Populated IP Addresses — Classification and Applications

Chi-Yao Hong*
UIUC
cyhong@illinois.edu

Fang Yu
MSR Silicon Valley
fangyu@microsoft.com

Yinglian Xie
MSR Silicon Valley
yxie@microsoft.com

ABSTRACT

Populated IP addresses (PIP) — IP addresses that are associated with a large number of user requests — are important for online service providers to efficiently allocate resources and to detect attacks. While some PIPs serve legitimate users, many others are heavily abused by attackers to conduct malicious activities such as scams, phishing, and malware distribution. Unfortunately, commercial proxy lists like Quova have a low coverage of PIP addresses and offer little support for distinguishing good PIPs from abused ones. In this study, we propose PIPMiner, a fully automated method to extract and classify PIPs through analyzing service logs. Our methods combine machine learning and time series analysis to distinguish good PIPs from abused ones with over 99.6% accuracy. When applying the derived PIP list to several applications, we can identify millions of malicious Windows Live accounts right on the day of their sign-ups, and detect millions of malicious Hotmail accounts well before the current detection system captures them.

Categories and Subject Descriptors: C.2.0 [Computer-Communication Networks]: General – *security and protection*

Keywords: Populated IP addresses, proxy, IP blacklisting, spam detection.

1. INTRODUCTION

Online services such as Web-based email, search, and online social networks are becoming increasingly popular. While these services have become everyday essentials for billions of users, they are also heavily abused by attackers for nefarious activities such as spamming, phishing, and identity theft [14, 29].

To limit the damage of these attacks, online service providers often rely on *IP addresses* to perform blacklisting and service throttling [15, 19, 24, 26, 29]. However, IP-based techniques work effectively on only those IP addresses that are relatively static and related to a few users. For IP addresses that are associated with a large number of user requests, they must be treated differently, as blocking/rate-limiting one such address may potentially affect many individual users.

In this paper, we define IP addresses that are associated with a large number of user requests as *Populated IP* (PIP) addresses. It is related, but not equivalent to the traditional concept of proxies, network address translators (NATs), gateways, or other middleboxes.

On the one hand, not all proxies, NATs, or gateways are PIP addresses. Some may be very infrequently used and thus are not of interest to online service providers.

On the other hand, while some PIP addresses may belong to proxies or big NATs, many others are not real proxies. Some are dial-up or mobile IPs that have high churn rates. Others include IP addresses from large services, such as Facebook that connects to Hotmail to obtain user email contacts. Additionally, not all PIPs are associated with a large number of actual users. Although many *good* PIPs like enterprise-level proxies are associated with a large number of actual users, some *abused* PIPs may be associated with few real users but a large number of fake user accounts controlled by attackers. In an extreme case, *bad* PIPs may be entirely set up by attackers. For example, we find that >30% of the IP addresses that issue more than 20 sign-up requests to Windows Live per day are actually controlled by attackers, with all sign-ups for malicious uses.

Therefore, the knowledge of PIPs and their abused scenarios are critical to service providers and can help them make informed decisions. For example, a relaxed rate-limiting threshold should be given to good PIPs, and more stringent rate-limiting policies should be applied to abused or bad PIPs. In addition, the information of PIPs can be used in conjunction with other defense mechanisms to detect attacks faster and more confidently.

Classifying PIPs is a challenging task for several reasons. First, ISPs and network operators consider the size and distribution of customer populations confidential and rarely publish their network usage information. Second, some PIP addresses are dynamic, e.g., those at small coffee shops with user population sizes changing frequently. Third, good PIPs and bad PIPs can locate next to each other in the IP address space. For example, attackers can buy or compromise Web hosting IPs that are right next to the IPs of legitimate services. In addition, a good PIP can temporarily be abused. Due to these challenges, not surprisingly, we find that commercial proxy lists offer a low precision in identifying PIPs

*Work was done while interning at MSR Silicon Valley.

CCS'12, October 16–18, 2012, Raleigh, North Carolina, USA.

and provide no support for distinguishing good PIPs from bad ones.

In this paper, we introduce *PIPMiner*, a fully automated method to extract and classify PIPs. We leverage machine-learning techniques to capture the patterns of good and bad PIPs. In particular, we use support vector machine (SVM), a state-of-the-art supervised classification technique, to train a robust classifier using a number of salient features extracted from service logs. These features capture the invariants of PIP properties, ranging from intrinsic population characteristics to more advanced time series and network level properties.

PIPMiner uses a small set of PIPs with labels to train a classifier and applies the classifier to PIPs that do not have labels. After the classification, PIPMiner outputs a confidence score for each PIP to quantify the abuse likelihood, with high scores given to good PIPs, intermediate scores given to abused PIPs, and low scores given to attacker-created PIPs. We implement PIPMiner on top of DryadLINQ [30], a distributed programming model for large-scale computing. Using a 240-machine cluster, PIPMiner processes a month-long Hotmail user login dataset of 296 GB in only 1.5 hours.

Our primary contribution is the concept of PIPs, which is different from the traditional concept of proxy IP addresses. The notion of good/bad PIPs is much more relevant in the security context, as populated IP addresses often become security loopholes: some service providers just whitelist PIPs because they cannot distinguish good PIPs from bad ones. Our second contribution is a method to automatically derive and classify PIPs. We implement PIPMiner and demonstrate its value through multiple large-scale security applications. Our key results include:

- **PIPMiner is effective.** PIPMiner identifies 1.7 million PIP addresses. Our evaluation demonstrates that PIPMiner can classify PIP addresses with 99.59% accuracy and is robust against evasion attacks (§4).
- **PIP results can help detect attacks.** Our PIP list helps detect 1.2 million malicious Hotmail accounts with 98.6% precision, months before the current user reputation system captures them (§5.2).
- **PIP results are applicable across datasets and can help classify non-PIP traffic.** When applying the PIP list derived from the Hotmail log to the Windows Live ID sign-up problem, we detect more than 3 million bad accounts right on the day of their sign-ups with 97% precision. In particular, over 77% of the detected bad accounts are registered using non-PIP addresses (§5.1).
- **PIP list is more applicable than proxy list.** We compare our PIP list to the GeoPoint data, a big commercial IP list provided by Quova. Using GeoPoint's IP Routing Type (IPRT) field, we find that 99.9% of the proxy IPs are not associated with any requests in our application logs (§2). For the PIP addresses that are associated with many good requests, Quova's proxy list misses a vast majority of them (Appendix B).

Keyword (case insensitive)	Percentage of coverage
PROXY	0.0054
GATE ∪ GW	0.0049
NAT	0.0046

Table 1: ***The keyword coverage of PIP addresses through rDNS lookup. The PIP address list is derived by PIPMiner using the Hotmail user login dataset (§3).***

The rest of the paper is organized as follows. We start by discussing related work and background (§2). We then describe our overall approach and design details (§3), followed by a comprehensive performance evaluation of our approach (§4). We apply the PIP results to two security applications (§5) before we conclude (§6).

2. RELATED WORK

In this section, we first discuss related work in classifying populated IP addresses (§2.1). Since we use email spam detection as the application of our work, we also briefly review existing approaches in this field (§2.2).

2.1 Populated IP Classification

To the best of our knowledge, PIPMiner is the first attempt towards automated PIP address classification on a global scale. Existing IP-based detection systems do not work effectively on PIPs because their characteristics differ significantly from normal IP addresses [28]. Hence, PIPs are often ignored in previous studies [14, 31]. Meanwhile, our study suggests that they are heavily exploited by attackers. PIPMiner focuses on these PIPs and fills the gap.

Next we review work on global IP addresses, proxies, NATs and gateways because they could overlap with and are related to PIP addresses.

Some PIP addresses might be inferred by examining Reverse DNS (rDNS) [10] records. An rDNS record maps an IP address into a domain name, offering a potential way to infer its address properties. For example, the rDNS record for an IP address 64.12.116.6 corresponds to a domain name `cache-mtc-aa02.proxy.aol.com`, which suggests that the IP address is used as a cache proxy in the AOL network. However, only a fraction of PIP addresses have rDNS records (62.6% in our experiment), and only 1.49% of them contain useful keywords for PIP identification (Table 1).

Certain services like CoDeen [25] and ToR [9] provide open access to a set of distributed nodes. These nodes are PIP candidates because they are accessible by many Internet users. While these IP addresses can be easily collected from these services, they are merely a small fraction of the total PIPs. For example, CoDeen has around 300 nodes on PlanetLab and ToR has fewer than 1000 exit nodes [4]. Our PIP list (1.7 million IP addresses; §4.1) covers most of these IPs.

Quova offers the most well-known commercial proxy list [3]. It produces two types of information that may be used to identify proxy IPs. The first is the IP routing type, such as regional proxy and mobile gateway. The second is the anonymizer status, which records whether an IP is associated with a known anonymous proxy. The first type includes more than 300 million proxy IP addresses as of July, 2011, orders of magnitude more than the second type. However, we find that most of them do not display PIP-like behavior:

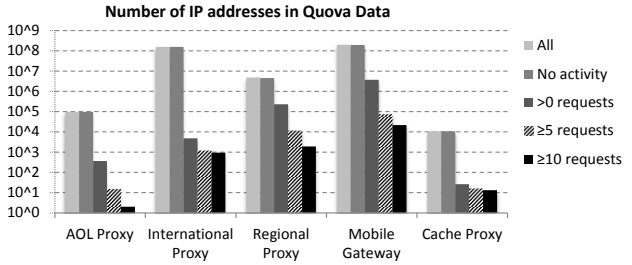

Figure 1: ***The number of proxy IP addresses in the Quova's GeoPoint dataset (using IP Routing Type field for proxy identification) as of July,*** 2011. ***The number of IP addresses with*** ≥ 10 ***Windows Live ID sign-up requests in a month is*** 3 ***to*** 4 ***magnitudes smaller than that of the inactive ones.***

over 99% of them are not associated with any user sign-up request to Windows Live (Figure 1). Our work aims to complement Quova with an automated PIP detection and classification method.

While there is no existing approach that automatically distinguishes good PIP addresses from bad ones, a large body of previous work has focused on identifying other important IP properties. Cai et al. studied how network address blocks are utilized through frequent ICMP probing [6]. Katz-Bassett et al. proposed a novel method to estimate the geographic locations of IP addresses by solving constraints derived from Internet topology measurements [16]. The closest study to PIP detection is the NAT and proxy detection work by Casado and Freedman [7]. Also closely related, Metwally and Paduano proposed methods to estimate the user population size per IP address [18]. Here, PIPMiner differs from these earlier efforts by focusing on PIP addresses, with special emphases on distinguishing good PIPs from bad ones.

2.2 Spam Detection

Email spamming has been a long studied topic and many spam detection methods have been proposed. In this paper, we focus on the email service abuse problem where attackers register or compromise millions of accounts and use them to send out spam emails.

To fight against such service abuse, service providers often monitor user behaviors carefully and detect malicious users by their actions, e.g., sending out spam emails, failing CAPTCHA tests, or exhibiting suspicious correlated behaviors [27, 31]. Since these detection methods rely on user activities, when they detect a malicious user, the user might have already conducted some malicious activities. PIPMiner complements these detection methods by focusing on PIP addresses that a user leverages to perform sign-up or login. Therefore, the proposed system can flag new users right on the day of the sign-ups, or the time they log in.

In addition, our derived malicious PIP list serves as an early signal to help service providers make informed decisions on whether to block or rate-limit PIP-associated accounts, and how to better allocate resources to PIP addresses in the presence of attacks. In this regard, the PIP list is related to IP blacklists such as DNS Blacklists (DNSBLs). DNSBLs is widely used for spam detection. For example, Jung and Sit observed the existence of low-profile spam sources by characterizing DNSBL traffic [15]. Ramachandran et al. proposed novel detection approaches by analyzing the sending patterns of network traffic (e.g., lookup traffic to DNSBL) [22, 23]. The main difference between our approach and IP blacklisting is that we not only provide a list of malicious IP addresses, but also a model to identify them. The model can help identify new malicious PIPs over time. In addition, the results of PIPMiner can be applied to different applications, such as detecting spammers (§5.2) or identifying malicious sign-ups (§5.1). Therefore, it is more general than systems that detect spam only (e.g., SNARE [12]).

3. SYSTEM DESIGN

Our primary goal is to design an automated method to identify PIPs and classify them. Towards this goal, we take a data-driven approach using service logs that are readily available to all service providers. However, simple history-based anomaly detection methods often fail to differentiate real attacks from normal behavior changes accurately (results in §4.2). Therefore, we employ a machine learning based approach that extracts features from training data. We propose a comprehensive set of features to capture the invariants of PIP properties, from intrinsic population characteristics to more advanced time series features.

However, real data is often noisy and there is no perfect training data. In addition, some PIPs are used by both legitimate users and attackers, so they display mixed behaviors that make our classification difficult. To deal with these issues, we train a non-linear support vector machine classifier that is highly tolerant of noise in input data. We represent the classification result as a score between -1 and 1 to indicate the likelihood of a PIP address being abused by attackers.

3.1 System Flow

Figure 2 shows the overall system flow of PIPMiner. The system reads in logs from one or more online services. It first selects PIPs and extracts features for each PIP. Then, PIPMiner labels a subset of data based on application-level information. For PIPs that have clear labels, PIPMiner uses them as training data to produce a classifier. Using the classifier, PIPMiner labels the remaining PIPs and outputs a score quantifying the likelihood of a PIP being good (a high value), bad (a low value), or abused (an intermediate value). The output PIP list with scores will be later used to combat attacks (§5).

Input Data: The primary online service log used in our study is a month-long user login trace collected at Hotmail in August, 2010. Each entry in the data contains an anonymized account ID, a source IP address, and a timestamp. To help derive application-specific features, we also use Hotmail account data, which provides account-level information such as the registration month of an account.

PIPMiner can potentially be applied to data from other online services. For example, it can help detect malicious account creations, malicious postings to online social networks, or the abuse usage of free storage services. It can also help Web servers fight against denial-of-service attacks by distinguishing good PIPs from bad ones.

PIP Selection: The first step of PIPMiner is to identify PIP addresses. PIPMiner first identifies PIPs by selecting

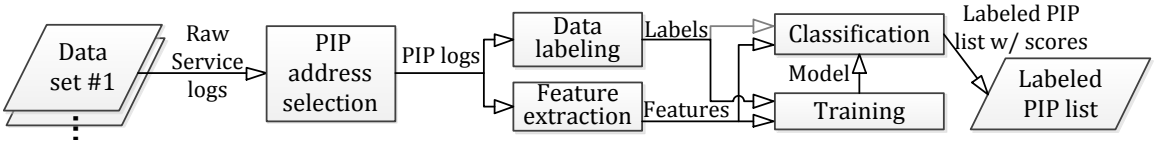

Figure 2: ***PIPMiner system flow.***

IP addresses that are associated with at least r_{L} requests. However, simply counting the number of requests per IP is not sufficient, as many of these requests could be from a few user accounts, e.g., through Outlook client's auto sync. Therefore, PIPMiner also considers *account population size*. PIPMiner marks an IP address as a PIP address if the IP address has been used by at least u_{M} accounts, together accounting for at least r_{M} requests. In our current implementation, we empirically set r_{L} to $1,000$ requests, u_{M} to 10 accounts, and r_{M} to 300 requests. However, PIPMiner performance is not very sensitive to these thresholds, which we will later show in Figure 6d.

Data Labeling: PIPMiner relies on a small set of training data obtained from service providers. In our implementation, Hotmail provides us with the account reputation scores that are computed based on account behavior history (e.g., spam sent and failed CAPTCHA tests) and external reports (e.g., rejected emails and user marked spams). Using the reputation scores, we classify user accounts as good, suspicious, or malicious, and we label an IP address as good if the majority of its requests are made by good accounts, and an IP address as bad if most of its requests are issued by malicious accounts. Since the reputation system relies on carefully observing the account behaviors over time, it is hard for the reputation system to accurately classify new accounts with few activities. Therefore, only a fraction of the PIPs have clear labels, while the remaining IP addresses are unlabeled and their goodness remains unknown.

Feature Extraction: During feature extraction, the system constructs a set of PIP features, ranging from account-level features (e.g., how many user accounts send requests via the PIP?), time series-related features (e.g., do the aggregated requests exhibit human-like behavior such as diurnal patterns in the time domain?), to PIP block level features (e.g., aggregated features from neighboring PIPs). We will explore the motivations behind these features in §3.2.

Training and Classification: In this study, we train a non-linear support vector machine classifier due to its robustness to noisy input, efficiency, and high accuracy. After the classifier is trained using labeled PIPs, we use it to classify unlabeled PIPs, resulting in a full list of classified PIPs. We further convert the classifier decision value to a score of how likely a PIP address is abused by attackers. It can be used to detect abused cases in the future and also in other applications, as good PIPs can temporary turn bad when they are abused. The details of the training and classification algorithms are in §3.3.

Having explained the system flow, next we present the proposed features (§3.2) and training methodologies (§3.3).

3.2 Features

We train a support vector machine with a collection of robust features that discern between good and bad PIPs.

Feature	Descriptions
Population size	The number of accounts
Request size	The number of requests
Request per account	The number of requests per account
New account rate	The percentage of accounts appear only in the last x days
Account request distribution	The distribution of the number of requests per account
Account stickiness distribution	The distribution of the number of PIPs associated with an account

Table 2: ***Population features***

In particular, we propose three sets of quantitative features that can be automatically derived from online service logs. *Population Features* (§3.2.1) capture aggregated user characteristics and help distinguish stable PIPs with many users from those that are dominantly used by only a few heavy users. *Time Series Features* (§3.2.2) model the detailed request patterns and help distinguish long-lasting good PIPs from bad PIPs that have sudden peaks. *IP Block Level Features* (§3.2.3) aggregate IP block level activities and help recognize proxy farms.

A subset of the features that we use (user/request density, user stickiness and diurnal pattern) have been used in other security-related studies [12, 23, 28]. However, we introduce new features and we later show that the performance improvement by our new features is sizable (§4.2).

3.2.1 Population Features

Population features capture the aggregated account characteristics of each individual PIP. We consider basic statistics such as the number of accounts, the percentage of new accounts, the number of requests, and the average number of requests per account, as shown in Table 2.

We also include features derived from basic distributions. We first look at activities of *individual PIPs*. The account-request distribution records the distribution of the number of requests per account. It reflects whether accounts on the PIP have a roughly even distribution of request attempts, or whether the requests are generated by just a few heavy accounts. In addition, we also look at activities across *multiple PIPs*. In particular, we look at *account stickiness distribution*, which records the number of other PIPs used by each account.

After obtaining a distribution, we bucket the distribution into 20 bins and then extract features such as the peak values, the central moments, and the percentage of the accounts/requests in the top 1/3/5 bins.

3.2.2 Time Series Features

The population features alone may not be enough to distinguish good PIPs from bad ones. For example, we find that the aggregated distribution of the number of requests

Feature	Descriptions
On/off period	The fraction of time units having $\geq x$ requests
Diurnal pattern	The percentage of requests and accounts in the peak and the dip periods of the day
Weekly pattern	The percentage of requests in weekdays and weekends
Account binding time	The distribution of accounts' binding time (time between the first and the last requests)
Inter request time	The distribution of accounts' medium inter-request time
Predictability	The number and the density of anomalies derived from the Holt-Winters forecasting model

Table 3: ***Time series features.***

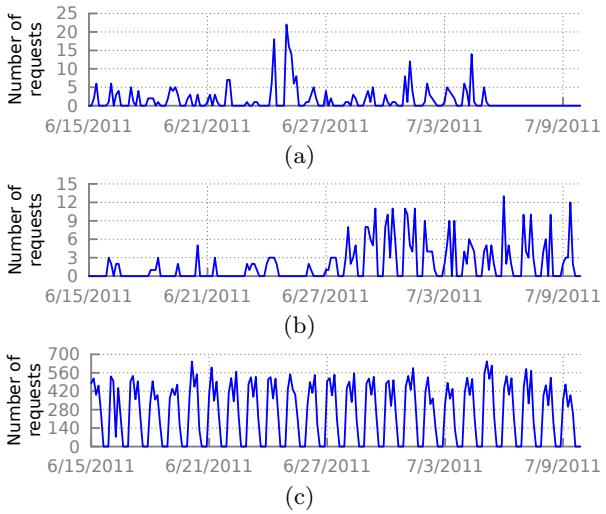

Figure 3: ***(a) and (b): Time series of the number of requests from two randomly selected PIP addresses that belong to the same PIP block. (c): Block level time series, i.e., the time series of the total number of requests aggregated from all the PIP addresses in the same PIP block. The selected PIP block has*** 94 ***PIP addresses, and the figures are plotted in a time granularity of*** 3 ***hours. All the PIPs in the PIP block are labeled as good.***

per good PIP looks nearly identical to that of bad PIPs. One reason is that population features do not capture the detailed request-arriving patterns. PIPs that are active for only a short but intensive period (a typical behavior of bad PIPs) may have similar population features as PIPs with activities that are spread out in a long duration with clear diurnal patterns (a typical behavior of good PIPs). In this subsection, we derive time series features to help distinguish good PIPs from bad ones.

To derive time series features, we first construct a time series of request counts, denoted by $T[\cdot]$, in a 3-hour precision. In particular, $T[i]$ is the number of logins in the i-th time unit. Thus there are $k = 8$ time units per day.

Additionally, we construct a time-of-the-day series $T_D[\cdot]$ such that the j-th element of the time series is the total number of requests in the j-th time unit over days: $T_D[j] = \sum_{k=0,1,\ldots} T[8j + k]$. Then we derive the peak time $P = \arg\max_j\{T_D[j]\}$ and the dip time $D = \arg\min_j\{T_D[j]\}$, and the time difference between the peak and the dip, i.e., $|P - D|$.

Table 3 lists the features that we derive from the time series. It includes on/off period, diurnal pattern, and weekly pattern. The on/off period feature records the fraction of time units having $\geq x$ requests in $T[\cdot]$. To assess if the time-of-the-day series displays a diurnal pattern, we check the time and the load difference between the peak and the dip. To check weekday pattern, we record the percentage of requests and accounts in weekdays and weekends. Due to timezone differences, the time for weekends of different regions might be different. One option is to look up the timezone of each IP address. In our experiment, we adopt a simper method — finding two consecutive days in a week such that the sum of the number of requests from these two days are minimal. For PIPs that have more traffic in weekends than weekdays, e.g., coffee shop IPs, our approach considers the least busy day during the week as weekend. But this does not affect the correctness because our main goal is to detect the repeated weekly patterns, i.e., the least busy days of the week repeat over different weeks.

We also study two distributions that reflect the request-arrival pattern of each account. The first is the account-binding-time distribution that captures the distribution between the last and the first account request time. This reflects the accounts' binding to an IP address. In addition, we check the inter-request time distribution that captures the regularity of account requests.

We also apply a forecasting model on time series to detect abnormal period, and we leave details to Appendix C.

3.2.3 IP Block Level Features

We find that many regional PIPs manifest clear time-series features such as coherent diurnal patterns and the weekday-weekend pattern. For example, for office IP addresses, the daily peaks are usually around 4 PM, while the daily dips are at around 4 AM. The peak load and the dip load differ significantly and they are usually predictable using our forecasting models.

However, there are exceptions. In particular, large proxy farms often redirect traffic to different outgoing network interfaces for load balancing purposes. They may rotate IP addresses at random intervals or at pre-configured timestamps from a pool of neighboring IP addresses (e.g., [2]). Such strategies will cause us to generate time series with high variations (e.g., sudden increase), as shown in Figure 3a and Figure 3b. We might mistakenly conclude that these cases are suspicious if we look at time series of individual PIPs in an isolated way. To resolve the problem, we aggregate traffic using IP blocks and use information of neighboring PIP addresses to help recognize PIP patterns, as shown in Figure 3c.

We use the following two criteria to determine neighboring IP addresses:

1. Neighboring IPs must be announced by the same AS. This ensures that the neighboring IP addresses are under a single administrative domain.

2. Neighboring IPs are continuous over the IP address space, and each neighboring IP is itself a PIP. This

ensures that there is no significant gap between a PIP and its neighboring PIPs.

After identifying neighboring PIPs, *PIP block IDs* are assigned to PIP addresses such that PIP addresses have the same block ID if and only if they are neighboring PIPs. Finally, we generate block-level features: for each PIP address, we extract all the features (i.e., population and time series features) using the requests aggregated from *all* the PIPs in the same block.

3.3 Training and Classification

Feature Preprocessing: For training data, we perform feature-wise normalization to shift and re-scale each feature value so that they lie within the range of $[0, 1]$. We apply the same transformation as in the training set to the testing set. However, certain feature values in the testing set can still be outside the range of $[0, 1]$, which we set to zero or one as appropriate. This procedure ensures that all features are given equal importance during the training.

Binary Classification: We use support vector machine (SVM) as our main classification algorithm due to its robustness and efficiency (comparison to other algorithms is shown in §4.2). SVM classifies an unlabeled PIP with feature vector $\mathbf{x}$ based on its distance to the decision hyperplane with norm vector $\mathbf{w}$ and constant b:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{\forall i} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) \quad (1)$$

where the sum is over each PIP i in the training set with label $y_i \in \{-1, 1\}$, feature vector $\mathbf{x}_i$, and coefficient α_i that indicates whether the PIP is a support vector ($\alpha_i > 0$) or not ($\alpha_i = 0$). The feature vectors $\mathbf{x}_i$ are mapped into a higher dimensional feature space by a non-linear kernel function $k(\mathbf{x}_i, \mathbf{x})$.

SVM tries to identify $\mathbf{w}$ and b, determining the optimal decision hyperplane in the feature space such that PIPs on one side of the plane are good, and the other side bad. It is a maximum margin approach as it tries to maximize the distance between training data and the decision hyperplane. Therefore, the resulting boundaries are very robust to noisy data, which are inevitable in our case because some PIPs can have biased features due to the lack of enough user requests. In our study, we experimented with linear, radial basis function, and polynomial kernel functions.

Probability Estimation: We convert the classifier decision value to a score of how likely a PIP address is labeled as good, i.e., a posterior class probability $P(y = \text{good}|\mathbf{x})$. Bayes' rule suggests using a parametric form of a sigmoid function to approximate the posterior [17]:

$$P(y = \text{good}|\mathbf{x}) \approx \frac{1}{1 + \exp(Af(\mathbf{x}) + B)}. \quad (2)$$

The parameters A and B are fit by using maximal likelihood estimation on the training set. However, it has been observed in [20] that the use of the same training set may overfit the model (Equation 2), resulting in a biased estimation. To solve this problem, we take a computationally efficient way to conduct a five-fold cross-validation before the maximal likelihood estimation.

We label a PIP as good (or bad) only if the score is larger than a threshold p (or smaller than $1 - p$). The remaining PIPs are classified as mixed, corresponding to good PIPs being abused.

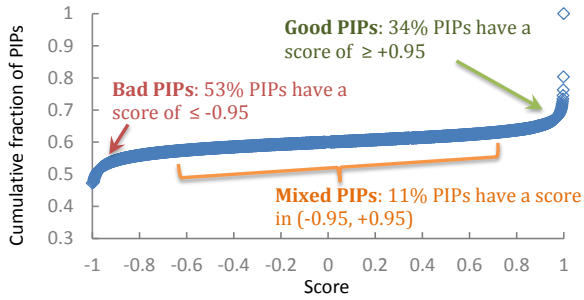

Figure 4: ***PIP score distribution.***

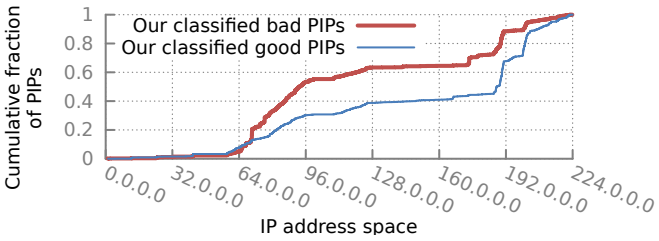

Figure 5: ***PIP address distribution.***

Parameter Selection: We check the parameters to assure good generalization performance and to prevent overfitting. We perform a grid search on kernel parameters using cross-validation. Specifically, good parameters are automatically identified by searching on the grid points, which are generally chosen on a logarithmic scale.

We implement PIPMiner on top of DryadLINQ [30], a distributed programming model for large-scale computing. We leave the implementation details to Appendix A.

4. PIP EVALUATION

This section presents our evaluation results of PIPMiner. We first describe the basic properties of the derived PIP list (§4.1). Using a subset of pre-labeled PIPs as the training data, we show that PIPMiner is highly accurate and efficient, and is generally applicable to various online services (§4.2). For PIPs that do not initially have labels, we validate that the majority of them have been correctly classified (§4.3).

4.1 PIP Addresses and Their Properties

We apply PIPMiner to a month-long Hotmail login log pertaining to August 2010 and identify 1.7 million PIP addresses. Although the PIP addresses constitute only around 0.5% of the observed IP addresses, they are the source of more than 20.1% of the total requests and are associated with 13.7% of the total accounts in our dataset. This finding suggests that the traffic volume from a PIP is indeed much higher than that from a normal IP address.

We classify a PIP as good (or bad) if the score given by the classifier is larger than 0.95 (or smaller than -0.95), which is evident in Figure 4. In our current process, around 34% and 53% PIPs are classified as good and bad, respectively. The remaining 13% PIPs are classified as mixed (abused).

Figure 5 plots the distribution of the PIP addresses across the IP address space. The distribution of the good PIP addresses differs significantly from that of the bad ones. For

Algorithm	Accuracy= $\frac{\#TPs+\#TNs}{\#All}$	Precision= $\frac{\#TPs}{\#TPs+\#FPs}$	Recall= $\frac{\#TPs}{\#TPs+\#FNs}$
SVM Linear Kernel	96.72%	97.31%	97.61%
SVM Polynomial Kernel	**99.59%**	**99.66%**	**99.83%**
Bagged Decision Trees	96.27%	96.64%	97.61%
LDA Linear	94.65%	96.88%	94.75%
LDA Diagonal Linear	89.54%	96.62%	86.80%
LDA Quadratic Linear	94.77%	95.67%	96.25%
LDA Diagonal Quadratic	92.50%	95.67%	92.57%
LDA Mahalanobis	94.46%	99.33%	91.97%
Naive Bayes (Gaussian)	92.60%	94.78%	93.73%
AdaBoost	92.56%	92.79%	96.09%
J48 Decision Tree (C45)	92.20%	93.84%	94.11%

Table 4: ***Accuracy of classification algorithms for initially labeled cases.***

bad PIP addresses, the majority of the IP addresses originate from two regions of the address space (64.0-96.255 and 186.0-202.255). Somewhat surprisingly, the distribution of the bad PIP addresses is similar to that of the dynamic IP addresses reported by Dynablock [28]. This observation suggests that attackers are increasingly mounting their attacks from dynamically assigned IP addresses, which likely correspond to end-user machines that have been compromised. To validate this hypothesis, we perform rDNS lookups to check if the domain name of a PIP address contains keywords that suggest the corresponding IP address as dynamic, e.g., dhcp, dsl, and dyn; we observe that 51.3% of the bad PIPs have at least one of these keywords.

4.2 Accuracy Evaluation of Labeled Cases

Among 1.7 million PIP addresses, $973K$ of them can be labeled based on the account reputation data. We train our classifier using the data sampled from these labeled PIP addresses. For the training data, we use the primitive good and bad PIP ratio, around 1:0.17. Note that, this ratio is not critical: we tried a few other very different ratios such as 1:1 and 1:6, and they give us similar results.

Classification Accuracy: Using 50,000 samples with 5-fold cross validation, we compare the accuracy of our classification in Table 4, using different learning algorithms based on the labeled PIPs. In general, we observe that all the classification algorithms achieve accuracies >89%. In particular, the SVM classifier with a polynomial kernel provides the best overall performance. We observed that the non-linear kernel of SVM outperforms the linear kernel, but at a cost of higher training time. Other classification algorithms such as linear discriminant analysis (LDA), naive Bayes, AdaBoost, and J48 decision trees all have very competitive performance. All these algorithms can serve as a building block of the PIPMiner system. Since SVM with a polynomial kernel provides the best result, we choose it in our implementation of the system.

Accuracy of Individual Components: Our classification framework relies on a broad set of features to achieve a high detection accuracy. To understand the importance of the features, we train a classifier on each single feature and Table 5 shows the performance breakdown of different types of features used for the training. Leveraging the account request distribution provides the best accuracy among the individual features, while the combination of more features does help reduce the classification error significantly.

Feature	w/ Block-level		w/o Block-level	
	Accuracy	Precision	Accuracy	Precision
Population Features	97.8%	98.2%	96.3%	96.8%
Population Size	92.6%	92.6%	90.5%	90.5%
Request Size	82.9%	82.4%	79.8%	77.8%
Requests per Account	89.2%	89.1%	89.2%	88.9%
New Account Rate	86.3%	86.7%	85.5%	85.3%
Account Request Dist.	94.2%	94.0%	93.6%	93.4%
Account Stickiness Dist.	88.1%	88.3%	88.2%	88.2%
Time Series Features	96.1%	97.2%	95.5%	96.4%
On/off Period	86.1%	89.2%	84.0%	88.0%
Diurnal Pattern	85.8%	85.1%	85.5%	85.1%
Weekly Pattern	90.6%	88.8%	89.6%	87.8%
Account Binding Time	90.8%	90.2%	90.3%	90.3%
Inter Request Time	92.5%	92.3%	91.8%	91.1%
Predictability	90.8%	89.6%	86.8%	82.6%
All Features	99.6%	99.7%	98.3%	99.1%
Previous Features	93.7%	93.5%	92.7%	93.1%

Table 5: ***Classification accuracy when trained on one type of feature. Account request distribution, population size, and inter request time provide the best individual accuracy. The classifier achieves the best performance when combining all types of features. The last row presents the performance when trained using the previously proposed features (account/request volume, account stickiness and diurnal pattern).***

Comparison with Previous Proposed Features: In Table 5, we demonstrate the effectiveness of our features by providing a comparison of our results to the classification results obtained by using the previously proposed features for the training. The previous features include account/request volume, account stickiness, and diurnal pattern. We observe that our feature sets provide substantial performance gains over the previous features, improving the detection accuracy from 92.7% to 99.6%.

Accuracy against Data Length: To quickly react to malicious attacks, the classifier needs to accurately differentiate bad PIPs from good ones based on as little activity history as possible. Figure 6a shows the classification results by forcing the classifier to train and test on the most recent k requests per PIP. A small value of k may yield biased features, resulting in a lower classification accuracy. Nevertheless we observe that the classifier requires only $k = 12$ requests per PIP[1] to achieve a very high accuracy. This indicates that our system could effectively identify newly emerged attacks.

Comparison with History-based Approach: Given that our classification framework works very well even with access to limited/partial history information, one could ask: do we need machine learning techniques to distinguish good PIPs from bad ones? To answer this question, we evaluate a baseline algorithm that uses only the historical ground truth to predict the PIP labels. For each sampled PIP from the pool of the labeled PIPs, we construct a time series of the reputation in a 1-day precision. Then we use an exponential weighted moving average (EWMA) model for reputation prediction. In particular, the reputation is predicted from the weighted average of all the past data points in the time

[1]Although the time series features become less useful with the decrease of k (due to the incomplete view in the time domain), PIPMiner can still achieve fairly high accuracy by combining the population features with the block-level time series features.

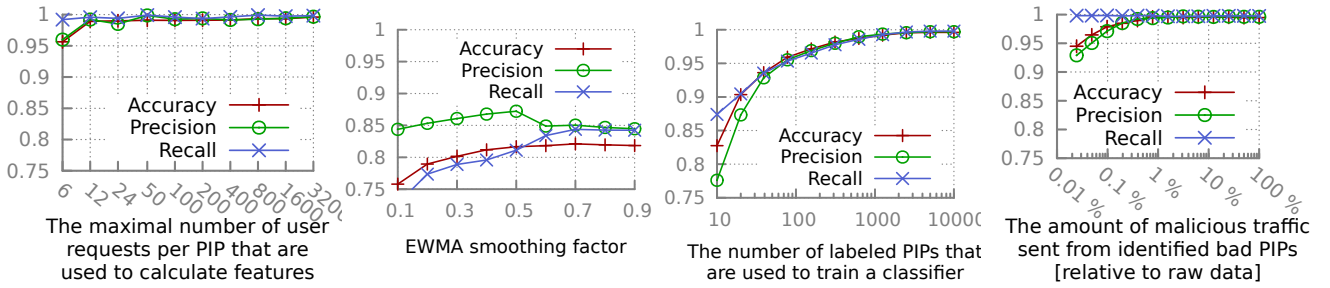

Figure 6: ***Classification Performance. (a) PIPMiner performance against data length, (b) Pure history-based approach performance, predicting PIP labels using all available prior reputation score records. (c) PIPMiner performance against the number of the PIPs for the training, (d) PIPMiner performance against the amount of malicious traffic. Accuracy=(tn+tp)/all; Precision=tp/(tp+fp); Recall=tp/(tp+fn).***

series. In this model, the weighting factors for each older data point decreases exponentially. Figure 6b shows that this baseline approach achieves only 82% accuracy with the best smoothing factor using all history information. Moreover, unlike our machine learning based approach, the baseline cannot be applied to the PIPs without labels. The machine-learning approach essentially learns the good/bad behavior patterns from many other PIPs, hence we do not require long activity histories.

Service Scale: PIPMiner is designed for analyzing PIP behaviors. One question is whether PIPMiner is useful for only very large service providers? Can we deploy PIPMiner for small-scale online services that have only a few PIPs? To answer these questions, we conduct an experiment to artificially remove part of the source IP addresses in the Hotmail trace randomly. As shown in Figure 6c, we demonstrate that PIPMiner still retains its high classification accuracy with only a few hundred labeled samples, which is fewer than 0.2% of the labeled PIPs in the trace. This suggests that PIPMiner is useful across a wider variety of business sizes.

Robustness to Attacks: With any detection scheme, adversaries will naturally try to invent methods to evade detection. The simplest choice would be to hide behind anonymized networks such as ToR [9]. However, such effort is not very effective because ToR's throughput is low and ToR IP addresses are public. Therefore, it is very easy for service providers to pay special attention to the requests coming from ToR IPs. Due to these constraints, we find that attackers have chosen to use open proxies or set up their own proxies to conduct malicious activities, rather than using ToR in our data. Such observation exactly motivates our work to classify PIPs.

Although the individual features of PIPMiner may be gamed, it is difficult for an attacker to evade the combinations of them. A malicious PIP that evades our classification (i.e., below the threshold of being classified as bad) may still be given a lower score by PIPMiner as long as it fails to emulate normal user behaviors for some features. The resulting score can be used in conjunction with other detection techniques to improve the detection confidence.

Moreover, PIPMiner raises the bar significantly for an attacker to evade the detection. The population and time series features will drive attackers to reduce their attack scale in order to approximate the behaviors from a real user population. To validate that PIPMiner can retain high accuracy when attackers reduce their attack scales, we synthetically remove traffic sent by classified malicious PIPs in both training and testing data. Figure 6d shows that PIPMiner is robust to such evasion strategies.

4.3 Validation of Unlabeled Cases

For the PIPs with known labels, we have shown that our classifier distinguishes good and bad ones accurately. It remains unclear whether we can accurately classify the *unlabeled PIPs*, which comprise 42.8% of the total PIPs. They are particularly hard to classify as many accounts on these PIPs are new with little history or few activities, so even the reputation data cannot help to generate their labels.

In the lack of ground truth for verifying the unlabeled PIPs, we combine information from both the *user registration month* and *future reputation* for validation.

User Registration Month: Intuitively, good PIPs are shared by many different normal users and hence exhibit a diverse distribution in the user registration months. For bad PIPs, as their requests are mostly generated by attacker-created malicious accounts, it is highly likely that most of these accounts were created in a short time window. To quantify whether our labels are correct, for each PIP we place the corresponding accounts into bins based on their registration month. Figure 7 shows the CDF of the percentage of accounts in the top bin, which has the largest number of accounts. Overall, we observe very different distributions between the classified good and bad PIPs. For the PIPs that we classify as good, only 4% of them have over 45% accounts registered in the same month. For the PIPs that we classify as bad, in contract, around 82% of them have at least 45% accounts registered in the same month. The distinct account behaviors suggest that the majority of the unlabeled PIPs are classified correctly.

We further validate the results using future reputation. Future reputation is a score reported by the reputation system months after our dataset's collection time. It gives the reputation system an opportunity to observe user activities for an extensive period of time, thus has a better chance of recognizing previously unclassified accounts, either good or bad. In particular, we use the reputation score of July 2011, which provides us with an updated user reputation

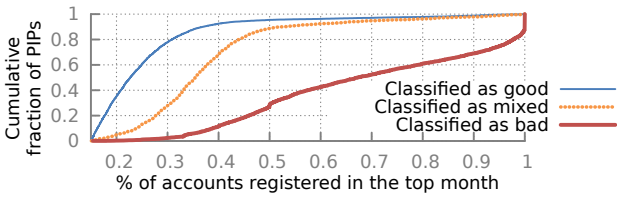

Figure 7: ***The cumulative fractions of the percentage of accounts that registered in the top*** 1 ***month. The function is plotted over initially unlabeled PIPs.***

Future reputation	Accounts on good PIPs	Accounts on bad PIPs
Good	61.6%	4.1%
Intermediate	7.4%	5.3%
Bad	7.5%	10.7%
Deleted	20.6%	75.4%
Unknown	2.9%	4.5%

Table 6: ***The future reputation (as of July,*** 2011***) of PIPs that do not have labels in August,*** 2010***.***

score almost one year after the testing dataset's creation time (August, 2010).

Table 6 summarizes the evaluation results based on future reputation. For the accounts on the initially unlabeled PIPs that we classified as good, around 62% of them are recognized as good after one year. For the accounts on the initially unlabeled PIPs that we classified as bad, around 86.1% of them are either deleted or marked as malicious after one year. We notice that a large portion of the accounts that were classified as bad had been deleted during the year and no longer exist in the updated reputation system. An account can be deleted if it displays malicious behaviors against user terms, or if there are no activities for a long period of time. The violation of terms includes spamming and phishing activities, reported by other mail servers.

These results demonstrate that PIPMiner can classify PIP addresses accurately. When combining information collected from multiple PIP addresses, we can recognize malicious accounts months before the existing system does, with a precision higher than 98.6% (§5.2).

5. APPLICATIONS

In this section, we investigate two applications of PIPs. First, we apply the PIP list to the Windows Live ID sign-up abuse problem to flag malicious accounts right on the day of their sign-ups (§5.1). Second, we apply the PIP list to the Hotmail malicious user sign-in problem and detect malicious accounts months before the Hotmail reputation system does (§5.2).

5.1 Windows Live ID Sign-up Abuse Problem

Our first application is to flag malicious (i.e., bot-generated) sign-ups using the PIP information. Since there is very little information about accounts available at the sign-up time, it is challenging to correctly distinguish all malicious accounts from good ones, given that they are all new. The flags that offered by PIP addresses could be used as a feature or a start-up reputation to detect malicious users more quickly and limit their damages.

Case	# Users [million]	% Good	% Intermediate	% Bad
Original	1.0	19.0%	39.9%	41.1%
New	2.8	1.4%	20.3%	**78.3**%

Table 7: ***Comparative study of our PIP list (first row) and the newly appeared "PIP-like" list (second row).***

PIP information		# Users [thousand]	% Good	% Intermediate	% Bad
Good	Abused	216.0	10.6%	35.3%	54.1%
	Normal	397.5	39.1%	34.0%	26.9%
Bad		410.5	3.9%	48.1%	48.0%

Table 8: ***User reputation distribution on different types of PIPs.***

We apply the 1.7 million PIP list derived from the Hotmail login log of August 2010 to the Windows Live ID sign-up trace in the same month. We study the sign-up behavior on two types of the PIP addresses. The first is the 1.7 million derived PIPs. The second is the set of IP addresses that have more than 20 sign-ups from the Windows Live ID system, but they are not included in the 1.7 million PIPs. We refer to the second set as the *newly appeared "PIP-like"* addresses.

As there is little information available at the sign-up time to distinguish good sign-ups from bad ones, we focus on the sign-ups related to Hotmail[2] and use the Hotmail reputation trace in July, 2011 (after 11 months) to determine whether a particular sign-up account was malicious or not.

Sign-ups from newly appeared PIP-like addresses: Table 7 shows a comparative study of our derived PIP list and the newly appeared PIP-like addresses. Although both lists are associated with a large number of malicious sign-ups, the fraction of bad sign-ups from PIP-like addresses is significantly higher (78.3% versus 41.1%) than that from our derived PIP list. Only 1.4% of the sign-ups establish good reputation scores after one year. This shows that good PIPs usually have activities across multiple services, as they are used by many normal users who have diverse interests. The PIP-like addresses that are dominantly used by just one service are more suspicious. They are more likely to be attacker-controlled IPs and leveraged by attackers to conduct attacks to a specific service. It is hard and also expensive for attackers to simulate normal user requests to all possible Web services. Hence, correlating activities across multiple services could yield a better PIP recognition accuracy and also attack-detection ability.

Sign-ups from our PIP addresses: Blindly applying our PIP list to the sign-up data, we observe a mixed user reputation (19.0% good, 41.1% bad, remaining as intermediate as shown in Table 7). To further classify the good sign-ups from bad ones, we leverage PIP address properties provided by our system: the goodness score and the number of good users.

The results are summarized in Table 8. For the PIPs with low goodness scores, not surprisingly, only 3.9% of the sign-ups remain good after 11 months. For the PIPs with high

[2]Windows Live ID is a unified ID, supporting multiple services such as Hotmail, Window Live Messenger, Xbox.

Actual (after 11 months)	Our prediction Good	Bad
Good	$155K$	$78K$
Bad	$106K$	$2506K$

Table 9: ***Using PIPs to predict user reputation right at the day of the sign-ups.***

Case	# Users in 1,000	% Good	% Intermediate	% Bad	% Delected	% Unknown
A ∩ B	551.9	0.6%	1.0%	32.1%	66.2%	0.1%
A ∪ B	1,154.2	0.3%	0.5%	21.6%	77.5%	0.1%

Table 10: ***The user reputation in July*** 2011***, for users who had positive reputation back in August,*** 2010***. We consider two criteria: (A) user name is within the top*** 3 ***naming patterns; (B) user registration date is in the top*** 1 ***registration month. First row: Both criteria are true. Second row: Either criteria is true.***

scores in the Hotmail login log, however, they could still be temporarily abused by attackers in the sign-up log. Here, we detect abuse scenarios by looking at the sign-up rate to the sign-in rate. If the ratio is larger than a threshold (set to 0.1 in the current experiment as the sign-up rate is usually a magnitude smaller than the sign-in rate from practical experience), we think that the IP address is potentially abused by attackers.

For the PIPs with high goodness scores, around 35% of them are potentially abused. Sign-ups from abused PIPs have much lower reputation scores compared to non-abused good PIPs. Among the $216K$ sign-ups from abused PIPs, we observe that only 10.6% sign-ups have good reputation scores. In contrast, almost 40% of the sign-ups from non-abused PIPs have good reputation scores and 34.0% are intermediate cases. Although we still observe 26.9% of the sign-ups from good PIPs display malicious behavior later, these could be low rate malicious sign-ups that escape the abuse detection system. As it is essentially hard to completely stop attackers from signing up malicious accounts for spamming, the ability to push attackers into a low rate mode and limit the number of their malicious accounts is of great value to the service provider.

Using the goodness scores of PIP addresses and the abuse detection method, we introduce the following sign-up early-warning system that flags new users right on the day of the sign-ups. We label users related to the "normal" PIPs as potentially good and label users related to the other cases as potentially bad. Using this method, we successfully detect more than 3 million bad sign-ups as shown in Table 9. We have a precision of 97.0% in detecting malicious accounts by misclassifying a small number of good users (i.e., false positives). We also have a high recall of 96.0% by accepting a moderate number of false negatives. The prediction of good users is less accurate, as 40.6% of them turn out to be malicious. While our techniques does not achieve a perfect accuracy, they provide early signals on the day of sign-ups and can be used in conjunction with previously proposed techniques (e.g., the Hotmail user reputation system, the anomaly detection methods) to speed up or improve confidence in malicious user detection. We leave the comparison to Quova's proxy list to Appendix B.

5.2 Hotmail Malicious Account Detection

In this section, we apply our PIP list to another application — the Hotmail malicious user sign-in problem. We use bad PIPs to help perform postmortem forensic analysis to identify malicious accounts that slipped through the Hotmail user reputation system.

Intuitively, there should not be many good users (with high reputation scores) using bad PIPs. However, this assumption might not be true because a normal user's computer can be hacked and turned into a bad PIP so this normal user's activities are mixed with bad users. To increase confidence, we correlated events from different bad PIPs together and identify users that login from multiple bad PIPs. To further distinguish bad users from good ones, we conservatively use the following two criteria:

Top Registration Month: This is similar to the validation method we use in Section §4.3. For each PIP, we place accounts into bins according to their registration month. As a bad PIP is dominantly used by malicious users, an account has a high chance of being malicious if it falls into the top bin that has the most number of accounts.

Top 3 Naming Patterns: The Hotmail group extracts the naming pattern (how the user name is structured, e.g., words plus 2 digits) of each user account, and we place accounts into into bins according to their naming patterns. Accounts that fall into the top 3 bins on a bad PIP are also very suspicious.

For users with good reputation scores as of August, 2010, but login from at least 2 bad PIPs in that month, we examine the above two criteria. We consider two cases here: **(i)** Both criteria are true, and **(ii)** either criterion is true. Similar to the previous sections, we check the reputation scores according to the reputation system in July 2011. Table 10 shows that only less than 0.6% of the users remain good 11 months later for both cases. In the "both true" case, we observed that more than 66% of the accounts are deleted, and more than 32% of the accounts are eventually classified as bad. This suggests that our PIP list can effectively detect malicious users months before the Hotmail reputation system does.

6. CONCLUSION

The ability to distinguish bad or abused populated IP addresses from good ones is critical to online service providers. As a first step towards this direction, we propose PIPMiner, a fully automated method to classify PIPs with a high accuracy. We demonstrate that the labeled PIP list can help identify attacks months before other detection methods. Although PIP lists are derived from service logs and thus may be application-specific, we show that they can be applied across datasets to detect new attacks. An interesting future direction is to study the overlaps and correlations among PIPs derived from different services, and to merge these PIPs for detecting attacks that would be hard to identify by a single service provider alone.

Acknowledgements

We thank Martin Abadi and Qifa Ke for their valuable advice and thank Mihai Budiu and Jon Currey their help on DryadLINQ. We are also grateful to Linda McColm, Keiji Oenoki, Krish Vitaldevara, Vasanth Vemula from the Hotmail team, and Jeff Carnahan, Harry Katz, Ivan Osipkov, Robert Sim from the Windows Live Safety Platform team for providing us with data and valuable comments.

References

[1] GML AdaBoost Matlab Toolbox. http://goo.gl/vh0R9.
[2] Networks enterprise data acquisition and IP rotation services. http://x5.net.
[3] Quova. http://www.quova.com/.
[4] ToR network status. http://torstatus.blutmagie.de/.
[5] J. D. Brutlag. Aberrant behavior detection in time series for network monitoring. In *USENIX Conference on System Administration*, 2000.
[6] X. Cai and J. Heidemann. Understanding block-level address usage in the visible Internet. In *SIGCOMM*, 2010.
[7] M. Casado and M. J. Freedman. Peering through the shroud: The effect of edge opacity on IP-based client identification. In *NSDI*, 2007.
[8] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011.
[9] R. Dingledine, N. Mathewson, and P. Syverson. ToR: The second-generation onion router. In *USENIX Security Symposium*, 2004.
[10] H. Eidnes, G. de Groot, and P. Vixie. Classless IN-ADDR.ARPA delegation. RFC 2317, 1998.
[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 2008.
[12] S. Hao, N. A. Syed, N. Feamster, A. G. Gray, and S. Krasser. Detecting spammers with SNARE: Spatio-temporal network-level automatic reputation engine. In *USENIX Security Symposium*, 2009.
[13] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed data-parallel programs from sequential building blocks. In *EuroSys*, 2007.
[14] J. P. John, F. Yu, Y. Xie, M. Abadi, and A. Krishnamurthy. Searching the searchers with searchaudit. In *USENIX Security*, 2010.
[15] J. Jung and E. Sit. An empirical study of spam traffic and the use of DNS black lists. In *IMC*, 2004.
[16] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe. Towards IP geolocation using delay and topology measurements. In *IMC*, 2006.
[17] H.-T. Lin, C.-J. Lin, and R. C. Weng. A note on Platt's probabilistic outputs for support vector machines. *Mach. Learn.*, 2007.
[18] A. Metwally and M. Paduano. Estimating the number of users behind IP addresses for combating abusive traffic. In *KDD*, 2011.
[19] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov. BotGrep: detecting P2P botnets using structured graph analysis. In *USENIX Security Symposium*, 2010.
[20] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 1999.
[21] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.
[22] A. Ramachandran, N. Feamster, and D. Dagon. Revealing botnet membership using DNSBL counter-intelligence. In *Usenix Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2006.
[23] A. Ramachandran, N. Feamster, and S. Vempala. Filtering spam with behavioral blacklisting. In *CCS*, 2007.
[24] G. Stringhini, T. Holz, B. Stone-Gross, C. Kruegel, and G. Vigna. Botmagnifier: Locating spambots on the Internet. In *USENIX Security Symposium*, 2011.
[25] L. Wang, K. S. Park, R. Pang, V. Pai, and L. Peterson. Reliability and security in the codeen content distribution network. In *USENIX ATC*, 2004.
[26] Y. Xie, V. Sekar, D. A. Maltz, M. K. Reiter, and H. Zhang. Worm origin identification using random moonwalks. In *IEEE Symposium on Security and Privacy*, 2005.
[27] Y. Xie, F. Yu, and M. Abadi. De-anonymizing the internet using unreliable ids. In *SIGCOMM*, 2009.
[28] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber. How dynamic are IP addresses? In *SIGCOMM*, 2007.
[29] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. Spamming botnets: Signatures and characteristics. In *SIGCOMM*, 2008.
[30] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, and J. Currey. DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language. In *OSDI*, 2008.
[31] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum. BotGraph: Large scale spamming botnet detection. In *NSDI*, 2009.

APPENDIX

A. IMPLEMENTATION

Our implementation consists of two stages. In the first stage, we parse the data sets and extract PIPs with their features and labels. We implement this stage in a parallel fashion by harnessing a cluster of 240 machines. The second stage is the training and testing stage. Fortunately, the input to this stage is small enough (e.g., less than 200 MBs for 1.7 million PIP addresses), allowing us to implement it on a single machine.

Data Parsing and Feature Extraction: The first stage is implemented on top of Dryad, an execution engine allowing distributed data-parallel computation [13]. Our primary input data is a large collection of user login records. The records are hash partitioned to a set of processing machines according to the user login IP address, and then each machine independently computes per-IP account counts and per-IP request counts to derive PIP list and blocks. We efficiently merge the remaining data sets such as the user

account data by performing distributed join operations. We then partition the records that are associated with a PIP address based on its block ID, ensuring that the records in the same PIP blocks are processed by the same computing node. This is a critical phase to minimize the system overhead, allowing computing nodes to derive PIP features independently.

We further optimize the system performance using three methods. First, we configure DryadLINQ to use dynamic partitioning range keys, which are determined at run time by sampling the input datasets. This helps balance the data load on each machine, especially for our second partition phase where the input key distribution is hard to estimate in advance. Second, we minimize the cross-node communication cost by compressing the intermediate results. This reduces the communication overhead by ∼68%. Finally, some PIP blocks consist of a very large number of IP addresses (e.g., mobile gateways), each of which can potentially be associated with a disproportionate amount of records. This can lead to an increased computation load on certain computing nodes, making them the bottleneck of the operations, especially for large data sets. To sidestep this, we randomly sample requests of *only* the top 20 PIPs, which originally carry the largest number of user requests. The per-PIP sampling rate is automatically chosen such that the top 20 PIPs will finally have a similar number of user requests of the 21st largest PIP. This scheme does not significantly bias the classification results because PIPMiner requires only a small number of requests per PIP to achieve high classification accuracy. This scheme, however, can reduce the overall computation time for feature extraction by $35\% - 55\%$.

Training and Testing: For training and testing, we ran our experiments on a single machine with Quad Core CPU and 8 GB memory. We use the LIBSVM [8] and LIBLINEAR [11] toolkits in our implementation. To compare with other classification algorithms, we use GML AdaBoost [1] and Quinlan's C4.5 [21] decision tree. We also tested the Matlab built-in classification algorithms, including Naive Bayes, bagged decision trees and linear discriminant analysis (LDA), for performance comparison.

B. COMPARISON TO QUOVA PROXY LIST

We have shown that our PIP list can help flag malicious sign-ups right on the day of the sign-ups. This section attempts to answer the question: Does our labeled PIP list have wider applicability compared to commercial proxy lists? To answer this question, we extract proxy IP addresses in Quova's GeoPoint data using the IP Routing Type field and apply the Quova proxy list to the Windows Live ID sign-up abuse problem. We find that >99.9% of the Quova proxy IPs are not associated with any sign-up activities, and 99.8% of the PIPs that have good sign-ups are not included by Quova's list. For Quova proxy IPs that do have activities, to ensure fair comparison, we only look at those that are associated with greater than 20 sign-up requests per month. These proxies are categorized into different types by Quova, including mobile gateways, AOL proxies, regional proxies, and international proxies. For each type of Quova proxies, Figure 8 shows the fraction of good and bad sign-ups. Clearly, mobile gateways, AOL proxies and regional proxies

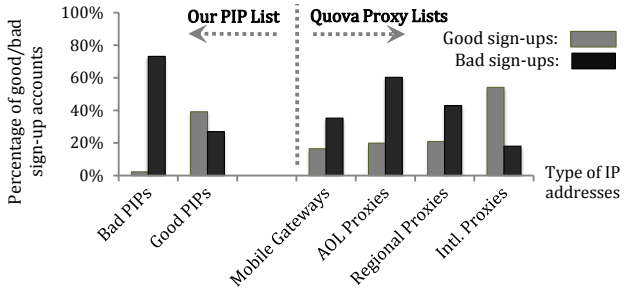

Figure 8: ***Given a certain type of IP addresses, the percentage of good and bad accounts. We use the Hotmail user reputation trace in July,*** 2011 ***to classify users into good, bad, intermediate, and unknown.***

Case	IP Address	Good Sign-ups
$\|P\|$: Intl. Proxies (Quova)	1,164	3,951
$\|G\|$: Our good PIPs	203,087	397,539
$\|P \cap G\|$: # Covered	939	3,467
$\|P \cap G\|/\|P\|$: % Coverage	80.6%	87.7%

Table 11: ***The number of good sign-ups that are originated from our good PIPs and from Quova's international proxies.***

all have mixed sign-up behaviors (20% of the sign-ups are good and 40% of the sign-ups are bad), letting through a lot of malicious sign-ups. Although Quova's international proxies are associated with a large percentage of good sign-ups, our good PIP list already covers most of them: 80.6% of the IPs and 87.7% of the good sign-ups as shown in Table 11.

C. TIME FORECASTING FEATURES

We apply forecasting models on data time series to find out abnormal periods, which might be the abuse periods of good PIPs. As we observed strong periodicities from many good PIPs, we adopt the additive Holt-Winters' seasonal forecasting model [5], which decouples a time series $T[t]$ into three factors: level $L[t]$, trend $B[t]$ and seasonal $S[t]$:

$$L[t] = \alpha(T[t] - S[t-\upsilon]) + (1-\alpha)(L[t-1] + B[t-1])$$
$$B[t] = \beta(L[t] - L[t-1]) + (1-\beta)(B[t-1])$$
$$S[t] = \gamma(T[t] - L[t]) + (1-\gamma)(S[t-\upsilon])$$

The season length is denoted by υ, and the update rate parameters are α, β, and γ. Then the forecast $F[t]$ simply adds up these three factors, i.e., $F[t] = L[t-1]+B[t-1]+S[t-\upsilon]$. To quantify whether the time series can be accurately predicted by the forecasting model, we call it an *anomaly* when the forecast error is relatively large (i.e., the difference and the ratio between the forecast value $F[t]$ and the actual value $T[t]$ is larger than some thresholds). A grid search is automatically performed to find the best parameters and thresholds that maximizing the cross-validation accuracy on the sampled training data. For the anomaly threshold, we search in different multiples of the standard deviation of the time series. Our features include the number and the density of the low-volume anomalies ($T[t] \ll F[t]$) and high-volume anomalies ($T[t] \gg F[t]$).