributors. We framed the problem as an implicit recommendation system. Within this framework, we proposed *Blacklisting Recommendation System* (or *BRS*) – 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 similarity between victims being attacked by the same sources, preferably at the same time. The second is a co-clustering algorithm that automatically discovers groups of attackers that attack a group of victims at the same time. We found that the application of computationally intensive techniques, such as SVD, that mine global patterns do not significantly improve the prediction accuracy; therefore, we choose not to include them in BRS.

As part of this study, we analyzed a real dataset of 1-month logs from `Dshield`, consisting of 100s of millions network security logs contributed by 100s of different networks. The analysis revealed several attack patterns at the IP layer, which, apart from being informative on their own, guided the design of BRS. We used the `Dshield` dataset to evaluate BRS and show that it brings significant improvement over state-of-the-art attack prediction methods, in terms of both prediction accuracy and robustness against pollution/poisoning of the dataset.

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 insightful 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.

REFERENCES

- [1] Dshield dataset, <http://www.dshield.org/>.
- [2] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, 2005.
- [3] Netflix Prize, <http://www.netflixprize.com/>.
- [4] G. Linden, B. Smith, and J. York, "Amazon recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, Feb. 2003.
- [5] "Google news," <http://news.google.com/>.
- [6] J. Zhang, P. Porras, and J. Ullrich, "Highly predictive blacklisting," in *USENIX Security*, San Jose, CA, USA, Jul. 2008.
- [7] SANS Internet Storm Center, <http://isc.sans.org/top10.html>.
- [8] S. Katti, B. Krishnamurthy, and D. Katabi, "Collaborating against common enemies," in *ACM IMC*, Berkeley, CA, USA, Oct. 2005.
- [9] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos, "Fully automatic cross-associations," in *ACM KDD*, Seattle, WA, USA, Aug. 2004.
- [10] The Spamhaus Project, <http://www.spamhaus.org/>.
- [11] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks," in *IEEE INFOCOM (Mini-Conference)*, San Diego, CA, USA, Mar. 2010.
- [12] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious URLs," in *ACM SIGKDD*, Paris, France, June 2009.
- [13] A. Le, A. Markopoulou, and M. Faloutsos, "PhishDef: URL Names Say It All," in *IEEE Mini INFOCOM (Mini-Conference)*, Shanghai, China, Apr. 2011.
- [14] J. Jung and E. Sit, "An empirical study of spam traffic and the use of DNS black lists," in *ACM IMC*, Taormina, Italy, Oct. 2004.
- [15] A. Ramachandran, N. Feamster, and D. Dagon, "Revealing botnet membership using DNSBL counter-intelligence," in *ACM/USENIX SRUTI*, San Jose, CA, USA Jul. 2006.
- [16] S. Sinha, M. Bailey, and F. Jahanian, "Shades of grey: On the effectiveness of reputation-based blacklists," in *Malware*, Fairfax, Virginia, USA, Oct. 2008.

- [17] T. Karagiannis, D. Papagiannaki, and M. Faloutsos, "Blinc: Multilevel traffic classification in the dark," in *ACM SIGCOMM*, Philadelphia, PA, USA, Aug. 2005.
- [18] A. Ramachandran, N. Feamster, and S. Vempala, "Filtering spam with behavioral blacklisting," in *ACM CCS*, Alexandria, VA, USA, Oct. 2007.
- [19] S. Hao, N. Feamster, A. Gray, N. Syed, and S. Krasser, "Detecting spammers with snare: Spatio-temporal network-level automated reputation engine," in *USENIX Security*, Montreal, Canada, Aug. 2009.
- [20] P. B. Z. Chen, C. Ji, "Spatial-temporal characteristics of internet malicious sources," in *IEEE INFOCOM (Mini-Conference)*, Phoenix, AZ, USA, Apr. 2008.
- [21] P. Barford, R. Nowak, R. Willett, and V. Yegneswaran, "Toward a model for sources of internet background radiation," in *PAM*, Adelaide, Australia, Mar. 2006.
- [22] Y. Koren, "Collaborative filtering with temporal dynamics," in *ACM KDD*, Paris, France, June 2009.
- [23] E. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.
- [24] E. Candes and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2009.
- [25] F. Soldo, A. Le, and A. Markopoulou, "Predictive Blacklisting as an Implicit Recommendation System," in *IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010.
- [26] F. Soldo, A. Markopoulou, and K. Argyraki, "Optimal filtering of source address prefixes: Models and algorithms," in *INFOCOM '09*, Rio de Janeiro, Brazil, Apr. 2009.
- [27] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *ACM IMC*, Rio de Janeiro, Brazil, Oct. 2006.
- [28] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," in *USENIX SRUTI*, Cambridge, MA, USA, Jul. 2005.
- [29] E. Deprettere, *SVD and signal processing: algorithms, applications and architectures*, North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 1989.
- [30] J. Elder, "Fusing the results of diverse algorithms," in *Proceeding of the 3rd International Conference on Multi-strategy Learning*, 1996.
- [31] R. Nisbet, J. Elder, and G. Miner, *Handbook of statistical learning and data mining applications*. Elsevier, 2009.
- [32] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and internet traffic matrices," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 267–278, 2009.



Fabio Soldo is currently a Ph.D. candidate in the Networked Systems Program at UC Irvine. He received his B.S. degree in Mathematics from Politecnico di Torino, Italy, in 2004, and M.S. degree in Mathematical Engineering from Politecnico di Torino and Politecnico di Milano, Italy, in 2006. He had internships with Telefonica Research, Docomo Euro-Labs and Google. His research interests are in the areas of design and analysis of network algorithms and network protocols, and defense mechanisms against malicious traffic.



Anh Le is currently a Ph.D. candidate in the Computer Science Program at UC Irvine. He received a B.S. degree in Computer Science - Mathematics from the University of Manitoba, Canada in 2006 and a M.Math. degree in Computer Science from the University of Waterloo, Canada in 2008. His research interests are in the area of network security, including network coding security, collaborative firewalls, phishing detection, and network intrusion detection systems.



Athina Markopoulou (SM '98, M'02) is an assistant professor in the EECS Dept. at the University of California, Irvine. She received a Diploma degree in Electrical and Computer Engineering from the National Technical University of Athens, Greece, in 1996, and M.S. and Ph.D. degrees, both in Electrical Engineering, from Stanford University in 1998 and 2003, respectively. She has been a postdoctoral fellow at Sprint Labs (2003) and at Stanford University (2004–2005), and a member of the technical staff at Arastra Inc. (2005). Her research interests include network coding, network measurements and security, media streaming and online social networks. She received the NSF CAREER award in 2008.