Detecting Hidden Enemy Lines in IP Address Space

Suhas Mathur and Baris Coskun
AT&T Security Research Center
New York, NY
{suhas, baris}@att.com

Suhrid Balakrishnan
AT&T Labs Research
Florham Park, NJ
suhrid@research.att.com

ABSTRACT

If an outbound flow is observed at the boundary of a protected network, destined to an IP address within a few addresses of a known malicious IP address, should it be considered a suspicious flow? Conventional blacklisting is not going to cut it in this situation, and the established fact that malicious IP addresses tend to be highly clustered in certain portions of IP address space, should indeed raise suspicions. We present a new approach for perimeter defense that addresses this concern. At the heart of our approach, we attempt to infer internal, hidden boundaries in IP address space, that lie within publicly known boundaries of registered IP netblocks. Our hypothesis is that given a known bad IP address, other IP address in the same internal contiguous block are likely to share similar security properties, and may therefore be vulnerable to being similarly hacked and used by attackers in the future. In this paper, we describe how we infer hidden internal boundaries in IPv4 netblocks, and what effect this has on being able to predict malicious IP addresses.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; I.5.3 [**Pattern Recognition**]: Clustering; G.3 [**Probability and Statistics**]: Multivariate statistics

Keywords

Clustering, Predictive modeling, Blacklists, Statistical fingerprinting

1. INTRODUCTION

Firewalls and Network Address Translation (NAT) devices have become the industry standard in providing perimeter defense for enterprise networks. By blocking incoming network connections and hiding hosts within an enterprise network from the external world, they have been quite successful in protecting internal hosts against attacks which actively seek and exploit victims, such as scanning, worm propagation, password guessing, etc. However, such perimeter defenses have started to lose their efficacy as attackers have adapted their strategies. For instance, a significant fraction of recent malware evades such perimeter defenses by tricking unsuspecting users to click malicious links in an email or on a website. Unfortunately, existing network perimeter defense systems provide little to no protection against such strategies since malicious network connections are all initiated from within the protected network. Furthermore, once an internal host is infected, the perimeter defense completely falls apart and infected hosts can freely communicate with their command and control servers (i.e. botnets) or exfiltrate sensitive information to external dropservers (i.e. advanced persistent threats), etc.

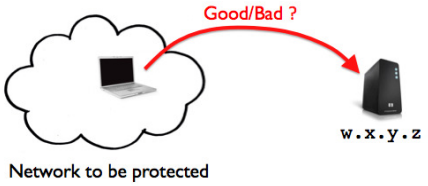


Figure 1: How should known bad IP addresses inform whether an outbound flow may be suspicious?

Motivated to fix this outdated form of protection, in this work we focus on network connections initiated from within the protected network. This is in stark contrast to conventional perimeter defense mechanisms. More specifically, we aim to answer whether an outgoing network connection initiated by an internal host is malicious or not, the question depicted in Figure 1.

One simple and standard way to partly address these security concerns is to utilize blacklists of IP addresses that have exhibited malicious behavior in the recent past. That is, a network flow destined to a known malicious IP is highly likely to be malicious as well. However, this approach has the obvious limitation that it can only apply in situations where the destination IP address is already in some blacklist. Malicious IP addresses that have not been blacklisted yet because they have been compromised only recently (and therefore have not exhibited excessive malicious behavior yet), or whose operators have just recently changed their IP addresses, can slip by completely unnoticed. This raises the following interesting question: *"Can we say whether a flow*

NSPW'13, September 9–12, 2013, Banff, AB, Canada.

http://dx.doi.org/10.1145/2535813.2535816.

is likely to be malicious even if the destination IP address is not on a blacklist, but it appears to be close in IP address space, to one or more IP addresses that are on a blacklist?" We believe the answer is *"Yes!"*. The intuition we appeal to is that a set of hosts with IP addresses in close vicinity are generally used and administered in a similar fashion therefore share similar security properties. For instance hosts in a lab of a university, or in a department of a corporation, or servers of a hosting company often connect to a same subnetwork and hence have IP addresses very close to each other on the IP address space. Since such a group of hosts are administered by a same entity, they share similar vulnerabilities, such as running the same vulnerable operating system version or being used by unsuspecting users with a similar profile, etc. Therefore if one host is known to be compromised then others in the vicinity might be vulnerable to a similar type of compromise.

In order to study whether malicious IP addresses tend to be concentrated in specific regions *within* the IP netblock in which they appear (as opposed being uniformly distributed inside the netblock), we utilized a dataset of about $74,000$ IP addresses blacklisted by SpamHaus, and a second dataset of about $4,000$ botnet C&C servers. To test the clustering hypothesis, given a netblock with N IP addresses, k of which are blacklisted, we let the null hypothesis be that the k addresses were chosen uniformly randomly from among the $\binom{N}{k}$ possibilities. We define the metric of interest M, to be the median of the distance between each of the k selected IP addresses and the nearest other selected IP address. Using Monte Carlo simulations, we compute the distribution of this metric under the null hypothesis for each setting of the (N, k) tuple that appears in our datasets. The observed value of the metric for a given netblock is then compared with the empirical distribution of M computed by simulation under the null hypothesis to determine a *p-value*, i.e., the area under the empirical distribution for metric values less than or equal to the observed value. The p-value therefore measures how likely it is to observe a given metric for a given netblock, if the null hypothesis were true. Therefore, a small p-value is evidence against the null hypothesis. Results from our study are summarized in Figure 2. Boxplots for the null hypothesis summarize the distribution of the mean value of M normalized by dividing by the size of the netblock (N). Boxplots for the observed cases summarize the distribution of the observed value of M, also normalized by dividing by N. 37% and 44% of the net blocks have p-values less than 0.05 in the SpamHaus dataset and in the Botnets C&C dataset respectively. Even though some netblocks seem to have malicious IP addresses close to uniformly distributed inside them, there is clear evidence of clustering of malicious IP addresses for a large fraction of netblocks.

However, it is unclear to what extent the influence of a known bad IP address is in terms of casting suspicion over IP addresses in its vicinity. For example, if X.X.0.1 is known to be a C&C server, should flows going to X.X.0.2 be considered suspicious? What about flows going to X.X.2.1? In this paper, we attempt to quantify the extent, in IP address space, that a bad IP address should have on other IP addresses.

It would be easy to answer this question if the internal IP address space boundaries of subnets containing similar hosts were known. In such a case, a known malicious IP address would cast suspicion over other IP addresses residing in the same internal subnet. Unfortunately, information on internal boundaries is almost never available in such fine granularity. In this paper we show that nonetheless, it is possible to *infer* these boundaries from network traffic characteristics observed for each IP address.

We view an IP address as a data point comprising a rich set of features extracted from network behavior observed at the backbone of a large ISP, over a period of several weeks. We then partition the IP address space around known malicious IP addresses such that IP addresses within a partition exhibit similar traffic feature characteristics. We deem any internal subnet that contains a known malicious IP address, as being potentially malicious. In other words, if an outbound flow is observed destined to an IP address that lies in the same inferred (by us) cluster as a known malicious IP address, then we deem that flow to be suspicious.

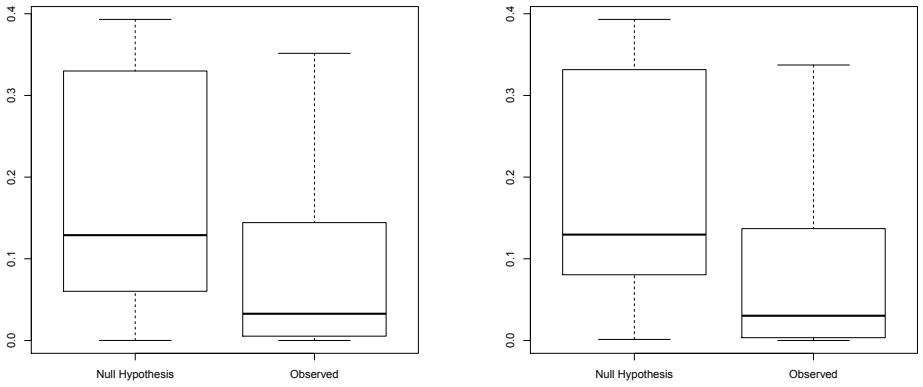
We discuss related work in Section 2, and then further motivate the new paradigm we are proposing in Section 3. In Section 4 we describe the data we collected in order to infer the hidden internal boundaries in IP netblocks. In Section 5, we describe our methodology for finding these hidden boundaries, and in Section 6, we study whether the inferred boundaries seem useful in predicting malicious IP addresses. We conclude in Section 7 with a list of planned future directions of exploration.

2. RELATED WORK

A number of prior research efforts have highlighted the fact that malicious hosts tend to heavily cluster in IP address space, rather than be scattered uniformly all over it. [16] aims to discover regions of IPv4 space that are run by parties that deliberately support malicious activities. [17] analyzes lists of known bad IP addresses for how well they cluster in IP address space. The reason malicious hosts tend to appear in a highly clustered manner in IP address space is that some networks are maintained well (regularly patched, well configured, etc.) while some networks are poorly maintained (e.g. running old, unpatched software, or misconfigured). This leads attackers looking for vulnerable internet-connected hosts to favor the poorly managed networks when they need to set up a malicious host. Our work build on top to the insights in [16, 17].

The problem of predicting whether a given IP address may be malicious is closely related to the problem of predicting whether a newly created domain name is malicious. The work in [8] builds a classification model for domains using several input features derived from DNS records, and properties of the IP addresses the domain name points to, etc. [9] using features derived entire from the structure of the domain name's string in order to classify it as a legitimate or malicious domain name.

A large body of work has focused on the reputation of IP addresses as senders of traffic [10, 11, 12, 13, 14, 15]. It is surprising to note the apparent lack of attention paid to IP addresses as *receivers* rather than initiators of malicious traffic, among researchers working on the notion of reputation. This focus on IP addresses as senders of malicious traffic is at apparent loggerheads with present-day reality of network malware, which seems to heavily rely on an outgoing flow from the (soon-to-be-) victim's network to a malicious host, C&C server or dropserver.



(a) SpamHaus Dataset

(b) Botnet C&C Dataset

Figure 2: Boxplots showing the distributions of the mean value of the metric of interest M, namely the median of the distances between a bad IP and the closest other bad IP. The mean is normalized by dividing by the size of the netblock. Boxplots are shown for the null hypothesis and the observed values, for each of the two datasets.

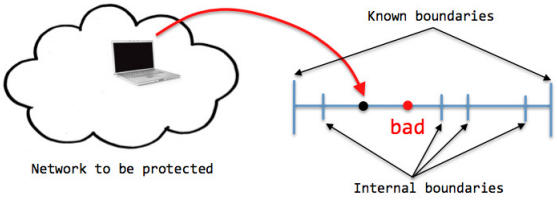


Figure 3: If an outbound flow from a network is observed destined to an IP address that shares an internal block with a known malicious IP address, then we propose that this flow should be considered suspicious, especially if there has been no traffic bound for that portion of IP space from this network in recent times.

3. CASTING SUSPICION BY INFERRING HIDDEN BOUNDARIES

Given a known malicious IP address, it is easy to ascertain the smallest publicly known registered IP netblock in which it lies, using the publicly available whois database. To take a concrete but hypothetical example, suppose we know that X.X.0.1 is a malicious IP address, then we can easily figure out using publicly available records, that it lies in the block X.X.0.0/16, owned by some organization. The question is: what is the chance that other IP addresses in the vicinity of this one known malicious IP address are compromised and either currently being used or will shortly be used for malicious purposes?

At one extreme, a policy we can adopt is to assume that all IP addresses that lie in the same netblock as the known malicious IP address are suspicious, and we would therefore block flows going to any of these IP addresses. In our hypothetical example, this would amount to blocking all flows destined to any IP address in the X.X.0.0/16 block, which contains $2^{16} = 65,536$ IP addresses. Clearly, this approach is not very practical because the number of publicly known netblocks containing least one malicious IP address is immense, and therefore blocking each such block would amount to blocking a very large fraction of IPv4 space.

The opposite extreme, blacklisting, is closer to the current de-facto standard. Here we only consider the individual bad IP addresses themselves to be suspicious, and worthy of being blocked, but allow all flows destined to any other IP address, even if it is just one address away from a known bad IP address. This strategy is also clearly lacking. What if another host, in the same network as a known bad IP address, is also infected and we allow traffic destined to it? What if a host with a known bad IP address is re-assigned a different IP address from within a small block of addresses? We seek to strike a balance between the two extremes above, by attempting to quantify which IP addresses in the vicinity of a known bad IP address are more susceptible to turning malicious in the future.

Our approach. Our central idea is that publicly available netblocks are further partitioned into contiguous internal blocks (subnets) of IP addresses (see Figure 3). An internal block may correspond to a set of hosts used for a specific purpose (e.g. web servers), or by a specific group of people (e.g. a specific office or lab), or it may simply represent address range of a Dynamic Host Configuration Protocol (DHCP) server. The common thread connecting IP addresses belonging to an inner block is that they are likely to be administered in a similar way and running similar software, and therefore have similar security properties. Therefore, if one IP is malicious in an inner block, the other IPs within the same block should be treated with suspicion. To leverage this suspicion spilling over to neighboring IP addresses, we envision a two-component system protecting an enterprise network:

- The first component constantly collects information on destination IP addresses for all outbound flows from the network perimeter, so as to build a probability distribution over IP address space. Essentially, this component serves the function of an anomaly detec-

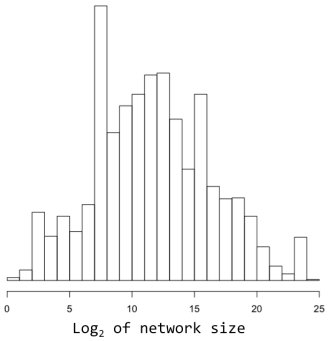


Figure 4: Histogram of the size (log base 2) of netblocks containing at least one malicious IP address.

tor, allowing the computation of a probability that the destination IP address (or the public netblock in which it lies) is anomalous for outbound flows from this network.

- When an anomalous outbound flow is detected using the first component (i.e. it is destined to a highly unlikely IP address), the second component attempts to ascertain whether the destination IP address lies within the same inner block as a known malicious IP address from the recent past, so as to warrant suspicion. If the flow is deemed to be suspicious, one can imagine variety of actions that can be taken, depending on the governing security policy, such as blocking, allowing with scrutiny, requiring further authorization from the user before allowing, etc.

Unsurprisingly, the efficacy of such a system would highly depend on precise knowledge of the inner block boundaries. As mentioned before, these inner boundaries are hidden from public view. In this work, we aim to infer these inner boundaries using a large number of features derived from network flows going to and from the IP addresses that constitute the larger publicly known netblock, over a large interval of time (weeks/months). Our intuition is that, since IP addresses within an internal block are used in a similar manner they are likely to exhibit similar network level behavior, which is somewhat different than the behavior of IP addresses that reside on the other side of the boundary. As a result, the boundaries between internal blocks should feature discernible changes in network behavior. This intuition is clearly observed in Figure 5, where network features we compute from observed network traffic flow sharply change their characteristics across what we believe are inner block boundaries.
To identify the hidden boundaries inside a given netblock, we utilize network flow records captured at the backbone of a Tier-1 ISP for an extended period of time. Then we apply a specific clustering method which is tailored to our problem. Details of data collection and clustering method are presented in Section 4 and Section 5, respectively. We conclude this section with a discussion on a number of aspects that impact our proposed system.

NATs: A NAT merges the traffic from multiple hosts situated behind a gateway, and thus appears to the public internet as a single IP address. A group of NAT gateways with consecutive IP addresses (e.g. a group of home routers of an ISP) will be grouped into a single contiguous cluster if the long term characteristics of traffic emerging from the NAT gateways are similar. If one of the NAT gateway IP addresses is known to be malicious (say, a NAT router was hacked into by an attacker exploiting a vulnerability), then we posit that flows destined to other NAT gateways in proximity of this IP address should be viewed with suspicion, since they may be vulnerable to the same exploit. Note that a host situated behind a NAT cannot appear on a malicious blacklist since it is not publicly routable.

Dynamic IP address allocation: IP addresses in some netblocks may be dynamically assigned and therefore, the same address may represent different hosts at different points in time. However, dynamic IP address assignments typically rotate a given IP address among hosts of the same type (e.g. a set of NAT gateways). In this case, even though hosts may change their IP addresses, as long as similar hosts rotate among a pool of IP addresses, that pool of IP addresses is likely to be inferred as a single contiguous block consisting of hosts with similar behavior over a long time interval.

IPv6: Our approach for partitioning an IP netblock into behaviorally distinct contiguous sub-blocks requires the collection of network flow data for all IP addresses in the netblock. This implies that applying this method to IPv6 may lead to scalability issues. It is therefore worth exploring methods to discover only boundaries that immediately enclose known malicious IP addresses, instead of having to partition entire netblocks. After all, we are only really interested in internal blocks that contain one or more known malicious IP addresses.

Missing data: It is possible to observe no network data from some IP addresses. In fact in our experimental datasets, we often observe no data of a certain type (e.g. no ICMP flows) from some IP addresses over the entire period of observation. Rather than let this be a hurdle in our attempt at characterizing IP addresses, we recognize that the absence of data from an IP addresses over a long measurement horizon, is itself a valuable piece of information because it tells us something about its behavior.

Shared hosting: Many IP addresses in the Internet host multiple web properties, often with a large number of domain names pointing to the same IP address. Often, attackers compromise such shared hosting IP addresses to use them as an attack endpoint. Suppose that of the several domain names pointing to an IP address, only one (or none) is malicious. Should an outbound flow from an enterprise be allowed to go to this IP address if it is based on one of the non-malicious domain names? We believe not. Therefore, even if an IP address is part of a shared hosting environment, if it appears in the same contiguous behavioral block as a known malicious IP address, we posit that a flow going to it should be looked at with suspicion.

4. DATA COLLECTION & PREPARATION

As mentioned in the previous section, our approach for assigning suspicion to IP addresses using a known malicious

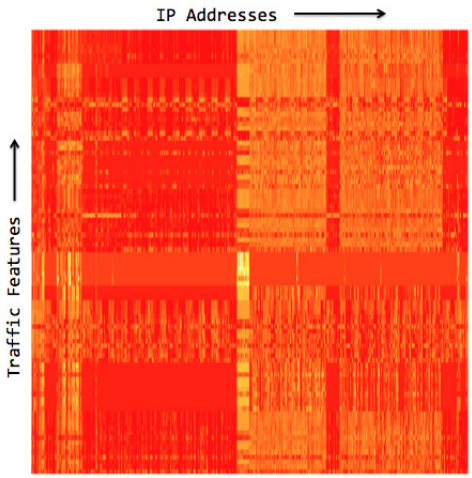


Figure 5: Network traffic features for IP addresses within a publicly available netblock. Vertical axis corresponds to different features and horizontal axis corresponds to IP addresses. Abrupt changes in features across inner block boundaries are clearly visible.

IP address is by monitoring all IP addresses in netblock containing one ore more known malicious IP addresses, and building network behavioral profiles of all these IP addresses over time. In other words, given a malicious IP address, we first determine using public registry information, the smallest registered netblock that contains that IP address. Then we collect netflow data for all IP addresses in the netblock, in order to build behavioral profiles for each IP address over a long period of time. These profiles are then used to infer clusters of contiguous blocks in IP-address space. We next describe the netflow data we collect. We follow this data description by detailing our feature based behavioral profile construction per monitored IP address.

4.1 IP netblock data

Given a publicly known IP netblock containing one or more bad IP addresses, we collect network flow record data for all IP addresses in the netblock. In our experiments, we collected netflow data for all IP addresses in 95 such IP netblocks, of sizes ranging from /23 to /17, for a period of 1 month. In addition, we also collected data for 256 IP addresses each immediately before and immediately after each of these 95 netblocks. This was done so that we can later test whether we are able to detect the known start and end boundaries of the netblock, apart from the internal boundaries inside it. Data captured for each flow includes source and destination IP addresses, source and destination ports, the number of bytes, and packets, and the protocol. It does not include payloads. The data is collected from peering points between a tier-1 backbone and its peers.

4.2 Behavioral profile/features per IP address

From the above netflow data per subnet, we extract 92 features per IP address that aim to summarize the traffic characteristics of each IP addresses in that subnet. All flows for a given monitored IP address X, are aggregated over the course of a day, and then the following 3 types of features are derived from each day's worth of data:

1. Features based on **counts**: Number of bytes, packets and flows associated with X.

2. Features based on the **IP addresses** that X communicates with: The arithmetic mean of these IP addresses (treating IP addresses as integers), the entropy of the distribution of these IP addresses, the standard deviation of this distribution, and the unique number of such IP addresses.

3. Features based on the **ports** used by an IP address: The most frequently used port, the entropy of the port number distribution, and the number of unique ports used.

For each type of feature above, we distinguish between TCP, UDP and ICMP protocols. For TCP flows, we distinguish between completed TCP flows (which represent real communications) and incomplete TCP flows (e.g. scan attempts). We also distinguish between whether the flow is outgoing or incoming based on whether the source IP or destination IP is in the corresponding network. Finally, for features based on ports, we distinguish between the remote port number and the local port number. This provides 92 real and categorial features for each IP address monitored, per day. Real features are then averaged across days, over a period of 1 month, to give mean values. The only categorical features are the ones that capture the most used port for each setting of the other variables. For these features, we compute the dominant/most-frequent port across the entire month. At the end, we have a rich dataset with 92 features per IP address, that represent its long-term behavior. Note that given a netblock, we obviously do not necessarily observe all netflow data going to and from it. However, this is not a hinderance because we are only interested in relative changes in the behavior of IP addresses inside the netblock.

4.3 Feature quantization

We wish to use the feature representation of each IP address to discover internal boundaries in subnets that contain one or more bad IP addresses. However, three challenges confront our using these features directly as the profile of an IP address.

- **Varying range:** The values in the data span varying and large ranges. For example, flow counts could be unobserved for some IP address, and in the millions for some other IP address

- **Heterogeneity:** The data are a mix of real-valued and categorical features. Examples of real-valued features are mean counts per day, flow distribution entropies etc. The categorical variables are the dominant ports for various types of flows (TCP/ UDP, incoming/outgoing, remote/local etc.)

- **Missing data:** There are a fair number of unobserved/missing values. This occurs often, as a given IP address may not have any flows of a certain type, say incoming UDP flows, for the entire duration of data collection.

We deal with all three of these challenges by transforming the data into a categorical dataset using an adaptive quantization scheme. This is similar in spirit to the transformation used in other IP-address clustering work in the literature [1]. We transform the data from each netblock individually. Within any dataset, we also transform each variable $v_j, j = 1, \ldots, 92$ independently. Our quantization scheme is based on first sorting all the values v_j takes and creating 50 bins based on 50 equally spaced percentiles: $2, 4, \ldots, 98, 100$. By construction, our bins have roughly equal number of points in them. To deal with the special case that the range of v_j we observe in the subnet dataset encompasses less than 50 distinct values, we employ dithering, a well-known method to randomize quantization error, by adding a very small amount of random noise to the feature values. This adaptive transformation to purely categorical data, addresses the large-range issue and issue of having a mixture of categorical and real values data. It also allows us a straightforward route to handle missing information in a reasonable manner. In this case, we simply encode the absence of information on a particular feature by a distinct 51^{st} bin. Note that this missing data/51^{st} bin is not balanced like the other bins, i.e., there is no guarantee that the number of points falling in the bin is equal to the number of points in the other bins.

5. A CLUSTERING-BASED APPROACH TO INFER INTERNAL BOUNDARIES

Given the categorical netblock data, we now turn to the task of generating clusters of IP addresses. While clustering itself is an extremely well studied problem with a vast literature [3, 2], because we are dealing with IP addresses, there is a strong structural constraint on the clustering results that we must respect, namely, we should only allow clusters to consist of contiguous regions of IP address space.

A second constraint that we impose is that our clustering procedure be computationally efficient, due to the ambitiousness of the task (clustering a large fraction of IP address space). Note that this constraint isn't a strict requirement, unlike the contiguous IPs in a cluster constraint, which is.

While seemingly innocuous, these additional constraints (mainly the contiguity requirement) rule out the application of many standard clustering algorithms like k-means, hierarchical agglomerative clustering etc. Indeed, the clustering problem simplifies with the structural constraint. While still large, the number of contiguous partitions of an ordered set of n items (our case) is much smaller than the number of unrestricted partitions of the same n items (generic clustering case).

5.1 Breakpoints

Partitioning discrete IP address space into contiguous clusters is equivalent to the task of finding cluster boundaries/ breakpoints in IP address space. For computational speed we propose the following one-pass breakpoint identification scheme: for each potential breakpoint i, between successive IP addresses, we compute a "breakpoint score", $\sigma(i)$ that evaluates how different the IP addresses in a window on the left are, compared to a equal-sized window of IP addresses on the right of the potential breakpoint (call the window size parameter $|\omega|$, details on how we compute this score follow). Intuitively, true breakpoints/cluster boundaries would have high breakpoint scores, because the behavioral profile of IP addresses in the left window/cluster would be different from the behavioral profile of IP addresses in the right window. Thresholding the breakpoint scores results in a list of candidate breakpoint IP addresses with corresponding breakpoint scores (call this scalar threshold parameter τ).

Of course, breakpoint scores would also be high for potential breakpoints that are just to the left or right of a true breakpoint (because a large number of points in the left and right window would have very different behavioral profiles). For this reason, to obtain the final clustering, we further apply a greedy breakpoint filtering scheme. In our filtering scheme, we iteratively pick the highest scoring breakpoint, and then filter/remove from the list the candidate breakpoints that lie in a contiguous window of IP addresses around the chosen breakpoint. For parsimony of parameters, we set the same window size $|\omega|$ in the filtering step as that used in the breakpoint score evaluation. We end when our candidate IP address list (list of all IP addresses that have breakpoint scores greater than our threshold parameter) is exhausted.

5.2 Summarizing windows of observations

Continuing on the path to computing breakpoint scores, recall that after our adaptive quantization transformation, our data are represented in terms of categorical features, which are challenging to work with when computing distances between pairs of observations. While numerous scores have been proposed in the literature to calculate distances between categorical valued observations [4], our problem is slightly complicated by the fact that we want to compute distances not just between pairs of observations, but sets of observations (the set of observations in the window to the left vs. the set in the window to the right of a potential breakpoint).

To tackle this challenge, we take inspiration from probabilistic clustering methodology to define our breakpoint score. Briefly, consider the the left window of observations. For any one categorical feature, we now have $|\omega|$ instances of this feature, corresponding to the values each of the observations takes. If we imagine a Multinomial distribution generates the set of observed values, it is straightforward to estimate the parameters of this distribution using maximum likelihood, and these component probabilities will be proportional to the frequency of the counts of the observed category values.

This is essentially what we compute, except with one minor twist, in that we also additionally smooth the parameter estimates by adding a small probability to all category values and normalizing the Multinomial parameters appropriately (Laplace smoothing, [5]). This smoothing helps us avoid zero probability estimates for categories not observed in the window of observations, and can be viewed as the expected value of a Bayesian posterior with combining a Dirichlet prior and the Multinomial observations we have.

5.3 Computing breakpoint scores

To make the score computation precise we introduce some notation. Let the features for the ith IP address in a netblock be referred to by $\mathbf{x}^i$, with the superscript ranging over the (say) M IP addresses in that netblock, i.e., $i = 1, \ldots, M$. As mentioned, we compute $N = 92$ categorical features per IP address/observation, so that the vector $\mathbf{x}^i = x^i_{j=1,\ldots,N}$. Further, since each x^i_j is a value taken on by a categori-

cal variable with 51 possible values, conceptually, we can also consider a sparse vector representation of x_j^i, which has $K = 51$ components, all zero, apart from the single component indicating the category value, for example, $\mathsf{x}_j^i = [0, \ldots, 1, \ldots, 0]$. Note in this view of the data, every observation $\mathbf{x}^i$, is now a (sparse) matrix of size $N \times K$, rather than a vector of size N. We will also denote the ordered index set of IP addresses in a window of size $|\omega|$ to the left of (and including) the IP address i by $\overleftarrow{\omega i} = \{i - |\omega|, \ldots, i\}$, and analogously the index set of the IP addresses in a $|\omega|$-sized window to the right by $\overrightarrow{\omega i} = \{i + 1, \ldots, i + |\omega|\}$.

With these definitions, consider any IP address i in a netblock (with M IP addresses in it). For any one of the categorical features j, the set of observations in an ω-window to the left has a summary count statistics vector $\mathbf{c}_j^{\overleftarrow{\omega i}} = [c_{j1}, \ldots, c_{jK}]$, where $c_{jk} = \sum_{\{z \in \overleftarrow{\omega i}\}} \mathsf{x}_j^z$. In words, $\mathbf{c}_j^{\overleftarrow{\omega i}}$ is just the vector of counts of the number of occurrences of the various categories for feature j in the left window observation set. Analogously, we can compute the right window count statistics vector: $\mathbf{c}_j^{\overrightarrow{\omega i}}$.

Assuming a small positive Laplace fraction ϵ, our smoothed Multinomial window distrbution $\mathbf{p}_j^{\overleftarrow{\omega i}}$ is a K-dimensional vector whose every entry is computed via:

$$\mathbf{p}_{jk}^{\overleftarrow{\omega i}} = \frac{c_{jk} + \epsilon}{(|\omega| + K * \epsilon)}.$$

An analogous expression for $\mathbf{p}_j^{\overrightarrow{\omega i}}$ results from using $\mathbf{c}_j^{\overrightarrow{\omega i}}$. Now that we have our final K-dimensional Multinomial window distributions for $\overleftarrow{\omega i}$ and $\overrightarrow{\omega i}$, we compute the Jensen-Shannon divergence between these distributions for each feature. The Jensen-Shannon divergence is a symmetric measure of similarity between two probability distributions P and Q defined as:

$$JS(P||Q) = H(X) - H(X|Z),$$

where H is the entropy, X is a random variable coming from the mixture distribution $0.5 * (P + Q)$, and Z is a binary indicator variable such that $Z = 1$ when X is from P and $Z = 0$ when X is from Q. Thus, for any breakpoint IP address i, for each feature j, we compute

$$s_{ij} = JS(\mathbf{p}_j^{\overleftarrow{\omega i}} || \mathbf{p}_j^{\overrightarrow{\omega i}}).$$

Finally, our breakpoint score $\sigma(i)$, between the left window ending at the ith IP address in the netblock, and the corresponding right window, is the just the sum of these N component divergences, so that $\sigma(i) = \sum_{j=1}^{N} s_{ij}$.

5.4 Tuning Clustering Hyperparameters

Now that our clustering methodology is defined, our remaining task is choosing the clustering procedure hyperpameters, namely the threshold τ and the window size parameter $|\omega|$. Recall that in line with our hypothesis about security risks of bad IP addresses being shared by member in the same administrative block, our objective is to capture intra-netblock boundaries accurately.

Unfortunately, we do not have ground-truth information on the internal boundaries of our netblock data. Therefore, we create a synthetic dataset using portions of our original dataset on 95 netblocks. By doing so, we create specific cluster boundaries. We then choose our clustering algorithm hyperparameters such that our performance on detecting these known boundaries is maximal. In order to measure performance of our clustering technique, we use the variation of information (VI) statistic [6], which is based on an information theoretic view of the degree of alignment between two clusterings. VI between two clusterings $\mathcal{C}_1$ and $\mathcal{C}_2$ is defined as,

$$VI(\mathcal{C}_1, \mathcal{C}_2) = H(\mathcal{C}_1) + H(\mathcal{C}_2) - 2I(\mathcal{C}_1, \mathcal{C}_2).$$

As before, H is discrete entropy, and additionally, I is the mutual information. $\mathcal{C}_i$ is a clustering/partition of a dataset into non-overlapping clusters and the entropy for a partition is defined in the usual way, and depends upon the relative evenness of the sizes of the clusters. Similarly, the mutual information depends upon the degree of overlap between clusters from two different clusterings. These expressions can be found in [7], which also shows a number of nice theoretical and practical properties associated with VI.

Mimicking our original dataset of 95 netblocks and 92 features per IP in each netblock, we generate 100 synthetic datasets which are spliced together from various portions of the original dataset. The splicing process creates artificial boundaries that we are aware of. It is these boundaries, created by us, that we seek via clustering. We assemble each of the 100 synthetic datasets by the following process:

1. Pick a uniformly random number N_1 of segments of IP addresses, between 3 and 10, both inclusive.
2. For constructing each of the N_1 segments:
   (a) First pick one out of the 95 netblocks we collected data for, uniformly at random.
   (b) Given one of the 95 netblocks, randomly pick one of the following 4 contiguous portions of length 256 IP addresses: the 256 IPs just before the start of the netblock (recall that in addition to each netblock, we also collected data for 256 IPs immediately before and 256 IPs after the netblock), the first 256 IPs of the netblock, the last 256 IPs of the netblock and the 256 IPs just after the netblock.
   (c) Given a portion of length 256 IP addresses, pick a uniformly random number N_2 between 128 and 256, and keep the first N_2 IP addresses from the portion picked above.
3. Concatenate the N_1 segments selected by (2) above.

We next split the 100 synthetic datasets into a training and test portion (with 67 randomly chosen training datasets and the remaining 33 as test datasets). In order to get a handle on our cluster hyperparameters, we ran a grid search on combinations of τ and $|\omega|$, seeking to minimize average training VI. This search revealed $\tau = 15$ and $|\omega| = 101$ as the best hyper parameters. Indeed, the clustering performance on our synthetic data is excellent, with mean test VI being 0.16. See Figure 6 for results on three test datasets, one where the clustering is almost perfect (top), one where the clustering performance is average (middle sub plot), and one where the performance is poor (lowest sub plot). As can be seen in the bottom figure, even when our clustering disagrees with the ground truth, it doesn't do so egregiously, and the clusters look extremely reasonable.

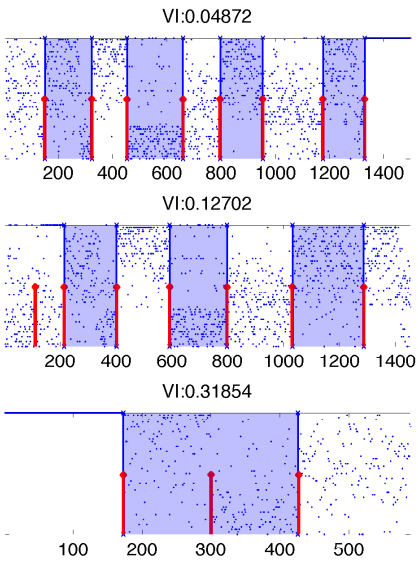


Figure 6: Plot showing results of our clustering algorithm on the assembled dataset. The plot shows three synthetic datasets. In each, the blue lines and alternating blue shaded regions show the ground truth clustering. The thick red lines show where our clustering algorithm places boundaries. The VI between the two clusterings can be found above each plot. Points in the background of the plot show the feature values for one quantized feature.

6. EVALUATION

In this section, we report our results from experiments on predicting malicious IP addresses via inferring internal boundaries. We first describe the malicious IP address data that we use in the evaluation.

6.1 Blacklist Data

We collected lists of malicious IP addresses from two separate sources (Table 1). Spamhaus is a major provider of blacklists, directed towards mitigating the spread of email spam and malware via emails. Given an IP address, it is possible to query Spamhaus, which replies with a code specifying whether the queried IP addresses is malicious (and why) or not. While we cannot acquire a copy of Spamhaus' malicious IP address database, we queried the Spamhaus database for a period of 4 months, using all IP addresses that we observed in raw netflow data crossing our peering points, in several hours worth of traffic. The BotnetsC2 list is a much smaller, manually curated list of confirmed botnet command & control servers. Table 1 gives the details on how long a period of time each blacklist corresponds to, the number of IP addresses in each list, and the number of netblocks, among our 95 netblocks, that contain more than 1 malicious IP address from each list.

With the malicious IP address data defined we now turn to the experimental protocol and evaluation metrics.

Blacklist	Duration	Size	Netblocks w/ > 1 bad IP
Spamhaus	4 months	73969	29
BotnetsC2	2 years	4031	23

Table 1: Two blacklists we use for evaluation.

6.2 Experimental Protocol

At a high level, our experiments will involve: (1). clustering the 95 netblock datasets. (2). evaluating the utility of these inferred clusters (the metrics we use are described shortly) in repeated experiments. In each experiment, given a netblock with $B > 1$ malicious IP addresses appearing in one of the blacklists, we leave one of the known malicious IP addresses out, and evaluate whether our clusters and the remaining $B-1$ malicious IP addresses provide us any predictive power to infer the maliciousness of this left-out IP address. This experiment is then repeated for all B choices of the held-out IP address.

The two metrics we track are:

Detection rate: We consider a value 1 for a detection if in our held-out IP address lies in one of the clusters already containing one or more of the $B-1$ malicious IP addresses. Otherwise this value is 0. We average this binary value over all selections of the held-out IP address, and further over over all netblocks containing more than one malicious IP address to obtain the detection rate. Clearly, a high detection rate (as close as possible to 1) is desirable.

Backlisted fraction: Measures the amount of IP address space we "blacklist", as a fraction of total IP address space. In particular, blacklisting fraction for any one netblock and held out IP address (one experiment) is the ratio of the size of the union of the clusters that contain the $B-1$ malicious IP addresses to the total size of that netblock. This quantity, averaged over all settings of $B-1$ IP addresses, gives the mean blacklisting rate for that netblock, and this, when averaged over all netblocks, gives the overall Blacklisted fraction. A low blacklisted fraction is preferable to a high rate (less traffic will be subject to scrutiny).

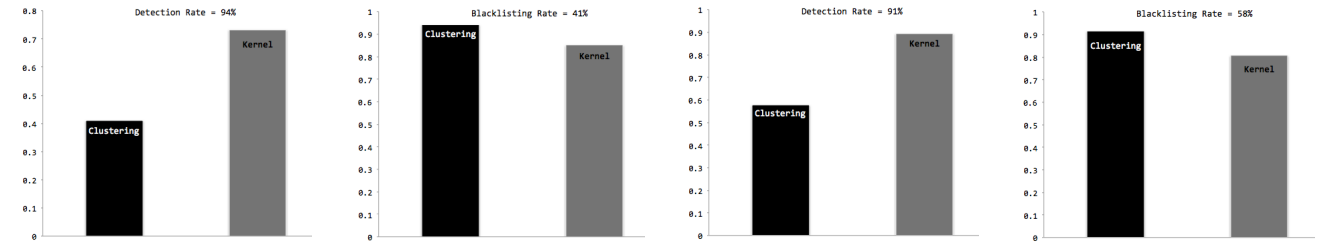
Note that it is easy to have a very high detection rate at the cost of a high blacklisted fraction. For example, by declaring an entire netblock to be a single cluster, one can get a detection rate of 1, but that will also produce a blocking rate of 1. With the metrics and protocol settled, we next turn to the results we obtain on our experiments.

6.3 Results

We first evaluate the detection and blacklisting behavior of our clustering scheme, with clustering hyperparameters set as: threshold $\tau = 15$ and the window size $|\omega| = 101$. Recall that these were the values of hyper parameters that we found via a grid search for minimal VI in Section 5. The following table shows the resulting detection rates and blacklisted fractions:

Blacklist	Detection Rate	Blacklisted Fraction
Spamhaus	0.83	0.27
BotnetsC2	0.61	0.13

While these are reasonable results, there are at least two issues with using these particular hyperparameter values in this set of security related experiments. Firstly, we used the



(a) Spamhaus, Blacklisted fraction (b) Spamhaus, Detection rate (c) botnetsC2, Blacklisted fraction (d) botnetsC2, Detection rate

Figure 7: Comparing our clustering with the kernel method (a) holding the detection rate const, and comparing the blacklisting rate, in the SpamHaus list and (b) holding the blacklisting ratio constant and comparing detection rate, on the SpamHaus list. (c)-(d): Similar results on the BotnetsC2 list.

.

synthetic datasets to estimate these clustering hyper parameters, and thus cannot be sure they apply directly to the real netblock data. Secondly, and more importantly, the detection rates are not particularly high. In our security problem setting, we wish to be able to capture most of the malicious IP addresses, even if it might mean tolerating a higher blacklisted fraction. This is natural given the asymmetric nature of the costs involved in security: additional scrutiny is an impediment, allowing malicious traffic is a disaster.

With this in mind, we re-cluster our original 95 netblock datasets with parameters which would produce higher detection rates (and hence have fewer boundaries, and larger clusters on average). With threshold $\tau = 15$, we now obtain the following performance on the Spamhaus and botnetsC2 blacklists:

Blacklist	Detection Rate	BL Rate	Windowsize
Spamhaus	0.94	0.41	151
BotnetsC2	0.92	0.58	201

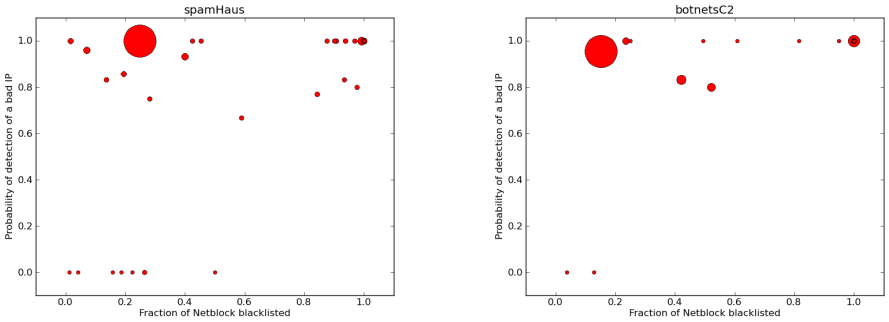
Using these values of the window size parameter $|\omega|$, we do a further comparison of our clustering algorithm's performance with another prediction method, which we will call the *Kernel* method. The Kernel method is rather intuitive and powerful: imagine that given $B-1$ malicious IP addresses in a given netblock, we are asked, as before, to predict where the B^{th} IP address will lie. This time, we place a window of K IP addresses, centered around each of the known malicious IP addresses, and blacklist any IP address that lies in such a window. We then use these blacklisted IP addresses as a predictor of the B^{th} address. As before, we can define a detection rate and a blacklisting rate for the Kernel method. It is obvious that if the kernel K is large enough, the blacklisting rate and detection rate would both be 1.0.

In Figure 7, we compare the performance of our clustering algorithm with the Kernel method, on both, the Spamhaus list and the botnetC2 list. First, we hold the detection rate constant for each method and compare the blacklisting rate (lower is better). Then we hold the blacklisting rate constant for the two schemes and compare the detection rate (higher is better). The results show a marked improvement in detection rate by our technique over the Kernel method with **an up to 13.7% increase** for the same blacklist rate (for the botnetsC2 list). On the flip side we see that for the same detection rate, our clustering scheme can blacklist much less IP address space, **up to 44%** (for the Spamhaus list), tremendously reducing the burden of scrutiny. Every difference between the clustering and kernel outcomes is statistically significant (using a paired sample t-test and a 5% significance level).

Finally, we study the performance of our clustering algorithm on each of the datasets separately. Figure 8(a) and (b) show the performance on a netblock-by-netblock basis on the Spamhaus and the botnetsC2 lists repectively. Figures 8 (a) and (b) convey that if a netblock has poor detection rate performance, it has very few known malicious IP addresses in it. In fact, we verified that all netblocks that have 0% detection rate (appearing along the bottom of the figure) are netblocks with 2, and sometimes 3 malicious IP addresses. This means that we are attempting to predict a malicious IP addresses using just one, and sometimes 2, IP addresses. Note that netblocks with just 2 or 3 known malicious IP addresses have a high chance of truly lying in different clusters, which in our leave-one-out evaluation scenario would always result in poor detection (knowing one malicious IP address gives you no information about the other if they are in separate clusters). Netblocks with larger number of malicious IP addresses (larger bubbles in Figure 8) are less likely to all lie in separate clusters, and always have much higher detection rates in our evaluation.

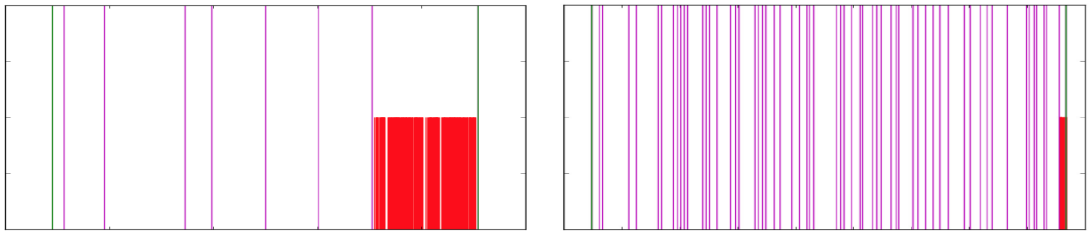
In Figure 9, we illustrate a couple of examples of net blocks with internal boundaries inferred by us along with the locations of known malicious IP addresses. In each of these cases, the malicious IP addresses are well contained within an internal cluster.

We end this section with a few facts about some of the challenges in an evaluation like ours. In reality, we would like to know whether other IP addresses in the same cluster as a known malicious IP address are also vulnerable to the same issue that caused the known IP address to turn malicious. The evaluation we have carried out using blacklists of known malicious IP addresses is really a proxy, because: (i) it doesn't capture the fact that all IP addresses in a cluster may share the same vulnerability but only a few of them are blacklisted. Therefore, we may be accurately inferring cluster boundaries despite low detection rates, (ii) it doesn't capture the fact that known malicious IP addresses may truly lie in different inner clusters, and when this happens, we do not have a way of making a prediction. That said, we believe the results we show are very promising, and point



(a) Spamhaus (b) botnetsC2

Figure 8: Performance of the clustering algorithm on individual subnets in predicting malicious IP addresses in the (a) Spamhaus list and (b) botnetC2 list. Size of a point is proportional to the number of malicious IP addresses known to be in the subnet.



(a) A /21 netblock with 263 bad IP addresses (b) A /18 netblock with 18 bad IP addresses

Figure 9: Two netblocks for which we inferred internal boundaries. The x-axis denotes IPv4 space. Vertical magenta lines indicate the locations of inferred boundaries. Red lines of half the height indicate the locations of known malicious IP addresses.

towards directions to make inroads towards solving the more difficult problem.

7. CONCLUSIONS & FUTURE WORK

In this work we present a new paradigm where we gauge the likelihood of outbound flows initiated from within protected networks being malicious, in contrast to traditional network perimeter defenses. Our approach is based on the intuition that, there are contiguous blocks of IP addresses (subnets) of IPv4 address space which are used and administered in a similar way, hence share similar security properties and vulnerabilities. Therefore, if one IP address is compromised and known to be malicious (i.e. blackisted), then other IP addresses within the same subnet might have been compromised as well. Consequently, any network flow destined to that subnet should be blocked or handled with caution. The critical component of our approach is to infer the boundaries of such subnets on the IPv4 address space. In this work we present a custom tailored clustering method, which attempts to infer such boundaries from IP traffic characteristics observed at the backbone of a Tier-1 ISP. In our experiments, we demonstrate that the proposed clustering method can accurately detect subnet boundaries and effectively predict the blacklisted IP addresses.

Accurately inferring subnet boundaries is central to our approach and there are few avenues that can be pursued to improve this task. For instance, our current clustering method relies on proper selection of threshold and window size parameters, which may not be trivial in practice. One way to address this is to utilize recent non-parametric Bayesian clustering methods which attempt to find a clustering that explains the observed data best. However, these Bayesian clustering methods are known to be very computationally expensive, and may not trivially scale up to very large datasets. We leave this exploration as future work.

Another important issue that requires further exploration is whether two IP addresses in two different subnets within a same publicly known netblock share similar security properties if the two subnets exhibit similar traffic characteristics in the feature space. Could they be two non-adjacent inner subnets used and administered in a similar way? We leave this as future work as well.

Finally, our ultimate goal is to design a system which can make accurate decisions on outbound network connections initiated from within a network. To achieve that, for each outbound flow, we envision to combine the suspicion level of its destination IP address obtained from this work with an anomaly detection scheme which specifies how surprising to have an outbound flow destined to that particular IP address. We believe such multi-layered decision mechanism would yield accurate decisions.

8. REFERENCES

[1] S. Coull, F. Monrose, and M. Bailey. On Measuring the Similarity of Network Hosts: Pitfalls, New Metrics, and Empirical Analyses. In Proceedings of

the 18th Annual Network and Distributed Systems Security Symposium, February, 2011.

[2] A. K. Jain, "Data Clustering: 50 Years Beyond K-Means" , Pattern Recognition Letters, Vol. 31, No. 8, pp. 651-666, 2010.

[3] A. K. Jain, M.N. Murthy and P.J. Flynn, Data Clustering: A Review, ACM Computing Reviews, Nov 1999.

[4] Shyam Boriah, Varun Chandola, Vipin Kumar, "imilarity measures for categorical data: A comparative evaluation", In Proceedings of the eighth SIAM International Conference on Data Mining

[5] SF Chen, J Goodman (1996). "An empirical study of smoothing techniques for language modeling". Proceedings of the 34th annual meeting on Association for Computational Linguistics.

[6] Meila, Marina, "Comparing Clusterings by the Variation of Information". Learning Theory and Kernel Machines, 2003, pp 173-187.

[7] Meila, Marina, "Comparing clusterings – an axiomatic view", Int. Conf. on Machine Learning, 2005.

[8] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. 2010. Building a dynamic reputation system for DNS. In Proceedings of the 19th USENIX conference on Security (USENIX Security'10). USENIX Association, Berkeley, CA, USA, 18-18.

[9] Yuanchen He, Zhenyu Zhong, Sven Krasser, Yuchun Tang, "Mining DNS for Malicious Domain Registrations," Proc. of The 6th International Conference on Collaborative Computing (CollaborateCom 2010), Chicago, 2010.

[10] S. Sinha, M. Bailey, and F. Jahanian. Shades of grey: On the effectiveness of reputation-based blacklists. In 3rd International Conference on MALWARE, 2008.

[11] J. Zhang, P. Porra, and J. Ullrich. Highly predictive blacklisting. In Proceedings of the USENIX Security Symposium, 2008.

[12] D. Anderson, C. Fleizach, S. Savage, and G. Voelker. Spamscatter: Characterizing internet scam hosting infrastructure. In Proceedings of the USENIX Security Symposium, 2007

[13] S. Hao, N. Syed, N. Feamster, A. Gray and S. Krasser. Detecting spammers with SNARE: Spatiotemporal network-level automatic reputation engine. In Proceedings of the USENIX Security Symposium, 2009

[14] Exploiting Network Structure for Proactive Spam Mitigation Shobha Venkataraman, Subhabrata Sen, Oliver Spatscheck, Patrick Haffner and Dawn Song In Usenix Security 2007, August 2007

[15] Tracking Dynamic Sources of Malicious Activity a Internet-Scale, NIPS 2009, Shobha Venkataraman, Avrim Blum, Dawn Song, Subhabrata Sen, Oliver Spatscheck

[16] Brett Stone-gross , Christopher Kruegel , Kevin Almeroth , Andreas Moser , Engin Kirda, FIRE: FInding Rogue nEtworks, ACSAC 2009 Proceedings of the 2009 Annual Computer Security Applications Conference.

[17] M. Collins, T. Shimeall, S. Faber, J. Janies, R. Weaver, and M. D. Shon. Using Uncleanliness to Predict Future Botnet Addresses. In ACM Internet Measurement Conference (IMC), 2007.