(Un)wisdom of Crowds: Accurately Spotting Malicious IP Clusters Using Not-So-Accurate IP Blacklists

Baris Coskun

***Abstract*—Most complex tasks on the Internet—*both malicious and benign*—are collectively carried out by clusters of IP addresses. We demonstrate that it is often possible to discover such clusters by processing datasets and logs collected at various vantage points in the network. Obviously, not all clusters discovered in this way are malicious. Nevertheless, we show that malicious clusters can accurately be distinguished from benign ones by simply using an IP blacklist and without requiring any complex analysis to verify malicious behavior. In this paper, we first propose a novel clustering framework which can be applied on datasets of network interactions to identify IP clusters carrying out a specific task collectively. Then, given such a list of identified clusters of IP addresses, we present a simple procedure to spot the malicious ones using an IP blacklist. We show that by choosing the parameter of the proposed clustering process optimally using a blacklist, hence making it blacklist-aware, we significantly improve our overall ability to detect malicious clusters. Furthermore, we mathematically show that even a blacklist with poor accuracy can be used to detect malicious clusters with high precision and recall. Finally, we demonstrate the efficacy of the proposed scheme using real-world login events captured at the login servers of a large webmail provider with hundreds of millions of active users.**

I. Introduction

Many malicious activities on the Internet have evolved into quite complex operations which are carried out by groups of IP addresses. Email spam, distributed password guessing attacks and malware distribution networks are few examples of this sort. Such collective effort from a group of IP addresses often leaves quite noticeable traces at various vantage points in the network. This enables defenders to cluster the IP addresses of such groups together using simple data analysis techniques. For example, attackers often use a group of IP addresses to log into compromised web accounts to carry out various malicious tasks [1], [2]. Therefore, a clustering analysis of the login events, based on common accounts accessed from different IP addresses, allows an online service provider to link these IP addresses together, since they are used to log in to the same set of web accounts.

Clearly not all clusters of IP addresses produced by such data analysis schemes are malicious. For example, the same clustering analysis of login events mentioned above often groups together gateway or proxy IP addresses of various kinds of networks, such as a mobile network or a corporate network, since they are used by same sets of users to log into their accounts. Obviously, such clusters of gateway IP addresses are not malicious at all. In fact, there are numerous other legitimate cases that could result in benign IP clusters, such as content distribution networks, peer-to-peer networks and cloud computing services. As a result, most IP clusters discovered via such data analysis techniques are actually benign. To spot malicious clusters, further analysis to verify malicious behaviors and traits is required. However, such behavioral analysis is often quite complex and requires intense domain knowledge.

In this work we argue that, one can easily spot malicious clusters in a statistically sound way using an IP blacklist alone, without requiring any complex behavioral analyses. Surprisingly, even a mediocre IP blacklist with not-so-great accuracy can potentially do the trick. The intuition is that, even if a blacklist frequently makes mistakes on individual IP addresses, it is highly unlikely that it makes mistakes on all IP addresses of a given cluster. Therefore, the expected number of blacklisted IP addresses in a malicious cluster is statistically distinguishable from that in benign ones.

Based on this intuition, we propose a novel and simple scheme to identify malicious clusters of IP addresses in a given dataset of network interactions, such as web account access logs or network flow records. In summary, we first extract clusters of IP addresses from the dataset using a *blacklist-aware* clustering method, which maximizes the statistical dependence between resulting malicious clusters and a given IP blacklist. We achieve this by representing the similarities between IP addresses in the dataset as an IP-IP graph and then applying a simple graph clustering scheme where we optimally pick its parameter using the blacklist. Then, once we obtain a list of IP clusters, we deem the clusters which exhibit high dependence on the blacklist to be malicious. We show mathematically and empirically that incorporating an IP blacklist into both clustering and identification phases allows us to detect malicious clusters of IP addresses with high precision and recall, as long as the clusters are large enough and the blacklist performs slightly better than random guessing (i.e. the true positive rate is greater than its false positive rate). We make the following contributions:

- We propose a novel IP clustering framework for a given dataset of network interactions, which maximizes the statistical *"dependence"* between resulting malicious clusters and a given IP blacklist.

Baris Coskun is with Amazon Web Services, New York, NY, USA. e-mail: barisco@amazon.com. This work was done when the author was with Yahoo! Research.

Malicious Activity	Dataset Observed In	Similarity Measure Between IPs
Utilizing compromised web accounts	Login Events observed at a web service provider	Number of common accounts logged in
Delivering email spam	Emails received at an email server	Content-based similarity between received emails
Botnet command and control	Network flow records captured at edge routers of a network	Number of common IPs or domains communicated with (excluding popular ones such as google, yahoo, etc.)

TABLE I. EXAMPLE CASES OF MALICIOUS IP CLUSTERS ALONG WITH THE DATASET THAT THEY CAN BE OBSERVED IN AND POTENTIAL SIMILARITY MEASURE THAT CAN BE USED IN CLUSTERING.

- We mathematically show how an IP blacklist, even a mediocre one, can be used to accurately determine whether a given cluster of IP addresses is malicious or not.
- We demonstrate the efficacy of the proposed scheme using real-world dataset collected at the login servers of a large webmail provider with hundreds of millions of active users. We show that the proposed scheme is able to identify malicious clusters of IP addresses utilizing compromised user accounts significantly more accurately than the baseline detection scheme.

The rest of the paper is organized as follows. In Sections II, we give a few real-world examples of IP clusters and propose a novel blacklist-aware clustering scheme. In Section III we present how a blacklist can be utilized to identify malicious IP clusters. In Section IV, we demonstrate how the proposed clustering and detecting scheme works in practice using real-world datasets. In Section V we discuss limitations of the proposed scheme. We present related work and conclusions in Sections VI and VII, respectively.

II. Clustering IP Addresses

Complex malicious activities on the Internet are carried out by groups of IP addresses due to several reasons. For example, the scale of most malicious activities, such as spamming or denial of service attacks, can be greatly improved by splitting the load over many hosts. Or adding redundancy to various components of a malicious activity (i.e. botnet command-and-control servers) can prevent single point of failures. Perhaps more importantly, utilizing many different hosts as the attack source, a malicious activity has a better chance of evading detection. Groups of IP addresses utilized by a malicious activity may belong to compromised hosts (i.e. botnets) or dedicated malicious servers [3]. They are often from all around the world. Nevertheless, they usually exhibit common traits specific to the group they are in, since they collectively carry out a common task. Such similarities allow organizations at the receiving end of a malicious activity to cluster these IP addresses together. In this section, we first present three example cases, which are also summarized in Table I, to discuss how IP addresses utilized by a malicious activity leave traces at various vantage points in the network and hence can be clustered together. After that, we present a novel clustering scheme which can be used under various scenarios.

A. IP Clusters in the Wild

1) Utilizing Malicious Web Accounts: Many web services, such as online social networks and web email providers, are plagued by malicious accounts, which are often used to send spam emails, post spam comments on blogs/forums, post fake product reviews, distribute political views, etc. [1], [2], [4], [5], [6]. These accounts can initially be legitimate accounts that are later taken over by attackers, or can be accounts specifically created by attackers for malicious purposes (i.e. sybil accounts). Regardless of their origins, attackers often rely on limited a pool of compromised hosts to login these accounts to carry out malicious acts [7]. This leads to multiple compromised hosts accessing same sets of accounts within a certain time window [1], [2]. Therefore, the IP addresses of such compromised hosts often manifest themselves as tightly connected clusters when a web service provider performs a clustering analysis on its access logs by linking together IP addresses which log in same sets of users.

2) Email Spam Campaigns: In a typical email spam campaign, spammers attempt to send vast number of email messages to vast number of recipients. To accomplish this, they often utilize several compromised hosts as SMTP (Simple Mail Transfer Protocol) servers. By originating spam from many different SMTP servers, spammers can scale up the number of mails that they send. In addition, incorporating many different sources in a spam campaign also helps spammers evade detection [8].

A group of IP addresses serving for a specific spam campaign often send emails which tend look similar to each other. This can be observed even for template-based spam campaigns, where each spam email is dynamically generated and slightly different than others, since spam templates usually have invariant and limited-varying portions in order to generate semantically sensible content [9][10]. As a result, by measuring the similarity between email contents sent by different IP addresses, email service providers targeted by a spam campaign can cluster together the IP addresses utilized by the campaign.

3) Botnet Command-and-Control Traffic: Substantial amount of malicious activities on the Internet, such as spam, denial of service attacks, brute force attacks, etc., are carried out by networks of compromised hosts called botnets. To successfully accomplish these attacks, members of a botnet (i.e. bots) coordinate with each other by regularly receiving command and control messages from their botmasters. There are mainly two architectures (i.e. centralized and peer-to-peer) adopted by botmasters to efficiently distribute these messages among bots [11]. In a centralized botnet, member bots reach out a central resource, such as a specific IRC channel or an IP address or a domain name, etc., to receive command and control messages. In a peer-to-peer botnet, on the other hand, member bots form a peer-to-peer overlay network to distribute command and control messages among themselves.

From the perspective of edge routers monitoring the traffic of a large network which unwittingly harbors many bots (i.e. a large organization or an Internet Service Provider), clustering the bots of a centralized botnet together is relatively straightforward. Basically, all IP addresses communicating with a specific central resource can be put in a same cluster [12]. This task becomes slightly harder for peer-to-peer botnets since there is no central resource. However, it is still possible to link together the members of a peer-to-peer botnet using a similarity measure defined as the number of common domain names (or IP addresses) that the two IP addresses have accessed within a specific time window—excluding popular domain names such as "google.com" or "yahoo.com" which are accessed by almost all IP addresses [13] [14] [15].

B. Blacklist-Aware Clustering

Above examples present various scenarios where IP addresses, which collectively carry out a common task, can be linked together by simple data analysis methods. Although the specifics varies significantly, the common underlying structure in all those examples is that, one can define a similarity measure between IP addresses to identify clusters. For example, in the case of malicious web accounts, the similarity measure can be defined as the number of common accounts that the two IP addresses have been used to log in within a specific time window [1]. Other example similarity measures are presented in Table I.

Once such an appropriate similarity measure is defined between pair of IP addresses, a given dataset can be represented as an undirected graph, where vertices represent IP addresses and the edge-weight between two vertices represents the similarity measure value between corresponding IP address.

We expect to see two kinds of edges on such a graph. First we expect to see edges which exist due to collective and coordinated behavior of clusters of IP addresses. We expect these edges, what we call *cluster edges*, to tightly connect the IP addresses of a clusters to each other on the graph. Second, we also expect to see considerable number of noise edges, which exist due to various arbitrary reasons and random events other than coordinated activities of IP clusters. Such noise edges, which usually comprise the majority of the edges on a typical graph in our experiments, may connect isolated and independent IP addresses to existing clusters or even connect multiple clusters together on the graph. Nevertheless, we expect the weights of the cluster edges to be substantially larger than noise edges, since cluster edges represent genuine relationships as opposed to arbitrary noise.

After building such a generic graph, extracting clusters of IP addresses essentially corresponds to distinguishing cluster edges from noise edges. To address this we propose a very simple clustering scheme as follows:

i) Remove all "**weak edges**" from the graph, where a "weak edge" is defined as an edge whose weight is less than a threshold.
ii) Output the connected components of the resulting graph as clusters.

Obviously, the crucial issue here is how to pick a proper threshold value which yields meaningful clusters. If we pick a too low threshold, then we would not be able to isolate malicious clusters and end up with very large clusters possibly comprising both malicious and benign IP addresses. On the other hand, if we pick a too high threshold, then we would run into risk of fragmenting malicious clusters into smaller ones, thereby making them much harder to detect. To address this, we propose to utilize a given IP blacklist to find the optimal threshold value. More specifically, we propose to pick the threshold value so that the "statistical dependence" between the blacklist and the malicious clusters produced by the clustering procedure is maximized. The intuition is that, the stronger the dependence between a cluster and the blacklist, the stronger the evidence that the cluster is malicious. Therefore by maximizing this statistical dependence, we would essentially enable the proposed clustering scheme to output malicious clusters with strongest statistical evidence of being indeed malicious. Below, we first present how to measure this dependence and then propose a simple procedure to pick the optimal threshold.

1) Measuring Statistical Dependence: Suppose we are given an IP blacklist 'B' and a dataset 'N' from which we extract clusters of IP addresses. To measure the dependence between the blacklist 'B' and an extracted IP cluster 'C' from the dataset 'N', we first define the following pair of events with binary outcomes for each IP address in the dataset:

i) $cl_i : i^{th}$ IP address of dataset N is in cluster C
ii) $bl_i : i^{th}$ IP address of dataset N is in blacklist B

Clearly, these events are independent if 'C' is benign. But for malicious clusters we expect these events to be measurably dependent. To measure the (in)dependence between these events, we compute *standardized residual*, which is the difference between observed and expected number of co-occurrences of these events divided by its standard error under null hypothesis (i.e. when the events are independent) [16]. Standardized residual essentially measures how much more frequently the IP addresses of cluster C are in blacklist B with respect to the case where C is benign. Therefore, higher standardized residual values indicate stronger evidence that the events are dependent. More specifically, standardized residual is defined as:

$$R = \frac{n - \hat{\mu}}{\sqrt{\hat{\mu}(1-p_1)(1-p_2)}} \tag{1}$$

where n is the observed number of times these events co-occur, $\hat{\mu}$ is the expected number of times these events co-occur if they were independent, and p_1 , p_2 are the probabilities of event 1 and event 2 respectively.

In our setting, null hypothesis represents cluster C is benign and the events cl_i and bl_i are independent. Therefore, the expected number of times that an IP address is both in cluster C and in blacklisted B under null hypothesis can be written as:

$$\hat{\mu} = |N| \frac{|C|}{|N|} \frac{|B|}{|N|} = \frac{|C|.|B|}{|N|}$$

where $|N|$ denotes and the number of IP addresses in the dataset, $|B|$ denotes the number of blacklisted IP addresses in the dataset and $|C|$ denotes the size of cluster C. Also we can write the probability of an IP address being in C is $p_1 = |C|/|N|$ and similarly, the probability of an IP address being in blacklist B under null hypothesis is $p_2 = |B|/|N|$. Plugging these into Equation (1), we can compute the standardized residual of a cluster C that has n IP addresses in blacklist B as:

$$R = \frac{n - \frac{|C|.|B|}{|N|}}{\sqrt{\frac{|C|.|B|}{|N|}(1 - \frac{|C|}{|N|})(1 - \frac{|B|}{|N|})}} \tag{2}$$

2) Finding the Optimal Threshold: Using this measure of statistical dependence, we now show how we can pick the optimal threshold for proposed clustering scheme. Ideally, we would like the clustering scheme to produce malicious IP clusters which have as high standardized residuals as possible. This would maximize the evidence that these clusters are indeed malicious. At the same time, we would not like the clustering scheme to produce many small and fragmented clusters since distinguishing between malicious and benign clusters is much more accurate for larger clusters, as we will discuss in the next section. These requirements can be satisfied by simply maximizing the average standardized residual over all clusters. This objective function can be written as:

$$\mathcal{O} = \frac{1}{k}\sum_{i=1}^{k} R_i \tag{3}$$

where k is the number of clusters and R_i is the standardized residual of i^{th} cluster as given in Equation (2). Notice that, standardized residual is expected to be zero for benign clusters, since $E[n] = \hat{\mu}$ under null hypothesis in Equation (1). Hence benign clusters do not contribute to the objective function at all in expectation sense. Therefore, by maximizing the average standardized residual, we actually optimize only over malicious clusters without explicitly identifying them first. Essentially, by maximizing $\mathcal{O}$, we force malicious clusters to have as high standardized residuals as possible. At the same time, by maximizing $\mathcal{O}$, we favor fewer clusters hence less fragmentation due to k in the denominator.

It may be very hard, to analytically find the optimal threshold which maximizes $\mathcal{O}$, since relationship between threshold and average standardized residual is far from trivial. While we could employ various numerical methods, such as gradient ascend, we decided to find the optimal threshold by exhaustive search over a range of possible threshold values. We believe that exhaustive search is the most practical solution in this case where we only deal with a single variable optimization problem over a limited range. The exhaustive search is quite straightforward and can be summarized as follows:

```
Data: IP-IP Graph ($\mathbf{G}$)
Data: List of candidate threshold values ($T$)
Result: $\hat{t}$: Optimal Threshold
$\hat{t} \leftarrow -\infty$;
$\hat{\mathcal{O}} \leftarrow -\infty$;
for $t \in T$ do
    $\mathbf{G}' \leftarrow$ remove edges with $weight < t$ from $\mathbf{G}$;
    Find connected components of $\mathbf{G}'$;
    compute $\mathcal{O}$ for connected components;
    if $\mathcal{O} > \hat{\mathcal{O}}$ then
        $\hat{\mathcal{O}} \leftarrow \mathcal{O}$;
        $\hat{t} \leftarrow t$
    end
end
```

Once we find the optimal threshold for a given dataset using the above procedure, we apply the proposed clustering scheme to obtain a list of clusters. As discussed previously, most of these clusters are not malicious at all. Next, we present how to identify the malicious ones within a list of IP clusters using the given IP blacklist.

III. Identifying Malicious Clusters

A. Not All Clusters Are Malicious

The optimal clustering scheme presented above outputs list of IP clusters which may or may not be malicious. Actually, significant number of resulting IP clusters are benign in general. For instance, consider the example of malicious web accounts discussed in previous section, where *"number of common users logged in"* is used as the similarity measure. In that case, a web service provider analyzing its access logs to identify IP clusters would observe that many users within a private network, such as a corporate network or a mobile network, access their accounts via gateway IP addresses (i.e. external facing IP addresses of Network Address Translation (NAT) devices) of the networks that they are in. Therefore, after applying the proposed clustering scheme, all gateway IP addresses of a private network would manifest tightly connected clusters. Clearly, these are not malicious clusters. In order to tell apart malicious clusters from benign ones, some additional information associated with maliciousness is required. One source of such information is to analyze behavior of identified clusters to check whether they exhibit any malicious traits [2]. However, this kind of analysis tend to be highly complex and domain specific. Instead, here we argue that IP blacklists are far simpler source of side information which allow us to accurately spot the malicious clusters.

B. Spotting Malicious Clusters Using a Blacklist

Recall from Section II-B that, given a blacklist B and a dataset N, the standardized residual (R) of a cluster C in dataset N indicates the strength of evidence that C is malicious. Using R, we essentially measure the amount of statistical dependency between two binary events, namely IP address being in cluster C and IP address being in blacklist B. Referring to Equation (1), interpretation of standardized

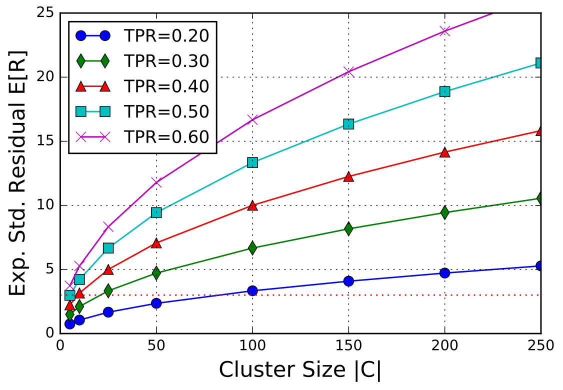


Fig. 1. Expected standardized residuals for different Cluster Sizes and True Positive Rates. The False Positive Rate of the blacklist is set to 10% and there are $N = 100,000$ IPs in the entire dataset.

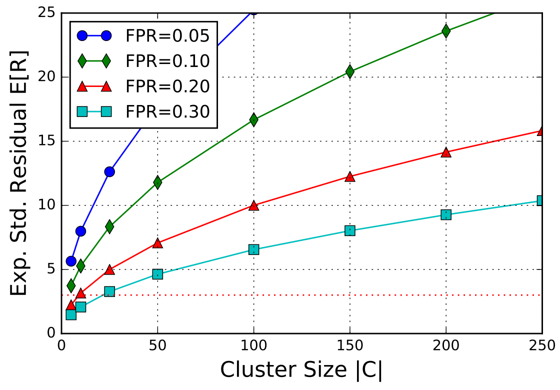


Fig. 2. Expected standardized residuals for different Cluster Sizes and False Positive Rates. The True Positive Rate of the blacklist is set to 60% and there are $N = 100,000$ IPs in the entire dataset.

residual is quite straightforward. Since R is standardized (i.e. normalized by standard error), $R = r$ means that the observed number of co-occurrences of these two events is r standard deviation away from its expected value under null hypothesis. Therefore, $R > 3$ is considered as a very strong evidence that the two events are correlated, since the probability of observing this simply by chance under null hypothesis is less than 0.3% [16]. As a result, to decide whether a given cluster of IP addresses is malicious or not, we compute the standardized residual using Equation (2) and declare the cluster to be malicious if standardized residual is greater than 3. Note that, we guarantee a low false positive rate with this approach, since the probability of a benign cluster having standardized residual greater than 3 is less than %0.3. This probability can be further reduced by setting a higher threshold, however, doing so naturally leads to fewer malicious clusters being detected.

C. Effect of Blacklist Quality

This simple procedure allows us to identify malicious clusters among all clusters produced by the optimal clustering scheme. To understand how the performance of blacklist B affects standardized residual and therefore our detection accuracy, we look at the expected value of the standardized residual for various blacklist performances. Note that the size of a cluster also affects detection accuracy and cluster size partly depends on the threshold used in optimal clustering scheme. However, recall that the threshold in the clustering process is automatically chosen to produce clusters with sizes as close as possible to their true sizes. While there may be some level of noise in this process, in order to investigate the effect of blacklist quality alone, in this part we assume that the optimal clustering scheme perfectly extracts true IP clusters from the data. To characterize the blacklist quality, we first define the true positive rate of the blacklist as:

$$TPR = Pr\left(\text{IP is in blacklist } B \;\middle|\; \text{IP is Malicious}\right)$$

and the false positive rate of the blacklist as:

$$FPR = Pr\left(\text{IP is in blacklist } B \;\middle|\; \text{IP is Benign}\right)$$

Here we implicitly assume that any arbitrary non-malicious IP address is equally likely to be a false positive. However, in practice a previously malicious but later cleaned IP address may be more likely to be a false positive than others due to poor blacklist maintenance. But for the sake of simplicity, we assume false positives are independent and identically distributed. Similarly, we also assume that true positives are i.i.d as well.

If cluster C is benign then blacklisted IP addresses in C are due to false positives and hence the expected value of standardized residual is $E[R] = 0$, regardless of TPR and FPR values. More formally, if C is benign then we have:

$$E[n] = \hat{\mu} = |C| \times FPR$$

and equivalently: $E[n - \hat{\mu}] = 0$

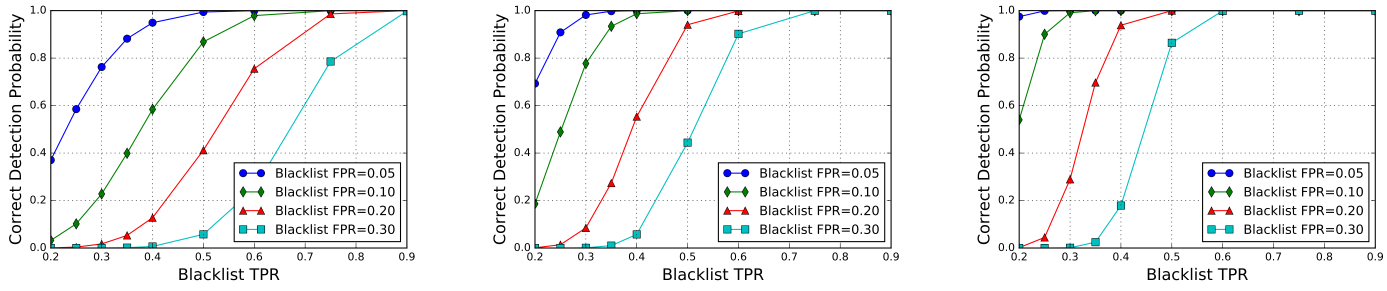
On the other hand, if cluster C is a malicious one, then the expected number of blacklisted IP addresses in C will be:

$$E[n] = |C| \times TPR$$

Since there are far more benign IP addresses than malicious ones in the dataset, the probability of an IP address in the dataset N being in blacklist B is roughly equal to the blacklist's false positive rate (i.e. $p_2 = FPR$). As a result, combining these with Equation (2), we can write the expected value of standardized residual as:

$$\mathbf{E}[R] = \frac{|C|(TPR - FPR)}{\sqrt{(|C|.FPR)(1 - \frac{|C|}{|N|})(1 - FPR)}} \tag{4}$$

Notice that, when the cluster is malicious, the expected standardized residual increases as the difference between blacklist's true positive rate and false positive rate increases. It also increases with cluster size as well. This allows us to make two fairly obvious observations:



(a) Cluster Size = 20 (b) Cluster Size = 50 (c) Cluster Size = 100

Fig. 3. Probability of correctly detecting a malicious cluster for different blacklist true and false positive rates and different cluster sizes. There are $N = 100,000$ IPs in the entire dataset. Since we consider a cluster to be malicious when standardized residual is greater than 3, falsely claiming a benign cluster to be malicious is less than 0.3% in all cases.

i) Malicious clusters can be identified more accurately if we utilize more accurate blacklist.
ii) Larger clusters can be identified more accurately than smaller ones.

Interestingly, however, it turns out the blacklist does not have to be very accurate in order to accurately identify malicious clusters. To demonstrate this, we plot expected standardized residual for different cluster sizes and blacklist true positive rates in Figure 1. We set the blacklist false positive rate to $FPR = 10\%$ and the size of the whole dataset to $|N| = 100,000$. As can be seen in Figure 1, the expected standardized residual quickly exceeds the critical value 3 even when blacklist true positive rate is 40%. So, a mediocre blacklist with 10% false positive rate and 40% true positive rate can be used to accurately identify most malicious clusters with size greater than 9. Furthermore, even a very poor performing blacklist with $TPR = 20\%$ and $FPR = 10\%$ can be used to accurately identify malicious clusters so long as their size greater than 80. Note that the aforementioned blacklist performances are comparable to many public blacklists which are widely used in practice [17][18].

On the other hand, to investigate the effect of false positive rate, we fix blacklist true positive rate to $TPR = 60\%$ and plot expected standardized residual for different false positive rates and cluster sizes in Figure 2. We observe that for low false positive rates, such as $FPR = 10\%$, the blacklist can be used to accurately identify malicious clusters with size as small as 5. For larger false positive rates, such as $FPR = 30\%$, malicious cluster size has to be greater than 20 in order to be reliable identified using the blacklist.

Notice that both Figure 1 and Figure 2 show the expected value of standardized residual and the arguments above are made in the expectation sense. The actual probability of detecting a malicious cluster using a blacklist is indeed equal to the probability of standardized residual being grater than 3. Using Equation (1) this probability can be simply written as:

$$Pr\left[R > 3\right] = Pr\left[n > \hat{\mu} + 3.\sqrt{\hat{\mu}(1-p_1)(1-p_2)}\right]$$

which, after combining with Equation (4), is equivalent to:

$$Pr\left[n > |C|.FPR + 3.\sqrt{(|C|.FPR)(1-\frac{|C|}{|N|})(1-FPR)}\right]$$

This probability can easily be computed, since n is the number of blacklisted IP addresses in cluster C and binomially distributed, such that:

$$Pr(n) = \binom{|C|}{n}(TPR)^n(1-TPR)^{|C|-n}$$

Using these equations, we plot probability of correctly detecting a malicious cluster (i.e. $Pr[R > 3]$) for various blacklist true and false positive rates and different cluster sizes in Figure 3. We clearly observe that, better blacklists yield more accurate results. We also observe that the detection accuracy improves rapidly with the cluster size. For instance, a mediocre blacklist with TPR=50% and FPR=10% is able to identify a malicious cluster of size 20 around 85% of the time. The same blacklist correctly identifies almost all of the malicious clusters when the cluster size grows to 50. Recall that, with standardized residual threshold set to 3, the probability of falsely claiming a benign cluster to be malicious is less than 0.3% in all cases.

In conclusion, we show that our simple procedure can accurately spot malicious clusters using even a mediocre blacklist. In the next section we demonstrate the efficacy of our proposed techniques using real-world datasets.

IV. Experiments

In this section we demonstrate how the proposed technique can be used to identify malicious clusters of IP addresses utilizing compromised webmail accounts for various purposes, such as sending email spam, stealing contact lists, up-voting spam emails to fool spam filters, register to third party websites for comment spamming, etc. Our main focus is on IP clusters that utilize compromised user accounts, and unlike some of the previous work [1][2], we are not particularly interested in mass-registered accounts created and controlled entirely by attackers. While the proposed scheme can potentially identify clusters of IP addresses utilizing such mass-registered

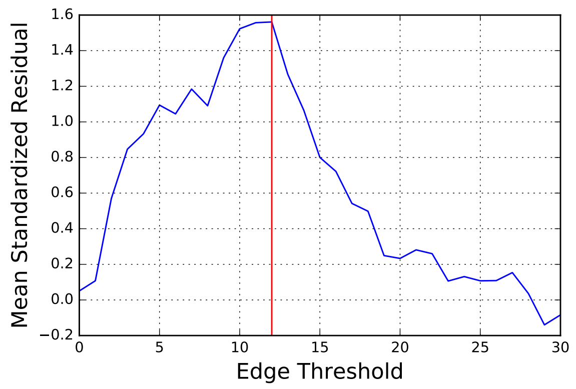


Fig. 4. Objective function value for a range of thresholds on a typical day. The optimal threshold, which is marked with red vertical line, is 12 on this particular day.

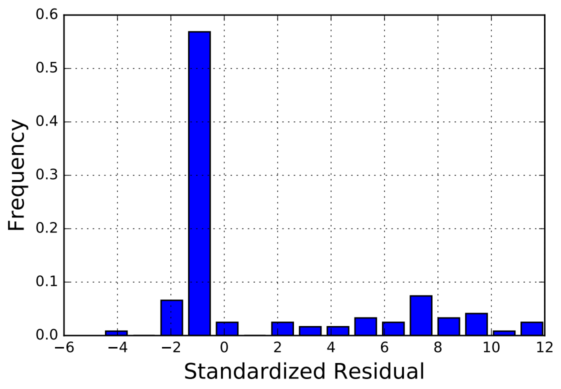


Fig. 5. The histogram of standardized residuals (R) for clusters obtained by optimal clustering on a typical day. Majority of clusters are benign, with standardized residual less than 3.

accounts, it has been shown that mass-registered accounts tend to yield much larger clusters exhibiting particular behavioral patterns, which lead to various detection techniques [1][2]. Such domain specific patterns and techniques are outside the scope of our experiments presented in this work.

To find malicious IP clusters utilizing compromised accounts, we use the dataset of real-world login events observed at the login servers of a large webmail service with millions of active users. We use IP address and the anonymized account id from each login event observed within a given 24-hour period to build an IP-IP graph. The reason why we use 1-day of data to build the graph is to utilize as much data as possible without introducing too much IP churn (i.e. hosts changing IP addresses) in the dataset. As explained in Section II-B, vertices are IP address on this IP-IP graph and an edge weight represents the number of accounts that are logged in from both of the corresponding IP addresses. We do not use any other field of login events dataset in this analysis. Also, for simplicity, we only consider successful login events coming via SMTP or IMAP protocols or from desktop browsers. We expect most malicious activities to originate from these login sources and hence exclude all other sources, including all kind of mobile devices. Here we would like to emphasize that the collected data is stored and utilized in full compliance with the webmail service provider's data governance policies.

After constructing the graph, we optimally cluster the IP addresses using the method proposed in Section II-B. Then we identify malicious clusters using a third party proprietary IP-blacklist. As explained in Section III, we use this blacklist to compute standardized residuals during both clustering and identifying malicious clusters. To provide realistic results, we use the blacklist snapshot captured one day before the analysis date ensuring only past information is used during the analysis. We do not perform any further post processing to verify malicious clusters by analyzing their behavior. In fact, one of the main contributions of this work is to eliminate such costly post processing analyses which incorporate specific kinds of domain knowledge, hence limit the scope to certain subset of malicious activities.

In order for our findings to be useful in practice, the analysis has to be performed frequently on the data from short periods of time. The reason is that IP addresses has long been known to switch frequently between being malicious and non-malicious due to IP churn, hosts getting cleaned, etc. [19], [20], [21], and therefore identified malicious clusters of IP addresses may quickly become inactive or cease to be malicious. For this purpose, we run our analysis daily on the dataset collected on each day, as opposed to previous works where analyses are performed on several months of aggregated data. Once daily results are produced, they can be utilized in variety of ways such as blocking identified malicious IPs or using the traffic from identified IPs as positive labels to train machine learning classifiers designed to detect various malicious behaviors. Below we explain each step of the daily process in detail.

A. Building the Graph

For a given day, we first build the IP-IP graph where nodes are IP addresses and there is an edge between two nodes if the corresponding IP addresses are used to log in at least one common account on that day. The actual number of common accounts logged in is represented by the edge weight. To give an idea about the scale, the resulting graph on a typical day has more than $700,000$ nodes and 2 million edges, after removing isolated nodes (i.e. IP addresses which have no edges).

B. Optimal Clustering

Once the graph is built, we find the optimum threshold value which we use to extract IP clusters from the graph. To find the optimum threshold, we compute the objective function given in Equation (3) for a range of threshold values and pick the one which maximizes the objective function. After finding the optimal threshold, we remove all the edges with weight less than the optimum threshold from the graph. Finally we output the connected components of the resulting graph as IP clusters. Note that, we ignore the clusters with size smaller than 5, since

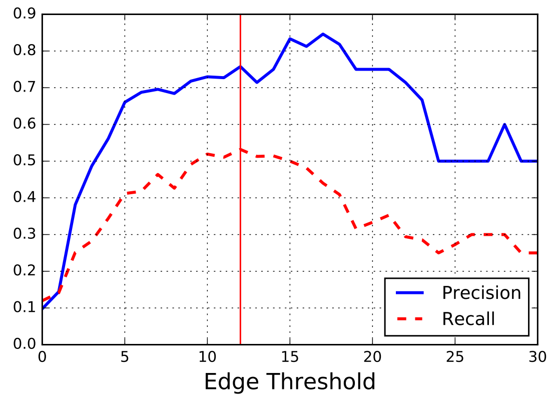


Fig. 6. Precision and Recall values for various edge weight thresholds on a typical day. We observe that the optimum threshold, which is marked with a vertical red line, is a reasonable choice yielding 70% precision and 50% recall.

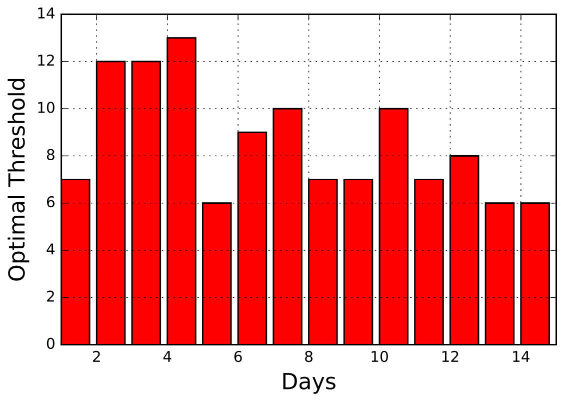


Fig. 7. Optimal thresholds for each day of daily analysis.

standardized residuals cannot be accurately computed for such small clusters [16]. After removing these small clusters, we typically end up with few hundred clusters for each day.

Finding the optimal threshold is essentially a grid search as presented in Section II-B2. Since the grid is 1-dimensional in our setting, the runtime complexity of optimization procedure is linear with the length of the grid. But each step of this linear-time procedure can potentially be computationally heavy since corresponding connected components need to be extracted and average standardized residual needs to be computed for each possible threshold value. On the other hand, this grid search process is highly parallelizable, since different parts of the grid can be searched independently in parallel. Therefore the actual CPU time to find the optimal threshold varies significantly depending on the available computational resources. Nonetheless, to give a rough idea, single threaded search for the optimal threshold within the range of (0,30) on a typical day's data takes about 60 seconds, where we use a virtual host with 2GHz 64-bit QEMU Virtual CPU on a cloud platform and employ NetworkX Python library [22] to extract connected components.

To depict this optimization process, we plot the objective function for a range of threshold values on a typical day in Figure 4. We clearly see that the optimum threshold on this particular day is 12, which means that a pair of IP addresses have to log in more than 12 common accounts in order to be connected on the graph and hence be on the same cluster.

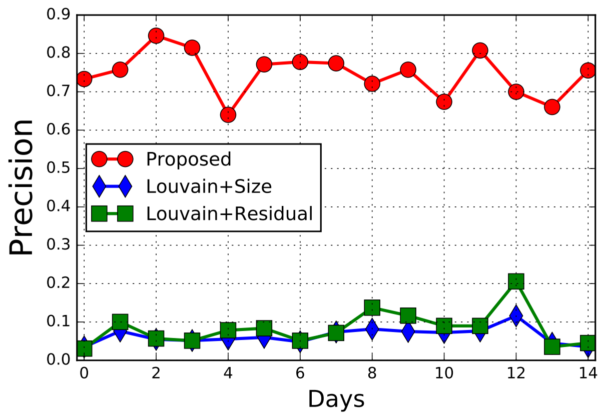
C. Making the Decision

Recall from Section III that, we deem a cluster to be malicious if its standardized residual is greater than 3. To see how malicious the resulting clusters are in a typical day, we plot the histogram of standardized residuals of the resulting clusters in Figure 5. We see that majority of clusters are benign, with standardized residual less than 3. But we also also several malicious clusters with very high standardized residuals.

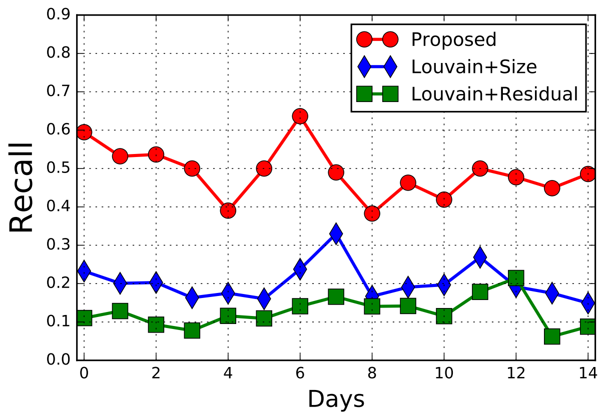
D. Ground Truth

To evaluate the detection performance of the proposed scheme, we need some form of ground truth information. As acknowledged by previous works, a complete ground truth is very hard to come by in security applications and our case is no exception. Nonetheless, at least to give us some insight, we employ a form of outbound spam email classifier as an incomplete source of ground truth. The outbound spam classifier, is a fairly complex system utilizing numerous features to decide whether an outbound email sent by a user of the webmail service is spam or not. To build the ground truth for each day, similar to [1], we label a cluster to be malicious if more than 10% of its IP addresses send at least one spam email on that day according to the outbound spam classifier. Note that the proposed scheme does not use this outbound spam classifier in any way in its decision process. We merely use it for evaluation purposes.

Using this ground truth, we can now compute *precision* and *recall* metrics for a given day. **Precision** indicates what fraction of the detected clusters is actually malicious. On the other hand, **recall** indicates what fraction of all the malicious clusters in the entire dataset is detected by the proposed scheme. Notice that, although we would like to see high precision and recall values in general, we should not expect to achieve perfect precision with this ground truth. The reason is that, the proposed scheme is likely to identify additional malicious clusters which are not used to send spam, hence missed by ground truth. After all, if our goal was perfect precision, than we would not need to design any other detection scheme since the outbound spam filter itself would already be doing the desired task perfectly. On the other hand, we would ideally like to see perfect recall, since we would like to be able to detect all known malicious clusters sending spam. However, in practice we usually do not observe perfect recall, which can be attributed to blacklist's true positive rate being too low for some of the malicious clusters, which in turn have small standardized residual values as explained in Section III. Although we can achieve satisfactory recall performance with the current blacklist, one way to improve recall is to use a



(a) Precision



(b) Recall

Fig. 8. Precision and Recall values for 15 days. The proposed scheme performs consistently better than the baseline both in terms of precision and recall

better blacklist or perhaps multiple blacklists simultaneously.

To closely inspect our detection performance, we plot precision and recall for different threshold values on a typical day in Figure 6. Notice that we do not use precision and recall values when searching for the optimal threshold and therefore we cannot expect the optimal threshold to yield the best possible performance in terms of precision and recall. Nevertheless, we observe that the optimal threshold is indeed quite reasonable. Although precision would be improved by a slightly higher threshold, recall would actually start to decline.

E. Running for Several Days

To investigate the performance of the proposed scheme over a larger time window, we repeat each of the above steps for several days. Depending on the traffic and the resulting graph, on each day we end up having a different optimal threshold ranging from 6 to 13, as shown in Figure 7. After performing optimal clustering using these thresholds, we declare the clusters whose standardized residual is greater than 3 as malicious for each day. Recall that, with this threshold, our false positive rate is less than in 0.3% since the probability of standardized residual for benign clusters being greater than 3 just by chance is less than 0.3%.

To serve as baseline, we also try to identify malicious clusters using suggestions proposed in [1]. In that work, authors first build a graph from a dataset containing IP address/account id pairs. Then they extract clusters using a modularity-based agglomerative graph clustering scheme, called Louvain Method [23]. Finally, authors observe that most of the malicious clusters tend to be very large as opposed to benign clusters. Based on this insight, we apply Louvain Method to the IP-IP graph we construct each day. We then declare clusters with size greater than a threshold as malicious. We call this baseline method **Louvain+Size**. We empirically pick this threshold to be 10, which seems to yield the best performance in terms of precision and recall. Note that, this is not a direct comparison of the proposed work to the work presented in [1]. We merely follow the insights presented in [1] to devise a baseline experiment. In fact, the work presented in [1] and this work are not compatible and cannot be compared directly. The major difference is that, in this work we deal with IP-IP graphs to identify malicious clusters of IP addresses, whereas in [1] authors work with account-account graph and their output is malicious clusters of user accounts.

Furthermore, in order to assess the effect of the proposed *blacklist-aware* clustering alone, we design another baseline scheme which we call **Louvain+Residual**. In this scheme we first extract IP clusters using Louvain Method and then deem the resulting clusters with standardized residual greater than 3 as malicious. Here we use the same blacklist as the proposed scheme, therefore the only difference between Louvain+Residual and the proposed scheme is the choice of clustering method.

After identifying malicious clusters with both the proposed scheme and the baseline methods, we compute precision and recall values each day using that day's ground truth. We plot the results in Figure 8. We observe that the proposed scheme performs consistently and significantly better than both baseline methods in terms of precision and recall. Also the results suggest that making the clustering *blacklist-aware* is crucial in detecting malicious IP clusters since the proposed scheme significantly outperforms Louvain+Residual method.

When we closely look at the detected clusters, we observe that the proposed scheme produces much smaller clusters than Louvain method does. This is somewhat expected since the proposed scheme ignores weak edges with weight less than the optimal threshold and only outputs clusters where IP addresses are strongly connected to each other. In a sense, the proposed scheme prunes loosely connected branches of IP clusters and outputs the core structures which are highly correlated with the blacklist. On the other hand, Louvain method involves no such pruning thereby producing much larger clusters with many more IP addresses several of which are rather loosely connected.

F. Additional Detections

We observe in Figure 8(a) that, precision of the proposed scheme is around 75% on average. As mentioned before, this is expected since the ground truth is not complete. To understand the nature of these additionally detected clusters, we manually investigate some of them. We observe some interesting cases where additional detections are seemingly malicious in a sense that we did not foresee. For example, we discover several clusters of IP addresses having large number of failed login attempts to numerous user accounts. At first glance, it seems these IP addresses should not form any clusters, since we only consider successful login events in our analysis. But upon closer inspection we realize that, while generating a lot of failed login attempts, many of these IP addresses managed to successfully log in same subsets of accounts and hence form clusters on the IP-IP graph. We believe that these are actually distributed password guessing attacks, where passwords of some user accounts are correctly guessed by multiple IP addresses, since they probably use the same method and dictionary to guess passwords.

We also observe more ambiguous cases as well. For example we see that some detected clusters are composed of very similar IP addresses sharing long prefixes with each other. These clusters essentially represent IP addresses within subnets. On one hand, such clusters could be false positives where a single infected machine in a subnet got assigned several different IP addresses over time and managed to get most of them blacklisted. On the other hand, those clusters can indeed be a set of malicious hosts sitting within a subnet and systematically utilizing compromised accounts in various ways. While we cannot immediately confirm maliciousness of those clusters using the data available to us at the time of the analysis, depending on the application requirements one can choose to ignore such detected clusters whose IP addresses seem to be within the same subnet.

G. Effect of Blacklist Performance

In Section III, we mathematically show that even mediocre blacklists can be used to accurately detect malicious clusters. To observe this in practice, we run an experiment where we gradually degrade the blacklist quality and measure the detection performance. Note that the original blacklist may not be very accurate in the first place. Nevertheless, we just would like to see how the cluster detection performance would be affected if we used lesser quality blacklists.

To degrade the blacklist quality, we first find all blacklisted IPs in our dataset for a typical day. Then we remove certain fraction of these blacklisted IPs from the blacklist and essentially making them “good”. Then, in order to keep the number of blacklisted IPs constant for consistency, we randomly pick same number of IPs from the entire dataset and add them to the blacklist. This way we significantly reduce the true positive rate of the blacklist. False positive rate, on the other hand, is not affected much and only increased by few percent since the number of randomly picked IP addresses is much smaller than the number of all IP addresses in our dataset. Since the experiment has a random component, to average out uncontrolled factors we repeat the same experiment 25 times and report the average. We plot precision and recall values for different percentages of blacklist corruption in Figure 9. We observe that, while recall is declining steadily as more of the blacklist is corrupted, the precision seems to stay stable around $75 - 80\%$ even until 80% of the blacklisted IPs are removed from the blacklist. Essentially, as blacklist gets worse, the proposed scheme misses some of the malicious clusters but continues to detect several others, which contain high numbers of blacklisted IPs. In conclusion, although the proposed scheme might miss some of the malicious clusters depending on the quality of the blacklist, the ones it detects are highly likely to be malicious.

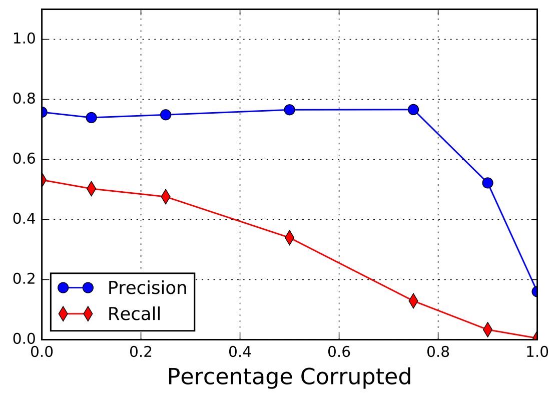


Fig. 9. Precision and Recall as the blacklist is gradually corrupted. Precision stay stable even until 80% of the blacklisted IPs are removed from the blacklist.

V. Limitations

The efficacy of the proposed scheme depends on two basic assumptions:

i) IP addresses in a malicious cluster exhibit some form of similarity to at least few other IPs in the same cluster
ii) At least some of the IP addresses in a malicious cluster are blacklisted.

Essentially, the detection performance would suffer from any effort by an attacker to break any of these assumptions. For example, to break the second assumption, an attacker might try to employ recently infected hosts whose IP addresses have not been blacklisted yet. Similarly, to break the first assumption, an attacker, who utilizes malicious web accounts, might force the sets of accounts logged in by IP addresses to be mutually disjoint. While one can design several other strategies to break these assumptions, implementing such strategies often increases the cost of the attack or limits its scope. For example, using only recently infected hosts significantly limits the number of IP addresses available to an attacker, hence limits the scale of the attack. Similarly, trying to force IP addresses to log into disjoint sets of compromised accounts requires more planning on attacker’s part, thereby increasing the cost of the attack. Basically attacker needs to design new methods to make sure the sets remain disjoint despite the hosts frequently go offline or change IP addresses.

Another limitation arises from the statistical nature of the proposed scheme. More specifically, as discussed in Section III, IP clusters need to be larger than a certain size—depending on the accuracy of the employed blacklist—in order for us to make accurate decisions. This essentially implies that the proposed scheme can only be employed by relatively large organizations interacting with thousands of IP addresses in a short time window (i.e. 24 hours). Smaller organizations whose data can only produce smaller IP clusters need to either utilize more accurate blacklists or collaborate with other smaller organizations to produce more accurate results collectively. On the positive side, such collaborations are not uncommon in the context of IP blacklisting [24][25].

IP addresses of cloud services, such as Amazon EC2 or Microsoft Azure, also pose a concern for the proposed scheme. More specifically, such IP addresses are shared resources and often utilized by independent users for independent tasks within a short amount of time. Therefore, they may be frequently switching back and forth between being part of malicious clusters and being part of benign clusters. In such cases, where identities and intentions of IP addresses change rapidly and continuously, IP clustering process may have hard time in placing those IP addresses into appropriate clusters, thereby increasing error rates. One way to address this issue would be to exclude IP addresses of known cloud service providers from the analysis and handle them separately.

VI. Related Work

Several works make the observation that group of IP addresses carrying out a certain task behave similarly. In [7],[1] and [2] authors present that compromised web accounts are utilized by similar sets of IP addresses. In [26] authors show that spamming IP addresses tend to send emails to similar sets of domains. Various detection methods based on similar network behavior that these groups of IP addresses exhibit are explored in several works. In [15] Nagaraja et. al. show that members of a peer-to-peer botnet communicate heavily with each other as part of their command and control activities and therefore form a tightly connected cluster on the who-talks-to-whom graph built from the network flow records captured at the routers of an Internet Service Provider (ISP). In [13] authors show that member of a peer-to-peer botnet within an enterprise network tend to communicate with similar sets of external IP addresses and therefore are likely to be connected each other on a mutual contacts graph, where vertices represent internal IP addresses and edges represent two vertices have communicated with at least one common external IP address. In [27] Gu et. al. show that botnet members in an enterprise network exhibit similar network behavior and therefore can be detected by applying a clustering scheme on various network level features. However, most of these works require network level data observed between IP addresses captured at ISP or enterprise level routers and therefore cannot be extended to datasets observed by web application service providers. The detection scheme presented in [2] is designed for web application service providers, however, it focuses on detecting groups of user accounts rather than groups of IP addresses. Furthermore, it requires costly behavioral analysis in order to identify malicious clusters. In the proposed work, we identify malicious clusters of IP addresses by simply using blacklists and without requiring any costly behavioral analysis. Similarly in [1] Stringhini et. al. present another scheme for web application service providers, where they aggregate months of data in order to reliably identify malicious clusters of user accounts. However, as mentioned before, our focus in the proposed work is on IP addresses rather than user accounts and therefore we are restricted to process data aggregated for much shorter time windows (i.e. 1 day), since hosts frequently change IP addresses due to dynamic IP allocations (i.e. IP churn). Despite this restriction, we are still able to present high detection accuracy in the proposed work.

The proposed work heavily relies on IP blacklists. Effectiveness of some well-known IP blacklists has been explored in various studies. In [17], authors show that IP blacklists suffer from "non-trivial" false positives and false negatives. There have been several works to improve this shortcoming. In [28] authors study potential reasons for poor performance of IP blacklists, such as targeted and low volume attacks and reactive nature of blacklists. To mitigate these issues, authors then propose dynamic thresholding and speculative aggregation strategies. In [29], authors propose personalized blacklisting strategy based on the observation that common attack sources are reported by same set of victims presented in [30]. Although these works mostly verify that blacklists are not perfect and often plagued by significant accuracy issues, in this work we show that even a mediocre blacklist is sufficient in order for the proposed scheme to accurately identify malicious groups of IP addresses.

Besides this work, there are few other studies where clusters of IP addresses are judged by using blacklists. In [14], authors cluster peer-to-peer communities together from network communication patterns and identify potentially malicious clusters using a blacklist. Authors apply a similar strategy to mobile networks in [31] and reveal several scam activities. In [32] and [28] IP addresses are clustered based on their proximity on the IPv4 address space and potential malicious groups are identified using blacklists. However, these methods are rather ad-hoc and does not provide any principled insight into the usage of blacklists. Furthermore, they use blacklists only after clustering is completed. In this work, we present a scheme which incorporate a blacklist into clustering scheme itself in order to produce optimal clusters with high statistical dependency to the blacklist. We also provide a sound mathematical analysis to understand how useful a blacklist of a certain accuracy would be in detecting clusters of various sizes.

VII. Conclusion

In this paper we have presented a novel scheme where we utilize mediocre blacklists to make accurate decisions on clusters of IP addresses. Basically, we argued that many complex activities on the Internet are carried out by clusters of IP addresses, which leave quite noticeable traces at various vantage points. We discussed that simple data processing reveals such clusters of IP addresses. To that end, we proposed an optimal clustering scheme which yields malicious IP clusters which are highly correlated with a given blacklist. Then we argued that, once a cluster of IP addresses is identified, it is surprisingly easy to determine whether it is malicious or not using a blacklist. In addition, we mathematically showed that even a very poor blacklist can yield quite accurate results, so long as it performs better than random guessing. The basic idea is that, as long as a blacklist's true positive rate is greater than its false positive rate, the expected number of blacklisted IPs in a bad cluster is distinguishably larger than that in a good cluster. We demonstrated the efficacy of the proposed technique using real-world dataset of login events observed at the login servers of a large webmail service with hundreds of millions of active users. Our results show that the proposed scheme consistently outperforms the baseline schemes during several days of experimentation. Finally, we also showed that the proposed scheme identifies additional malicious activities which we did not foresee at the beginning.

References

[1] G. Stringhini, P. Mourlanne, G. Jacob, M. Egele, C. Kruegel, and G. Vigna, "Evilcohort: Detecting communities of malicious accounts on online services," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015.

[2] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum, "Botgraph: Large scale spamming botnet detection," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2009, April 22-24, 2009, Boston, MA, USA*, 2009, pp. 321–334.

[3] B. Stone-Gross, C. Kruegel, K. Almeroth, A. Moser, and E. Kirda, "Fire: Finding rogue networks," in *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, Dec 2009.

[4] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao, "You are how you click: Clickstream analysis for sybil detection," in *22nd USENIX Security Symposium*, 2013.

[5] J. Zhang and G. Gu, "Neighborwatcher: A content-agnostic comment spam inference system," in *20th Annual Network and Distributed System Security Symposium, NDSS*, 2013.

[6] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "The socialbot network: When bots socialize for fame and money," in *Proc. of the Annual Computer Security Applications Conference 2011*, 2011.

[7] K. Thomas, C. Grier, and V. Paxson, "Adapting social spam infrastructure for political censorship," in *Presented as part of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2012.

[8] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna, "The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns," 2011.

[9] A. Pitsillidis, K. Levchenko, C. Kreibich, C. Kanich, G. M. Voelker, V. Paxson, N. Weaver, and S. Savage, "Botnet judo: Fighting spam with itself," 2010.

[10] B. Coskun and P. Giura, "Mitigating sms spam by online detection of repetitive near-duplicate messages," in *in IEEE International Conference on Communications (ICC)*, 2012.

[11] M. Bailey, E. Cooke, F. Jahanian, Y. Xu, and M. Karir, "A survey of botnet technology and defenses," in *Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology*, March 2009.

[12] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, vol. 7. Cambridge, MA, 2007.

[13] B. Coskun, S. Dietrich, and N. Memon, "Friends of an enemy: Identifying local members of peer-to-peer botnets using mutual contacts," in *Annual Computer Security Applications Conference (ACSAC)*, Dec 2010.

[14] L. Li, S. Mathur, and B. Coskun, "Gangs of the internet: Towards automatic discovery of peer-to-peer communities," in *IEEE CNS'13: First IEEE Conference on Communications and Network Security*, 2013.

[15] S. Nagaraja, P. Mittal, C. yao Hong, M. Caesar, and N. Borisov, "Botgrep: Finding p2p bots with structured graph analysis," 2010.

[16] A. Agresti, *An Introduction to Categorical Data Analysis*, 2nd ed. Wiley Inter-Science, 2007.

[17] S. Sinha, M. Bailey, and F. Jahanian, "Shades of grey: On the effectiveness of reputation-based blacklists," in *3rd International Conference on Malicious and Unwanted Software (MALWARE)*, Oct 2008.

[18] "https://www.intra2net.com/en/support/antispam/."

[19] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "My botnet is bigger than yours (maybe, better than yours): Why size estimates remain challenging," in *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*, ser. HotBots'07, 2007.

[20] ——, "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '06, 2006.

[21] B. Stone-Gross, M. Cova, B. Gilbert, L. Cavallaro, M. Szydlowski, C. Kruegel, G. Vigna, and R. Kemmerer, "Your Botnet is My Botnet: Analysis of a Botnet Takeover," in *Proceedings of the Computer and Communications Security Conference (CCS)*, Chicago, IL, November 2009.

[22] "http://networkx.github.io/."

[23] V. Blondel, J. Guillaume, R. Lambiotte, and E. Mech, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, 2008.

[24] "https://www.projecthoneypot.org."

[25] "https://www.dshield.org."

[26] A. Ramachandran, N. Feamster, and S. Vempala, "Filtering spam with behavioral blacklisting," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007.

[27] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th USENIX Security Symposium (Security'08)*, 2008.

[28] S. Sinha, M. Bailey, and F. Jahanian, "Improving spam blacklisting through dynamic thresholding and speculative aggregation." in *Symposium on Network and Distributed System Security (NDSS)*, 2010.

[29] J. Zhang, P. A. Porras, and J. Ullrich, "Highly predictive blacklisting," in *Proceedings of the 17th USENIX Security Symposium*, 2008.

[30] S. Katti, B. Krishnamurthy, and D. Katabi, "Collaborating against common enemies," in *ACM Internet Measurement Conference*, 2005.

[31] N. Boggs, W. Wang, S. Mathur, B. Coskun, and C. Pincock, "Discovery of emergent malicious campaigns in cellular networks," in *ACSAC'13: Annual Computer Security Applications Conference (ACSAC)*, 2013.

[32] S. Mathur, B. Coskun, and S. Balakrishnan, "Detecting hidden enemy lines in ip address space," in *NSPW'13: New Security Paradigms Workshop*, 2013.