Towards the Effective Temporal Association Mining of Spam Blacklists*

Andrew G. West
University of Pennsylvania
Philadelphia, PA, USA
westand@cis.upenn.edu

Insup Lee
University of Pennsylvania
Philadelphia, PA, USA
lee@cis.upenn.edu

ABSTRACT

IP blacklists are a well-regarded anti-spam mechanism that capture global spamming patterns. These properties make such lists a practical ground-truth by which to study email spam behaviors. Observing one blacklist for nearly a year-and-a-half, we collected data on roughly *half a billion* listing events. In this paper, that data serves two purposes.

First, we conduct a measurement study on the dynamics of blacklists and email spam at-large. The magnitude/duration of the data enables scrutiny of long-term trends, at scale. Further, these statistics help parameterize our second task: the mining of blacklist history for temporal association rules. That is, we search for IP addresses with correlated histories. Strong correlations would suggest group members are not independent entities and likely share botnet membership.

Unfortunately, we find that statistically significant groupings are rare. This result is reinforced when rules are evaluated in terms of their ability to: (1) identify shared botnet members, using ground-truth from botnet infiltrations and sinkholes, and (2) predict future blacklisting events. In both cases, performance improvements over a control classifier are nominal. This outcome forces us to re-examine the appropriateness of blacklist data for this task, and suggest refinements to our mining model that may allow it to better capture the dynamics by which botnets operate.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: *Security and protection*; K.6.5 [**Management of Computing and Info. Systems**]: Security and Protection; H.2.8 [**Database Management**]: Database applications—*Data mining*

Keywords

Email spam, IP blacklists, measurement study, temporal data mining, association rule learning, botnet detection.

*This research is supported in part by ONR MURI N00014-07-1-0907. POC: Insup Lee, *lee@cis.upenn.edu*

1. INTRODUCTION

Email spam is a topic that requires little introduction. In 2010, it was estimated that spam compromised nearly 90% of all email sent [37], consuming significant resources. IP blacklists have become a standard tool in mitigating such traffic, with research reporting high detection rates [24, 39].

Efficiency is likely one reason IP blacklists are commonly employed. While spam detection using natural-language processing (NLP) is known to be effective, it also requires considerable resources. Therefore, there has been much recent research on pre-filtering mails using only "network-level properties" [31, 32]. In particular, "temporal" [20, 39] and "spatial" [20, 30, 33, 38, 39] techniques have been proposed. Such mechanisms can be used to *proactively* blacklist malicious addresses (before they can send spam), by extrapolating from current evidence or leveraging historical patterns.

By the end of 2010, sources indicate that 88% of email spam originates from *botnets*, a significant increase over earlier statistics [37, 41]. This rising prevalence may prove problematic for spatio-temporal detection schemes. For example, botnets' massively distributed nature means that infected hosts are not confined to organizational/geographic boundaries, or contiguous regions of address space [37, 43]; assumptions on which many spatial methodologies depend.

Thus, to a large extent, the spam detection task has become equivalent to the botnet identification one. At the network-level, the ability to recognize botnet membership is limited. Given botnets' use of P2P communication, recent proposals [18, 27] suggest processing expansive network flows/graphs. While effective over simulated traces, scope and privacy issues hamper practical application.

Beyond the network-level, more significant progress has been made by investigating spam email content. Much research identifies spam campaigns (and therefore, botnets) using the similarity of: (1) spam email bodies, (2) URLs in spam email bodies, and/or (3) content at those URL destinations [6, 9, 22, 25, 28, 29, 41, 43]. Once generated, such signatures can block spam from *ongoing* campaigns, and the set of source IPs could be useful against *future* ones.

Such content-analysis, however, is computationally expensive and may have privacy implications. Therefore, this paper proposes an alternative model to arrive at botnet groupings leveraging only IP blacklists and their temporal listing patterns. This builds on a simple intuition: IP addresses which have a history of being blacklisted "together" are likely participants in the same spam campaigns/botnet. In practice, factors such as infection lifetimes and DHCP address space complicate trend development.

To gain insight into these dynamics and to investigate the feasibility of spam mitigation using temporal correlations, we subscribe to a popular blacklist provider [1]. Nearly *one-and-a-half years* of blacklist data has been collected, encompassing roughly *500 million listings*. This extensive dataset is first used to conduct a measurement study on long-term blacklist and botnet behavior. Then, techniques from data mining and association rule learning are applied to search for temporally correlated IP sets.

Disappointingly, we find that statistically sound groupings (*i.e.,* association rules) are rare (*i.e.,* most have low *interest metrics*). Reinforcing this, we evaluate these rules/metrics in two respects. First, we measure the degree of correlation between known members of the Cutwail and Kraken botnets. Second, we gauge the ability of rules to predict future blacklist events. In both tasks, temporal learning allows our methods to outperform pure chance. However, the rarity of meaningful rules means our technique is only effective for trivial portions of the problem space.

This outcome demands a discussion of why our technique was not more successful. We consider that: (1) blacklists may be too narrow a data source to capture the complexities of spam campaigns, and (2) that our straightforward mining technique may be ill-equipped to model the extremely dynamic nature of botnet operations.

Given that our technique does outperform a control classifier, it could potentially be used to improve more robust predictors. However, we believe the more valuable contributions of this work are: (1) an adaptation of temporal data mining techniques for use over spam blacklist data, creating a foundational model on which future refinements can build, and (2) a discussion and quantification of the temporal dynamism by which blacklists and botnets operate.

This paper proceeds as follows: first, related work is reviewed (Sec. 2). Next, our dataset and blacklist operation are discussed, before conducting a measurement study (Sec. 3). Thereafter, a model for temporally mining blacklist history is presented (Sec. 4). This model is then empirically evaluated and performance shortcomings are discussed (Sec. 5). Finally, concluding remarks are made (Sec. 6).

2. RELATED WORK

Here, related work is surveyed – both as it pertains to blacklists, spam, and botnet operation (Sec. 2.1), as well as association rule mining (Sec. 2.2).

2.1 Blacklists & Botnet Dynamics

IP blacklists are this writing's ground-truth for investigating spam behavior. Previous writings have addressed such lists, examining their operation and comparative effectiveness [12, 23, 31, 39]. While blacklists are rarely criticized for being inaccurate (*i.e.,* false-positives), latency and lack of coverage have been identified as issues [33].

Much prior work has focused on extrapolating from ground truth about spamming IP addresses – blacklists included – to discover additional malicious IPs. Of particular interest are techniques leveraging the spatial distribution of spamming IP addresses [30, 33, 38, 39]. At various granularity (*e.g.,* AS, subnet, rDNS host), these mechanisms calculate reputations for groups and apply them to mails sent from group members. However, most such groupings are contiguous in address space, a dangerous assumption given the rising prevalence of botnets [37, 43]. In Sec. 4, we propose a novel means to discover non-contiguous spamming groups (*i.e.,* botnets). Once discovered, such groups can easily be interfaced/integrated with existing reputation frameworks [39].

Of course, we are not the first to address botnet-node detection. One line of research has focused on parsing P2P communication patterns from expansive (*i.e.,* non-local) network graphs [18, 27]. Our proposed technique (Sec. 4) captures global trends by the very nature of blacklist construction (Sec. 3.1), without the need for broad views.

An alternate means of botnet discovery has been using similarity algorithms [8] to correlate spam email bodies or the URLs they contain (and therefore, their senders) [6, 9, 22, 25, 28, 29, 41, 43]. Once found, such patterns can be used to generate anti-spam signatures or form the basis for analyzing botnet operations. Our technique (Sec. 4) is similar, but correlates using only blacklist timestamps. Further, since we are not reliant on content analysis, our proposal may have greater scalability and less privacy implications.

No matter the technique, an understanding of botnet operations is critical to the detection task. Recent botnet infiltrations [24, 35, 36] and measurement studies [6, 9, 22, 28, 37, 43] are invaluable in this regard. We rely on these to help us parameterize our learning technique in Sec. 4. Our own measurement study (Sec. 3.3) rooted in *blacklist data* complements these findings.

2.2 Association Mining

To detect temporal correlations between IP addresses in our dataset, we rely on data mining techniques from the domain of *association rule learning*. Seminal algorithms such as Apriori [4] and FP-Growth [19] may be familiar to readers, but are reliant on discrete and unordered transactional data. *Sequential mining* [5, 34] does enforce ordering, but only uni-directionally (*i.e.,* in a forward-looking fashion).

Our blacklist data defies both assumptions. First, it is continuously distributed and therefore not well "binned" into transactions. Second, we wish to capture bi-directional associations (*i.e.,* the blacklisting of one address must not strictly precede/follow another). Therefore, our work falls into the broader domain of *temporal association learning*. In this field, data-specific solutions are the norm, with much literature simply describing the challenges [7, 10, 21, 42].

In Sec. 4.1 we describe the basic concepts of association mining as we introduce our learning strategy. Although guided by existing research, our approach of Sec. 4 is quite ad-hoc and adapted to the peculiarities of our dataset. Modifying these algorithms for online use will be non-trivial, but likely build on the suggestions of [17, 21].

3. BLACKLISTS

Next, we discuss general blacklist terminology and operation (Sec. 3.1), before focusing on the specific blacklist(s) used for data collection (Sec. 3.2). Then, this data is the basis for conducting a measurement study (Sec. 3.3).

3.1 Blacklist Operation

IP blacklists are lists of IP addresses that are believed to be sources of email spam. Such lists are installed on (or queried by) email servers, which usually refuse receipt of messages from listed senders. Blacklists are popular: one provider [1] claims to protect 1.5 billion inboxes.

Blacklists evolve over time, as providers add and remove IP addresses. An IP address appearing in the most recent

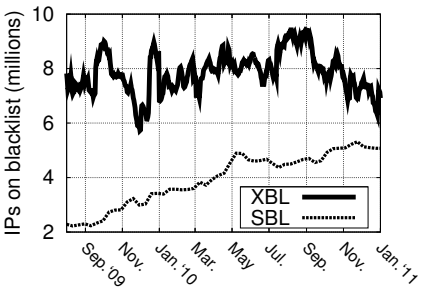

Figure 1: Blacklist size

PROPERTY	XBL-#
listings collected	467 mil.
unique IP addresses	126 mil.
average snapshot size	7.8 mil.
daily churn/turnover	0.9 mil.
unique AS homes	23,296
AS for 25% of listings	11
AS for 50% of listings	40
AS for 90% of listings	378

Table 1: XBL properties over 17-month data collection

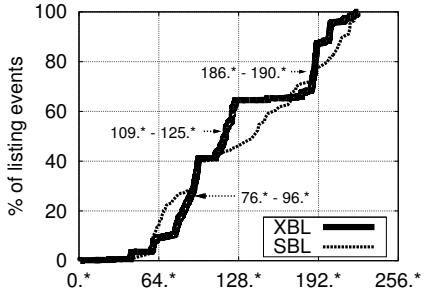

Figure 2: Address space

version of a blacklist is said to be *actively listed*. When an IP is added to the list it is a *listing* and removals are called *delistings*. When an address becomes active (again) after de-listing it is termed a *relisting*. Relistings are of particular interest given our use of pattern mining.

Blacklist providers must also decide which IP addresses to list, and the specifics of these decision policies are carefully guarded to prevent evasion. Anecdotally, *major* mail providers (with broad Internet views) are believed to be significant sources of evidence, along with honeypot data.

Similarly, policies dictate how/when delisting should occur. Many blacklists offer channels by which false-positives can be submitted and corrected. Beyond this, common delisting strategies include: (1) manual verification of innocence, (2) static time-to-live (TTL), and (3) dynamic TTL (based on behavior during the blacklist period). No doubt, the legal challenges brought against blacklist providers [2] force these policies to be somewhat conservative.

3.2 Data Collection

To collect blacklist data, we subscribe to the Spamhaus service [1], which provides three blacklists of interest:

- Exploits Block List (XBL): Listing of IPs caught sending spam as the result of hijacking, trojans, and other malicious infection (*i.e.,* botnet nodes).
- Spamhaus Block List (SBL): Manually-maintained listing of IPs of spamming organizations. Typically static IP blocks controlled/owned by spam gangs.
- Policy Block List (PBL): Preventative listings of IPs that should not be sending *any* mail "on principle."

Because the XBL: (1) captures botnet behavior, (2) is large, and (3) is quite dynamic (as made explicit in Sec. 3.3), it is our primary focus[1] (see Tab. 1 for properties of this list). Nonetheless, the other two lists are of casual interest. The SBL aids in quantifying broad spamming patterns. The PBL (not disjoint from the other two lists) is used to reason about the prevalence of DHCP botnet IPs [40].

Data was collected between 2009-08-01 and 2011-01-01 (17 months, or 518 days), with blacklist updates pulled at 30-minute intervals[2]. The `diff` was calculated between consecutive versions to determine new listings and delistings. These events were written to a blacklist history, H:

$$H = \{L_1, L_2, \ldots\},\ L = (\mathsf{IP},\ \mathsf{ts_{in}},\ \mathsf{ts_{out}},\ \mathsf{AS})$$

which is composed of individual listings. Listings, L, are a 4-tuple recording: the (1) IP blacklisted, (2) time of listing, (3) time of delisting (initially, $\mathsf{ts_{out}} = \varnothing$), and (4) the AS(es) which home the IP address (per [3]).

3.3 Blacklist Analysis

Having collected blacklist history, a measurement study is now conducted. While quantifying spam behaviors is interesting in its own right, our findings also help parameterize our temporal mining efforts (Sec. 4). While none of the measures quantified here are novel (see [31, 39]), we emphasize that they: (1) are measured over blacklist data, (2) capture global trends, (3) are of a broad duration, and (4) reflect recent trends (critical given the evolving spam landscape).

Blacklist Magnitude: For analysis, it is desirable that the size of the blacklist is roughly proportional to the total volume of email spam being sent. Fortunately, Fig. 1 shows that the XBL size correlates well with such graphs in literature [37]. Similarly, noteworthy spam events can be identified in Fig. 1. For example, Dec. 2009 shows a spam spike typical of the holiday season [15]. Moreover, the downward trend beginning in Sep. 2010 can be attributed to the closure of a major spam affiliate at that time [15, 37].

Focusing on more narrow time windows produces graphs that are very jagged in nature. This is intuitive: spam *campaigns*[3] are likely to produce multiple listings in short duration (the key intuition behind our temporal mining efforts). Similarly, the XBL's static delisting policy (see below) dictates that campaign IPs will be delisted together.

IP Address Spectrum: Fig. 2 visualizes listings as a function of IP address space. Clearly, certain regions are responsible for a majority of XBL blacklist activity. A similar observation was made in [31], although the problematic regions have shifted slightly since that writing. It is precisely these patterns that spatial methodologies [30, 33, 38, 39] attempt to harness, although [31] found that these same regions also account for a significant portion of legitimate mail.

Accordingly, geo-locating IP addresses shows that several geographical regions are responsible for many blacklisting events, namely Asia and Eastern Europe. Interesting to note is the `186.* - 190.*` space, mapping primarily to Brazil. Research indicates that Brazil was a targeted infection region for the Rustock botnet [37], prior to its 2011 take-down.

[1]For the remainder of this work, the XBL is the sole focus of discussion and processing, unless otherwise made explicit.
[2]The actual blacklist file stores listing times at second accuracy.

[3]Campaigns are coordinated spam-sending efforts promoting the same product or service. By using aggressive sending strategies, campaigns intend to maximize the utility of domains/content/*etc.* against the latency of blacklisting and signature-based detection.

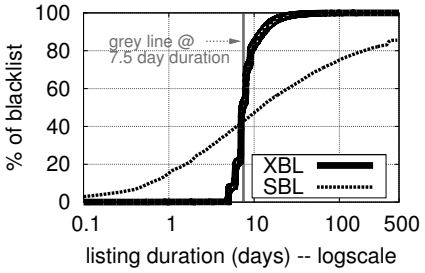

Figure 3: Listing duration

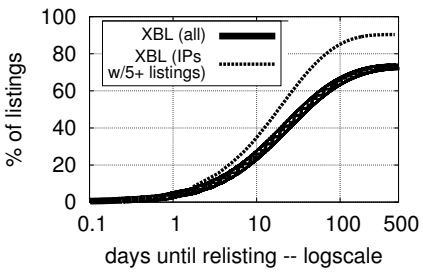

Figure 4: Relisting intervals

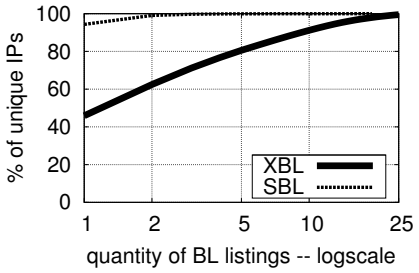

Figure 5: Listing quantity

Botnet operators are aware of these spatial patterns and the ease with which they can be identified. Thus, at the campaign-level, botnets often leverage their distributed nature to span address space. Our temporal mining (Sec. 4) attempts to discover such non-contiguous patterns.

DHCP Addresses: To determine what percentage of XBL listings are homed in dynamic IP space, the PBL is leveraged. The PBL is really two lists: (1) a dynamic space list maintained by Spamhaus (acquired from Dynablock [40]), and (2) IP space, identified by local network administrators, that should not be sending *any* mail. While the latter contains some DHCP addresses, Spamhaus also encourages the addition of mail-server prohibited *static* space (*e.g.,* computer labs, corporate desktops, *etc.*). While distinction is made between the two top-level lists, static vs. dynamic annotations are not made in the "local admin" set.

Of the ≈126 million unique IP addresses appearing on the XBL, some 119 million (90%) also eventually appear on the PBL[4]. Concentrating just on the PBL portion which is known to be dynamic that figure falls to 79.0%. Regardless, this is significantly higher than the proportion of PBL IPs in all address space, which we estimate to be 18.4% (after bogon and special-purpose IP space is discarded).

This result has a significant affect on our temporal mining methods (Sec. 4). For an association rule to be useful, entities need to have persistent identifiers throughout the learning window. A high level of dynamism therefore makes narrow training windows desirable, the trade-offs and practical sizing of which is discussed in Sec. 4.2.

Listing Duration & Churn: A contradictory factor in sizing the training window is that it must be sufficiently long: only IPs with multiple (re)-listings in a window will have a pattern history sufficient for rule generation. Crucially, relisting can only occur as fast as delisting takes place.

As Fig. 3 shows, most XBL listings have a duration near 7.5 days. This consistency speaks to the "static TTL" policy the XBL employs (the smaller and more static SBL uses manual verification). The near-uniform listing interval also means that delisting times ($\mathsf{ts}_{\mathsf{out}}$) hold little value, since they are implicit based on arrival times. Thus, when temporally mining, only arrival timestamps ($\mathsf{ts}_{\mathsf{in}}$) are correlated.

Given that XBL delisting occurs frequently, it is intuitive that the list can be characterized as highly volatile and dynamic – and therefore capable of producing interesting temporal patterns. The XBL churns or "turns-over" 11.4% of its 7.8 million listings each day (see Tab. 1). In comparison, the SBL churns just 0.46% of its volume daily. This static nature is the primary reason the SBL, although an effective blacklist, is downplayed in this work.

Relisting & Turnover: Given that an XBL IP is delisted after one week's time, we are interested in how quick/often spammers re-utilize that address, resulting in a subsequent relisting event. While naïve reuse maximizes address utility, it is also characteristic of spamming behavior and can be easily captured by simple reputation functions [39].

Fig. 4 indicates that at median, a relisting will happen ≈5 weeks after delisting (*not* including the week on the blacklist) with the first quartile reappearing in under 10 days. Notice that in 25% of cases a subsequent listing is not seen in our study period. Since only IPs with multiple listings are relevant for temporal mining, Fig. 4 also plots relisting intervals for those IPs with 5+ listings, showing a more favorable median time-to-relisting of around 18 days.

The prevalence of IP addresses with significant relisting histories, as plotted in Fig. 5, is paramount. While 45% of unique IPs have only a single listing, 20% have 5+ listings. These upper 20% account for 66% of all blacklisting events. This is encouraging: non-trivial histories exist for a considerable portion of the problem space. Moreover, an IP's listing times tend to be temporally clustered, not uniformly distributed over the data period.

4. TEMPORAL MINING

The mining of blacklist histories for temporal association rules, as described in this section, is a novel contribution of this work. Although established mechanisms are the basis for our mining model[5], their application to blacklist data requires both special considerations and careful variable selection. Discussion begins with a mathematical description of our model (Sec. 4.1). Then, results from our blacklist measurement study are used to parameterize (Sec. 4.2) an implementation (Sec. 4.3) of the technique.

4.1 Association Mining Model

The goal of the mining process is to produce association rules which capture the temporal patterns of IP addresses in a blacklist history. For example, one such rule might be: $\mathsf{IP}_x \Rightarrow \mathsf{IP}_y$. That is, IP_y typically appears on the blacklist around the time IP_x does. Such a rule can then be applied in an online fashion. For instance, if IP_x has just been blacklisted, one could proactively blacklist IP_y (Sec. 5.3). Alternatively, one could conclude that IP_x and IP_y are members of the same botnet (Sec. 5.2).

[4]Given that the PBL is primarily a monotonically growing list, only the PBL snapshot at our study's conclusion is considered.

[5]Even so, we present these concepts somewhat pedantically, both for completeness and for the benefit of an anti-spam audience.

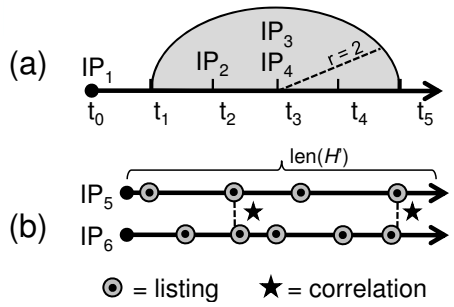

Figure 6: (a) Timeline structure with correlation radius; (b) IP pair with two correlation events.

The input to our mining model is a blacklist history, $H' = \{L'_1, L'_2, \ldots L'_n\}$ composed of listings, $L' = (\mathsf{IP},\ \mathsf{ts_{in}})$, a subset of the data described in Sec. 3.2. The time duration which H' spans is termed the *learning window*, having length $\mathsf{len}(H') = \forall L'_n \in H', [\mathsf{max}(L'_n.\mathsf{ts_{in}}) - \mathsf{min}(L'_n.\mathsf{ts_{in}})]$.

It is helpful to visualize H' as a timeline of listing events, as done in Fig. 6a. For example, IP_1 is listed at time t_0, IP_2 at time t_2, and so on. Using this, it is easy to see *correlation events*, the atomic notion on which rules are built. Two listings, $L'_i, L'_j \in H'$, are said to be correlated, $\mathsf{corr}(L'_i, L'_j) = 1$, if if they occur within, r, a *correlation time radius* of each other, *i.e.,* $(L'_j.\mathsf{ts_{in}} - r \leq L'_i.\mathsf{ts_{in}} \leq L'_j.\mathsf{ts_{in}} + r)$.

Note that correlation relationship is symmetric but non-associative. For example, choosing $r = 2$ as shown in Fig. 6a, we see that $\mathsf{corr}(\mathsf{IP}_3, \mathsf{IP}_2) = 1$ and $\mathsf{corr}(\mathsf{IP}_2, \mathsf{IP}_3) = 1$ (symmetry). While $\mathsf{corr}(\mathsf{IP}_1, \mathsf{IP}_2) = \mathsf{corr}(\mathsf{IP}_2, \mathsf{IP}_3) = 1$, we observe that, $\mathsf{corr}(\mathsf{IP}_1, \mathsf{IP}_3) = 0$ (*i.e.,* association does not hold).

Note that the radius is drawn only after the input parameters are known. This is a practical requirement given that listing timestamps are continuously distributed. In contrast, most mining algorithms operate over transactional data [10]. Coercing our data into such "bins" is undesirable: two listings occurring a single time unit apart could be binned separately, making them "non-correlated".

Ultimately, we are not interested in correlations between listings, but correlations between IP addresses. Thus, we construct a function, $\mathsf{list}(\mathsf{IP_x}) = \{L'_1, L'_2 \ldots L'_n\}$, producing a set of all listings in which $\mathsf{IP_x}$ appears. This is useful in counting the number of correlations two IP addresses have over entire training window:

$$\mathsf{corr\_pair}(\mathsf{IP_x}, \mathsf{IP_y}) = \sum_{\substack{l_x \in \mathsf{list}(\mathsf{IP_x}),\\ l_y \in \mathsf{list}(\mathsf{IP_y})}} \mathsf{corr}(l_x, l_y)$$

For example, looking at Fig. 6b we see that, $|\mathsf{list}(\mathsf{IP}_5)| = 4$, $|\mathsf{list}(\mathsf{IP}_6)| = 5$, and $\mathsf{corr\_pair}(\mathsf{IP}_5, \mathsf{IP}_6) = 2$.

Such absolute counts are more meaningful if normalized by the size of the search-space/training-window. This is a well-known data mining concept known as *support* [16]:

$$\mathsf{supp}(\mathsf{IP_x}) = \frac{|\mathsf{list}(\mathsf{IP_x})|}{\mathsf{len}(H')/2r}$$

$$\mathsf{supp\_pair}(\mathsf{IP_x}, \mathsf{IP_y}) = \frac{\mathsf{corr\_pair}(\mathsf{IP_x}, \mathsf{IP_y})}{\mathsf{len}(H')/2r}$$

Here, the correlation diameter $(2r)$ is treated as the "bin size", and the denominator counts the number of bins in the training window. Then, we compute the fraction of bins in which an event of interest (listing/correlation) occurs[6].

Any pair[7] of IP addresses can be written as an association rule of the form, $\mathsf{IP_x} \Rightarrow \mathsf{IP_y}$ (*i.e.,* implication). However, a rule is only useful if it is *interesting*, and *interest metrics* are used to quantify this notion. The interest metric used in this work is *lift* [16]:

$$\mathsf{lift}(\mathsf{IP_x} \Rightarrow \mathsf{IP_y}) = \mathsf{lift}(\mathsf{IP_x}, \mathsf{IP_y}) = \frac{\mathsf{supp\_pair}(\mathsf{IP_x}, \mathsf{IP_y})}{\mathsf{supp}(\mathsf{IP_x}) \times \mathsf{supp}(\mathsf{IP_y})}$$

Lift is a probabilistic ratio that relates the *actual support* to the *expected support* that would occur as the result of random chance. High lift is indicative of an association that is unlikely to occur randomly. Lift was chosen to quantify rules because of its, (1) probabilistic interpretation and (2) symmetric nature, which can reduce rule search-space by half. Alternative interest metrics are described in [16].

Having this, one can now compute a lift-measure for all pairs of IP addresses in H':

$$\mathsf{lift}(H') = \begin{bmatrix} - & \mathsf{lift}(\mathsf{IP}_1, \mathsf{IP}_2) & \cdots & \mathsf{lift}(\mathsf{IP}_1, \mathsf{IP_n}) \\ \mathsf{lift}(\mathsf{IP}_2, \mathsf{IP}_1) & - & \cdots & \mathsf{lift}(\mathsf{IP}_2, \mathsf{IP_n}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{lift}(\mathsf{IP_n}, \mathsf{IP}_1) & \mathsf{lift}(\mathsf{IP_n}, \mathsf{IP}_2) & \cdots & - \end{bmatrix}$$

Needless to say, the $\mathcal{O}(n^2)$ complexity of this operation is problematic. Thus, the search space is often constrained using the notion of *minimum support*, M, whereby no rules producing $\mathsf{supp\_pair}() < M$ are considered. Therefore, addresses having $|\mathsf{list}(\mathsf{IP_x})| < M$ can be discarded from the matrix, using the *downward closure property*. In Sec. 4.3 we describe programmatic steps to improve scalability.

Having produced the $\mathsf{lift}(H')$ matrix, each row is sorted by the lift-metric (in descending order) to produce an ordered "best-pairs" list for all IP addresses (associated lift values are also retained, but not displayed):

$$\mathsf{best\_pairs}(\mathsf{IP_x}) = [\mathsf{IP_a}, \mathsf{IP_b} \ldots \mathsf{IP_z}],\ \text{where}$$
$$\mathsf{lift}(\mathsf{IP_x}, \mathsf{IP_a}) \geq \mathsf{lift}(\mathsf{IP_x}, \mathsf{IP_b}) \geq \ldots \mathsf{lift}(\mathsf{IP_x}, \mathsf{IP_z}) > 0$$

This list, generated over prior observed events, can then be applied online. For reasons described in the next section, we believe this model can capture botnet signatures. Therefore, we are hopeful that top-ranking pairs can capture shared botnet membership (Sec. 5.2), and later we use best-pairs as the basis for predicting *future* listing events (Sec. 5.3).

4.2 Variable Selection

We are hopeful our temporal mining technique is capable of capturing malicious behaviors. However, the dynamic nature of spamming operations, botnet infections, and address space complicate this task. To account for these, the free variables of the model must be properly selected.

The breadth/complexity of experiments (Sec. 4.3) make empirical trial-and-error costly. Thus, parameters are ar-

[6]Bin-size $(2r)$ is constant per our correlation radius. Thus, when computing $\mathsf{corr}()$, in effect, we simply slide bins until an IP address of interest lies at some bin's midpoint. This does not affect the number of bins, nor the *random* probability of two events appearing in the same bin. Thus, our support calculation captures the same notion it would over inherently transactional data.

[7]We present these notions pair-wise for simplicity. They are also easily generalized to sets of IP addresses (*i.e.,* $\{\mathsf{IP_x}, \mathsf{IP_y}\} \Rightarrow \{\mathsf{IP_z}\}$).

rived at logically[8]. Assisting in this regard are the blacklist measurement study of Sec. 3.3 and empirical botnet literature. We now address the free variables individually:

Correlation Radius: Given that spam campaigns are initiated *en masse*, one would expect to see associated participants blacklisted in close temporal proximity. Thus, the correlation radius (r) needs set to capture the arrivals of a typical campaign, while minimizing other blacklist traffic (*e.g.,* intersections with *other* campaigns). Fig. 1 indicates that localized listings spikes are not uncommon.

In their post-mortem analysis of the Cutwail botnet (see also Sec. 5.2) the authors of [36] show that blacklisting rates peak 1–2 hours after a node comes "online." An earlier study of the Storm botnet [24] reported that nodes arrived on a blacklist within 1.5 hours, at median, after being instructed to begin spamming. Additional literature supports timings in the "several hour" range. For example, [43] states that 70% of spam emails are sent by campaigns lasting less than 8 hours. The same study indicates that large botnets tend to operate the least aggressive campaigns.

Given Cutwail's large size [37] and the recency of [36], we believe $r = 2$ hours is a reasonable radius selection (this will makes the capture window 4 hours in breadth). In making this choice, we choose not to accommodate the lengthy tails of "time to blacklist" distributions – instead prioritizing a narrow radius to minimize noise. We note that there exists conflicting evidence that some campaigns are very lengthy [28] (*e.g.,* weeks). Our model cannot reasonably accommodate such a radius, and we acknowledge that nodes participating in such campaigns may correlate poorly.

Training Window: Determining an appropriate training window size is challenging. First, we discuss two aspects supporting a narrow training window: (1) infection lifetime and (2) IP dynamics (DHCP). Then, we consider two contradictory factors: (3) the need for sufficient pattern development and (4) bot-to-campaign allocation.

For association rules to develop, it is important that a node stays under botnet control throughout the training window. Anecdotal evidence places infection lifetimes at "years" [14]. Even if inflated, infections lasting "months" are sufficient given the other constraints of this section.

In particular, understanding dynamic address space is critical, as blacklists identify only IPs, not unique machines. DHCP has been researched as it pertains to spamming [40] and the measurement of botnet size [35, 43]. The former study reports that 42% of spam mail is received from dynamic hosts (far lower than our estimations of Sec. 3.3; see also Sec. 5.4). More important than *if* an IP is dynamic is *how dynamic* it is: dynamic IPs can be treated as persistent identifiers if they do not churn inside the training window.

While dial-up/DSL connections tend to churn hourly or daily, [40] found that 70% of one cable-Internet company's subscribers retained their addresses for over a month (the extent of their study). Anecdotal evidence shows cable Internet IP retention-times can be as long as years [11].

Fortunately, highly dynamic IPs should implicitly exclude themselves from association rules due to minimum support (M) constraints. For example, given that it takes at minimum 7.5 days (per Fig. 3) for an IP to relist, IPs churning weekly (or faster) cannot conform to $M \geq 2$. Only IPs with expected renewal in the range $[7.5M, \mathsf{len}(H')]$ are likely to be noisy for our model. Fortunately, [40] suggests that volatility might be bi-modally distributed based on service type, minimizing the quantity of IPs in this "dangerous range."

[8]In Sec. 5 we gauge the accuracy of our variable selections over ground-truth. However, reverse-engineering optimal values from this data is inappropriate because: (1) it could favorably bias results and (2) parameters may not speak to global behavior.

PROPERTY	LISTS	PCT	UNIQ-IP	PCT
total listings	69,188,643	100%	41,455,265	100%
min. supp. ≥ 2	43,809,390	63%	16,076,012	39%
min. supp. ≥ 3	26,395,188	38%	7,368,911	18%
min. supp. ≥ 4	13,559,703	20%	3,090,416	7%
min. supp. ≥ 5	5,089,743	7%	972,926	2.3%
min. supp. ≥ 6	1,177,078	1.7%	190,393	0.5%

Table 2: Training window support levels, interval Sep. 1, 2010 (inclusive) – Dec. 1, 2010 (exclusive)

With the previous two factors supporting a short training window, we now discuss aspects of botnet operation that suggest lengthy windows might be more appropriate.

First, the training window must be long enough to allow patterns/support to develop. Fig. 3 shows that relistings require at least 7.5 days, and when combined with Fig. 4, the median time-to-relisting is just over one month. However, since highly dynamic IPs are unlikely to accumulate minimum support, the median turnover of "relevant" IPs is somewhat quicker (also, Fig. 4), at about three weeks.

While this is encouraging at the single IP level, correlations are non-singular. That is, even if IPs are in the same *botnet*, they must participate in the same *campaigns* in order to associate. Work shows that 80% of botnets use less than half of their available bots in a campaign [43], while [36] confirms one bot served multiple "clients" simultaneously. Of benefit to our model, one source [22] reports that some botnets partition campaigns statically across available nodes.

Synthesizing these trade-offs into a single parameter is non-trivial. However, combining the above evidence and the computational constraints of Sec. 4.3, we believe $\mathsf{len}(H') = 3$ months is a reasonable starting point for analysis.

Minimum Support: Generally, raising the minimum support, M, of the model increases scalability at the expense of excluding IPs from analysis. Such exclusion reduces the portion of problem space to which rule application is possible (*i.e.,* the evaluation tasks of Secs. 5.2, 5.3). That being said, there is no use in generating weak rules. Tab. 2 examines this trade-off at varying support levels over a 3-month training window. Because it is feasible to compute (see Sec. 4.3), the conservative[9] choice of $M = 3$ is made.

4.3 Implementation & Scalability

We next implement the lift-matrix and best-pairs computations of our model (Sec. 4.1). With the average training window containing 7.4 million unique IPs (with $M \geq 3$, Tab. 2), the lift-matrix has 54.3 *trillion* entries.

Fortunately, the lift-matrix is sparse: just ≈2% of all elements are non-zero (the IP-pair having at least one correlation). Leveraging this, the history (H') is indexed both by time-stamp and IP address. In this manner, lift-matrix rows can be efficiently computed using a multi-set built from

[9]Once computed and persistently stored, rules can be further filtered by support, lift, *etc.*. based on application-specific criteria.

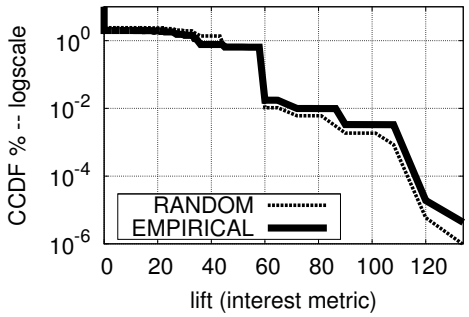

Figure 7: Lift Distribution CCDF

	RANDOM		EMPIRICAL		
LIFT	prob.	exp.	prob.	exp.	**ERR**
≥ 0	100%	7.4 mil.	100%	7.4 mil.	1.00
> 0	2.410%	177,557	1.991%	146,730	1.21
≥ 45	1.370%	100,931	0.768%	56,617	1.78
≥ 60	0.599%	44,122	0.641%	47,270	0.93
≥ 72	0.008%	626	0.014%	1032	0.61
≥ 90	0.006%	446	0.009%	728	0.61
≥ 120	0.002%	137	0.003%	243	0.56
≥ 135	<.001%	0.51	<.001%	1.40	0.36

Table 3: (Prob)abilities and (exp)ected row distributions (length = 7.4 million), per Fig. 7. ERR (error) = exp(random))/exp(empirical)

set intersections (all IPs in some radius). Then, each row is immediately condensed into a best-pairs list, which is truncated at reasonable length and persistently stored.

While we make no claims about the optimality of this approach, at minimum, it makes the problem a tractable one. Keeping the structures/indexes entirely in memory, it takes ≈3 days to compute all 7.4 million best-pairs lists using a multi-threaded implementation and a single machine with modern hardware (8GB+ memory, quad-core processor). The task is trivially parallel-izable and nodes were leased from a large cloud provider to speed experimentation.

Of course, these statistics represent just a single training window, and a live implementation needs retraining. Fortunately, given that blacklisted IPs require at least one week to turnover (Fig. 3), a predictive application would require retraining on approximately the same interval[10]. This simply establishes the feasibility of a working implementation: incremental/online calculations could improve scalability considerably, but are not investigated in this writing.

5. MINING RESULTS

In this section, we examine the results of temporal learning. First, we examine the association rules and their statistical significance (Sec. 5.1). Then, we evaluate these rules' capability to capture botnet membership (Sec. 5.2) and predict future blacklistings (Sec. 5.3). Finally, we discuss these results and suggest refinements to our model (Sec. 5.4).

5.1 Association Rule Significance

Having exhaustively computed lift-pairs for all IPs meeting minimum support, focus now turns to the statistical significance of those pairs and the best-pairs lists which aggregate them. All results are generated using a training window over the three months preceding Dec. 1, 2010 (see Tab. 2), the same one used in predictive evaluation (Sec. 5.3).

Pairwise Lift Distribution: The nature of botnet operations indicates that strong correlations should occur at rates exceeding pure chance; we now test this notion.

To do so, we compare the *empirical* distribution of lift values seen when computing $\mathsf{lift}(H')$ against those of temporally *randomized* versions, H'_{rand}, of the same history (processed until convergent). Fig. 7 and Tab. 3 display this comparison, showing the probability that an IP pair, picked at random, will produce some lift value[11]. Tab. 3 also presents the more intuitive notion of "expected row distribution." That is, if one were to pick a row at random from $\mathsf{lift}(H')$ or $\mathsf{lift}(H'_{\mathsf{rand}})$, what lift values should one expect to see, and in what quantity? (*i.e.,* what should the best-pairs list look like?).

First, we observe that for ≈98% of pairs, $\mathsf{lift} = 0$, indicating they have *no* correlations. In what may seem counter-intuitive, for $\mathsf{lift} \leq 45$, the random distribution outperforms the empirical one. However, this is expected behavior: as a consequence of some pairs correlating well (*i.e.,* at $\mathsf{lift} \geq 60$), the inverse should also hold (*i.e.,* some pairs correlate *worse* than random). Moreover, strong correlations across the entire problem space would be unusual. After all, shared-botnet pairs are likely a *very* small fraction of all pairs. It is unsurprising to see that only about 0.6% of pairs (see Tab. 3) correlate stronger than random chance.

In practice, only the highest lift values require examination, since best-pairs lists are truncated for spatial efficiency. Unfortunately, there are no lift-values (and therefore, no pairs of IPs) that are overwhelmingly correlated. Even in the best cases, where $\mathsf{lift} \geq 135$, there is still a 36% probability the pair is the result of random chance. Clearly, these are not ideal assurances. Nonetheless, by examining the entire best-pairs vector, further traction can be made.

Aggregate Policies: Single lift values encode far less information than the entire best-pairs vector. For example, when assessing $\mathsf{IP_x}$, one should obtain $\mathsf{best\_pairs(IP_x)}$. One could simply examine that head of that list. Alternatively, one could query the top-n pairs using a voting strategy. If a strong-majority agree on a particular assessment/outcome, the error rate (of the assessment) will be a function of voter error-rates and far less than that of any single element.

We can imagine arbitrarily complex policies being written in this fashion, involving: (1) the best pairs list, (2) the associated lift values, (3) voting thresholds, (4) a relative measure of how a best-pairs list compares to its expected-value, and (5) deeper mining, such as the $\mathsf{lift}()$ value for any two-pairs voting in agreement.

We make no claims as to the optimality of any aggregate approach (indeed, they are likely application specific). However, as a proof-of-concept we test one such voting policy in Sec. 5.3, showing that it out-performs naïve pair-wise efforts.

[10]Consider that if $\mathsf{IP_x}$ is blacklisted at this instant, it will be one-week until it is delisted. Given the known blacklist status of $\mathsf{IP_x}$, there is no need to predict it. Thus, any rules created/modified as a result of the current listing are irrelevant until delisting occurs and retraining can be delayed until (at most) that time.

[11]Recall that *lift* does have a probabilistic interpretation: the ratio of observed support to expected (random) support, as a function of a single trial. Given our multi-*trillion* element trial space, lift values become most useful when interpreted relatively.

PROPERTY	{cutwail}	PCT	{kraken}	PCT
unique IP addresses	319,631	100%	318,022	100%
PBL address	271,370	85%	277,581	87%
PBL-DHCP (§3.3)	237,326	74%	254,429	80%
listed in window	216,284	68%	247,040	78%
min. support ≥ 3	123,125	39%	141,043	44%
$\Downarrow$	...	...	...	...
min. support ≥ 3	123,125	100%	141,043	100%
... + PBL address	119,076	97%	137,237	97%
... + PBL-DHCP	106,182	86%	126,659	90%

Table 4: Botnet ground-truth statistics, with focus on IP subset where min. support ≥ 3

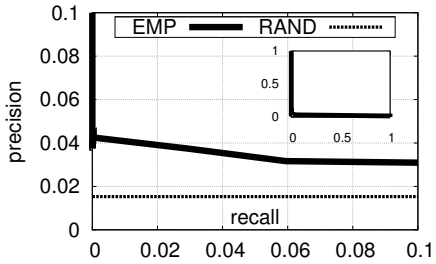

Figure 8: Zoomed precision-recall curve for the shared botnet membership task. (inset) Full graph.

5.2 Case Studies: Cutwail and Kraken

We next evaluate association rules' capability to capture shared botnet membership, using actual botnet data.

Data/Method: We were fortunate to obtain ground-truth regarding the membership of two botnets. First, a list of 319,631 IPs known to be "Cutwail" participants between May 1, 2010 and Aug. 1, 2010 was obtained (see [36]). Second, a list of 318,022 IPS which connected to a "Kraken" sinkhole on May 27, 2010 was procured (see [13, 26]).

Mathematically, we refer to these sets as $\{\mathsf{cutwail}\}$ and $\{\mathsf{kraken}\}$, respectively. See Tab. 4 for statistics about these sets, especially as they relate to the XBL blacklist. Training windows were drawn over the entire data period (Cutwail) or centered about the snapshot date (Kraken). Computation used the same parameters as previously discussed (*i.e.,* $\mathsf{len}(H') = 3$ months, $r = 2$ hours, and $M = 3$).

To evaluate, the $\mathsf{lift}()$ matrices are computed. Then, we characterize and quantify $\mathsf{lift}(\mathsf{IP_x}, \mathsf{IP_y})$ values where $\mathsf{IP_x} \wedge \mathsf{IP_y} \in \{\mathsf{kraken}\}$, those cases where two IPs share botnet membership. The $\{\mathsf{cutwail}\}$ set is handled equivalently.

Results: Preliminarily, we see that 68–78% of known botnet members are listed on the XBL during their respective training window (Tab. 4). It is impossible to determine why the remainder of IPs were not listed: we have no confirmation they even sent spam email. Concentrating on those IPs that were caught, 123k-141k of them were listed 3+ times in the training window, meeting minimum support.

Given that requirement, 39–44% of known botnet members are present in the lift-matrix computation. Focusing specifically on Kraken, we know the size of the lift matrix (and therefore, best-pairs vectors) to be 9.3 million (all IPs meeting min. support in the training window). Thus, if one examines any $\mathsf{best\_pairs}(\mathsf{IP_x}), \mathsf{IP_x} \in \{\mathsf{kraken}\}$ vector, there will be 141,043 elements ($\mathsf{IP_y}$) meeting $\mathsf{IP_x} \wedge \mathsf{IP_y} \in \{\mathsf{kraken}\}$. Thus, the probability of picking an $\mathsf{IP_y}$ at random is 1.513%.

This probability is the control line in Fig. 8, which is the precision-recall curve built from using lift values to predict shared botnet membership. Most encouragingly, we find that 6.0% of shared-membership pairs have non-zero lift values, compared to 2.8% of disjoint ones. Moreover, this is a probability difference present for roughly 95% of recall space.

However, when viewed in absolute terms, results are less inspiring. Even the highest lift values only produce precision values around 4.5% for Kraken data. The inset of Fig. 8 displays the entirety of the zoomed outset graph, showing the classifier's benefit is virtually imperceptible at traditional scale. Cutwail analysis produced similarly poor results.

5.3 Predicting Blacklist Events

The previous section underscores that single IP-pairs are shallow/weak predictors of behavior. Thus, set-based voting strategies are applied to predict *future* blacklist events in the hope that aggregate evidence proves more beneficial. Intuitively, proactive blacklisting can reduce spammer utility and reduce end-user spam exposure.

Data/Method: To gauge predictive capability, we *train* over the 3-months immediately preceding Dec. 1, 2010, with the 5 proceeding days being the *test* period (an interval justified in Sec. 4.3). Tab. 2 shows some basic properties about this training period, chosen due to its recency[12].

The prediction process is as follows: the blacklist history is *replayed* in-order. At any point, any number (including zero) of *predictions* may be made, leveraging only prior information. A prediction is a statement of the expectation that some IP address (for example, $\mathsf{IP_x}$) will soon be blacklisted. Predictions are evaluated using:

```
IF IPx actively blacklisted || IPx ∈ {predictions} : ∅
ELSE :  {predictions} = IPx ∪ {predictions}
  IF IPx listed in next r′ time : correct++;
  ELSE : incorrect++;
```

Notice that an IP cannot be predicted multiple times, preventing ballot stuffing. Our evaluation metric is the *percentage of correct predictions, to all predictions.* This metric is not ideal: an incorrect prediction is not equivalent to an email false-positive (the IP might not be sending *any* mail). In contrast, correct predictions have a strong interpretation. Given that the IP is soon blacklisted, we know that it is actively (or will soon begin) sending spam email: traffic that proactive blacklisting can mitigate. Lastly, the "success window" is set to one day (*i.e.,* $r' = 24$ hours).

Given these ground rules, a *policy* must be written to make predictions. Fig. 9 visualizes our proof of concept approach (at replay position t_n). Summarily, we use a structure, $\{\mathsf{recent}\}$, to store XBL listings occurring the past r time. Whenever a new listing occurs, it is added to this set, which is run over all best-pairs lists[13], calculating the union. The size of the union is then normalized by the length of the list, producing a metric useful in predictive thresholding. Less formally: "if y% of $\mathsf{IP_x}$'s best-pairs have recently been blacklisted, predict $\mathsf{IP_x}$ to be blacklisted".

[12] 5-day sliding windows were computed for all of Dec. 2010. Result similarity and brevity preclude presentation of this data.

[13] This operation is why $\mathsf{lift}(H')$ must be exhaustively computed. It is impossible to know *a priori* where a recently listed IP might appear in best-pairs rule predicates.

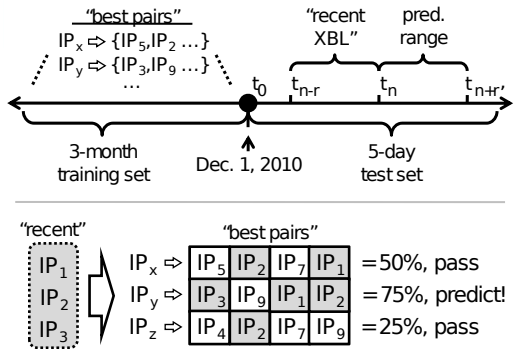

Figure 9: (a) Structures in predictive application. (b) Filtering recent XBL history over best pairs.

Results: Preliminarily, the 3-month training window has 41.4 million unique IPs, reducing to 7.3 million with minimum support (Tab. 2). There are 3.5 million listing events in the 5-day test period, with 847,186 (23.8%) of those IPs also appearing in the supported training set. This is an upper-bound on the number of predictable blacklistings.

Given $r' = 24$ hours, we can simulate the random probability of a correct prediction to be 2.77%. Implementing the voting policy described above (and visualized in Fig. 9) at varying prediction thresholds produces the results of Tab. 5.

Tab. 5 suggests there is some benefit in using aggregate evidence, rather than simple lift-pairs. For example, performance is 25%+ at points, while the precision never exceeded 4% in the botnet-membership task. Where strong voting majorities exist, they are accurate. However, such majorities seem hard to amass – making them useful over little problem space (just 93 of 847k listings can be found at 25%+ accuracy). Unfortunately, this likely speaks to the high-level of noise present across the best-pairs vectors. Clearly, the technique is inappropriate as a stand-alone predictor. However, with scalability improvements, it could be an incremental feature in a more robust network-level classifier [20, 39].

5.4 Discussion & Refinement

While *many* points could be discussed post-mortem, we prefer to streamline discussion on three acute issues:

Dynamic IPs: From the outset, we recognized that that some elements of spamming operations were too dynamic to be pattern-mined. In particular, DHCP space was identified as one area of concern. However, we assumed that such addresses would naturally exclude themselves from problem space by failing to accrue minimum support (Sec. 4.2).

Clearly, this assumption was flawed. Tab. 4 shows that 90%+ (possibly 97%) of botnet IPs meeting minimum support are dynamic. While a fraction of these IPs may have lengthy leases (*e.g.,* cable Internet), this seems unlikely to account for this entire set. Moreover, botnet operators may knowingly reserve static IP addresses for control channels, meaning they never send mail or become blacklisted (Tab. 4). Thus, these IPs fall outside the scope of rule learning (a segment we anticipated successfully mining).

How these dynamic IPs are able to attain minimum support remains an open question. However, it is clear this area demands more research. Our analysis suggests that dynamic IPs may be becoming more prevalent in botnet operations than previously reported ([40], see Sec. 3.3). The reliable identification of DHCP space holds enormous potential, given that 96% of mail servers at dynamic IPs send only spam [40]. If it were possible to not just detect dynamic IPs, but also estimate their churn rates, one could better model how association rules retain/lose value as time passes.

THR	CORR%	×RAND	LIST
$> 0\%$	3.47%	1.25×	47,184
$\geq 10\%$	3.66%	1.32×	5,600
$\geq 20\%$	17.34%	6.26×	161
$\geq 25\%$	25.98%	9.38×	93
$\geq 30\%$	35.63%	12.86×	57

Table 5: Prediction at varying (thr)esholds: absolute (corr)ectness; relative to (rand)om chance; and correctly predicted (list)ings count.

Window Sizes: As discussed in Sec. 4.2, there are contradictory factors in sizing the training window and correlation radius. Casual experimentation with alternative parameters (*i.e.,* aside from $\mathsf{len}(H') = 3$ months and $r = 2$ hours) produced no significant benefits. Even rules which well-capture training window patterns may be "expired" when they need to be applied, Thus, we advocate improving our model with less rigid notions of correlation and history.

For example, correlations could be modeled using a Gaussian function. In this manner, IPs which correlate extremely tightly (*i.e.,* list at the exact same time) are weighted more heavily than those occurring near the radius boundary. Similarly, training windows can time-decay correlation events so that those occurring in the recent past carry more weight.

As the model grows increasingly complex, it is important to consider the scalability of these efforts. Indeed, scalability improvements over content-processing approaches were a primary motivator of this research, and one reason initial efforts were so simply modeled. Moreover, efficiency is critical in making this a practical working system.

Appropriateness of Blacklist Data: More fundamentally, we must investigate if blacklists are the best ground-truth over which to temporally mine. Despite our extensive investment in blacklists, the answer is "probably not." Blacklists are attractive due to their concise encoding of global spam histories (500 million listings = 12GB storage). However, this same fact may be precisely their weakness.

Blacklists are a second-hand and aggregate data source whose latency and lack of coverage [33] may skew known results about botnet operation. Their binary-nature and static TTL delisting (of the XBL) provide little insight about the behavior of an address that is actively listed. Most damningly, blacklists tend to evolve at rates not permitting significant support to develop. Minimal support generates ambiguous rules, precisely why we described aggregate voting strategies (Sec. 5.1) to better utilize available information.

Ultimately, we must consider that our model may be most appropriate over an alternative ground-truth. For example, it could be installed on an email server. In this manner, individual spam mail arrival times and source IPs could be recorded and mined. Quantitative (non-binary) aggregates could be derived and timestamps would be known accurate (without latency). Further, metadata fields collected by the mail server [20] could be helpful in inferring campaign groupings, without ever inspecting message content.

6. CONCLUSIONS

Herein, we have described and tested a model for temporal association learning over spam blacklist data. Unfortunately, this effort delivered only trivial improvements over control classifiers at significant computational cost. Nonetheless, there is still utility in this negative result.

Our measurement study contributes to the growing knowledge about spam behaviors and does so from a blacklist perspective. Moreover, our proposed temporal learning model for blacklist data could serve as a foundation for more refined efforts; those better capturing the dynamics of botnet operation. Finally, evaluating our model allowed us to identify not only our own weaknesses, but also future areas of research that will benefit the broader anti-spam effort.

Acknowledgements

The authors are grateful for the help of Paul Royal and his co-authors [13], who provided the Kraken botnet packet capture. Similarly, Gianluca Stringhini and his co-authors [36] are thanked for the Cutwail botnet data. The authors would also like to acknowledge the helpful advice of University of Pennsylvania professors Sampath Kannan, Lyle Ungar, and Linda Zhao – and UPenn Ph.D. students Adam Aviv and Jian Chang.

References

[1] Spamhaus Project. http://www.spamhaus.org/.
[2] SpamLinks: Legal issues. http://spamlinks.net/legal.htm.
[3] Univ. of Oregon Route Views. http://www.routeviews.org/.
[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of the International Conference on Very Large Databases*, pages 478–499, 1994.
[5] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the 11th Intl. Conf. on Data Engineering*, 1995.
[6] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker. Spamscatter: Characterizing Internet scam hosting infrastructure. In *Proc. of the USENIX Security Symposium*, 2007.
[7] C. M. Antunes and A. L. Oliveira. Temporal data mining: An overview. In *KDD Wkshp. on Temporal Mining*, 2001.
[8] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the Web. In *WWW*, 2007.
[9] P. H. Calais, D. E. V. Pires, D. O. Guedes, W. M. Jr., C. Hoepers, and K. Steding-Jessen. A campaign-based characterization of spamming strategies. In *CEAS'08: Proceedings of the 5th Conference on Email and Anti-Spam*, 2008.
[10] X. Chen and I. Petrounias. Discovering temporal association rules: Algorithms, language and system. In *Proc. of the 16th Intl. Conference on Data Engineering*, 2000.
[11] Comcast Corporation. Forum search: Static or dynamic IP address? http://forums.comcast.com/.
[12] C. J. Dietrich and C. Rossow. Empirical research on IP blacklisting. In *CEAS'08: Proceedings of the Fifth Conference on Email and Anti-Spam*, 2008.
[13] A. Dinaburg, P. Royal, M. Sharif, and W. Lee. Ether: Malware analysis via hardware virtualization extensions. In *CCS'08: Computer and Communications Security*, 2008.
[14] J. E. Dunn. Botnet PCs stay infected for years. *TechWorld*, September 2009. http://www.networkworld.com/news/2009/092209-botnet-pcs-stay-infected-for.html.
[15] D. Fischer. Spam volume recovers after holiday break. ThreatPost: The Kapersky Lab Security News Service. http://threatpost.com/, January 2011.
[16] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comp. Surveys*, 38(9), 2006.
[17] T. F. Gharib, H. Nassar, M. Taha, and A. Abraham. An efficient algorithm for incremental mining of temporal association rules. *Data Knowledge Engineering*, 2010.
[18] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering analysis of network traffic for protocol and structure-independent botnet detection. In *SECURITY'08: Proc. of the 17th USENIX Security Symposium*, 2008.
[19] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. of the ACM-SIGMID Intl. Conf. on Management of Data*, pages 1–12, 2000.
[20] S. Hao, N. A. Syed, N. Feamster, A. G. Gray, and S. Krasser. Detecting spammers with SNARE: Spatio-temporal network-level automated reputation engine. In *Proc. of the USENIX Security Symposium*, 2009.
[21] N. Jiang and L. Gruenwald. Research issues in data stream association rule mining. *SIGMOD Record*, 35, March 2006.
[22] J. P. John, A. Moschuck, S. D. Gribble, and A. Krishnamurthy. Studying spamming botnets using Botlab. In *NSDI: Networked Systems Design and Implementation*, 2009.
[23] J. Jung and E. Sit. An empirical study of spam traffic and the use of DNS black lists. In *Proc. of the 4th ACM SIGCOMM Conference on Internet Measurement*, pages 370–375, 2004.
[24] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage. Spamalytics: An empirical market analysis of spam marketing conversion. In *CCS'08: Computer and Communications Security*, 2008.
[25] F. Li and M.-H. Hsieh. An empirical study of clustering behavior of spammers and group-based anti-spam strategies. In *CEAS'06: Third Conf. on Email and Anti-Spam*, 2006.
[26] A. Moscaritolo. Kraken re-emerges 318,000 nodes strong. SC Magazine. http://www.scmagazineus.com/, June 2010.
[27] S. Nagaraja, P. Mittal, C. yao Hong, M. Caesar, and N. Borisov. BotGrep: Finding P2P bots with structured graph analysis. In *USENIX Security Symposium*, 2010.
[28] A. Pathak, F. Qian, Y. C. Hu1, Z. M. Mao, and S. Ranjan. Botnet spam campaigns can be long lasting: Evidence, implications, and analysis. In *SIGMETRICS'09*, 2009.
[29] A. Pitsillidis, K. Levchenko, C. Kreibich, C. Kanich, G. M. Voelker, V. Paxson, N. Weaver, and S. Savage. Botnet judo: Fighting spam with itself. In *NDSS*, 2010.
[30] Z. Qian, Z. Mao, Y. Xie, and F. Yu. On network-level clusters for spam detection. In *NDSS*, 2010.
[31] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *SIGCOMM*, 2006.
[32] A. Ramachandran, N. Feamster, and S. Vempala. Filtering spam with behavioral blacklisting. In *CCS*, 2007.
[33] S. Sinha, M. Bailey, and F. Jahanian. Improving spam blacklisting through dynamic thresholding and speculative aggregation. In *NDSS*, 2010.
[34] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of the 5th Intl. Conf. on Extending Database Technology*, 1996.
[35] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *CCS'09: Computer and Communications Security*, 2009.
[36] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna. The underground economy of spam: A botmaster's persepective of coordinating large-scale spam campaigns. In *LEET'11: Large-Scale Exploits and Emergent Threats*, 2011.
[37] Symantec MessageLabs Intelligence. 2010 Security Report. http://www.messagelabs.com/resources/mlireports.aspx.
[38] S. Venkataraman, S. Sen, O. Spatscheck, P. Haffner, and D. Song. Exploiting network structure for proactive spam mitigation. In *USENIX Security Symposium*, 2007.
[39] A. G. West, A. J. Aviv, J. Chang, and I. Lee. Spam mitigation using spatio-temporal reputations from blacklist history. In *ACSAC '10: Proceedings of the 26th Annual Computer Security Applications Conference*, December 2010.
[40] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber. How dynamic are IP addresses? In *Proceedings of SIGCOMM '07*, 2007.
[41] Y. Xie, F. Yu, K. Achan, R. Paingrahy, G. Hulten, and I. Osipkov. Spamming botnets: Signatures and characteristics. In *Proc. of SIGCOMM '08*, 2008.
[42] J. S. Yoo and S. Shekhar. Similarity-profiled temporal association mining. *IEEE Trans. on KDE*, 21(8), August 2009.
[43] L. Zhuang, J. Dunagan, D. R. Simon, H. J. Wang, and J. Tygar. Characterizing botnets from email spam records. In *LEET: Large-Scale Exploits and Emergent Threats*, 2008.