Comparison of IP reputation aggregation features: BGP prefix vs. Fixed prefix

Ali Davanian

July 13, 2017

Contents

Abstract

Chapter 1

Introduction

IP reputation is often one of the elements in a vector of features used for malice detection. [1] uses IP reputation, among other features, for spam detection. [2] uses IP reputation in a vector of features to detect malicious DNS addresses. The reputation of an IP address is usually computed based on its appearance in public black lists. Since reputation by definition depends on the historical records, it has limited capability in malice detection.

Single IP address reputation has several limitations. Firstly, due to short life of IP addresses [3, 4], the reputation of an Internet host can not always accurately be attributed to the malicious activity of the corresponding IP address. Secondly, an IP reputation database needs 4GB of storage space (based on IP address space size) to store the reputation of every IP address. Last but not the least, single IP address reputation has poor prediction capability, or in other words, high False Negative (FN) rate [5].

The above limitations of single IP address reputation encourage the usage of aggregated reputation of Internet hosts. In aggregation of the reputation, the reputation of a few Internet hosts is aggregated and then attributed to the group that holds those Internet hosts. For instance, based on the reputation of 1.1.1.1, 1.1.1.2 and 1.1.1.3 IP addresses we can derive the reputation of 1.1.1.0/30 subnet and assume that every traffic from this subnet is malicious. Several studies confirm the effectiveness of the aggregated reputation [6, 7, 1, 8].

The aggregation of Internet host reputation is conducted based on a feature, hereafter as aggregation feature. Aggregation feature can be based on Border Gateway Protocol (BGP) prefix[7, 8], fixed /24 prefix [1, 9], DNS [2, 8], autonomous system number (ASN) [10], geographical location [11] and organization name [12]. These aggregation features address the short life problem of the IP addresses, the reputation database size and FN rate problems with different tradeoffs. For instance, fixed prefix is more granular than geographical location because the latter is often as granular as city and not more. On the other hand, granularity entails storing more records and hence larger database size.

In this study, we are interested in an aggregation feature that can have

a better tradeoff for a network appliance. In particular, we have two main requirements that the aggregation feature shall address. Firstly, the aggregation feature must have a small footprint i.e a small reputation database. Secondly, based on the database size, it is desired to have an acceptable detection accuracy performance. For instance, a database size reduction to half shall not result in a detection accuracy performance of less than half.

We compare BGP prefix and fixed prefix aggregation features. These two features are both based on the IP prefix and this entails deriving such features from IP address can be done via masking. We, henceforth, find these two features more related and better comparable. Our comparison is based on the the reputation lookup performance, which depends on the database size, and the detection performance accuracy based on hit rate. In particular, we answer the following questions:

1. How does aggregation based on BGP prefix perform in comparison to fixed prefix in terms of Hit Rate?

2. How does aggregation based on BGP prefix perform in comparison to fixed prefix in terms of database size and lookup time based on Order of the lookup algorithm?

3. Given the tradeoffs(between detection accuracy and detection lookup performance) of BGP and fixed prefix, which aggregation feature should be selected for a network appliance with limited computation resources?

In the rest of this paper, in Section **??**, we first review the related works. In Section **??**, we present our methodology for measurement and our data collection approach. In Section **??**, we explain our experimental results based on our collected data. In Section 6.6, we answer our research questions based on our results. In Section **??**, we conclude our paper and discuss the future works.

Chapter 2

Literature Review

In this section, we review Internet Host Reputation Systems(IHRSs) in state of the art literature based on different characteristics. Internet host reputation systems output the reputation of a host based on its historical activities. The historical malicious activities, in practice, are mainly found in public blacklists based on IP or DNS address. We use the term *reputation oracle* for any medium that state of the art refer to for the malicious activities of Internet hosts in past. The data from the reputation oracle is the input to a *reputation function* in IHRSs. Different works use different reputation functions and algorithms for further processing. Eventually, the *reputation output* of a reputation system can be a trained classifier or a reputation database. The former is usually an engine that takes traffic and outputs malice based on a backend reputation oracle. The latter outputs the identifiers of malicious hosts. Different works use different metrics such as IP or DNS to identify hosts. State of the art reputation systems choose various benchmarks to evaluate their results. Figure 2.1 shows how Internet Host Reputation Systems work in practice. This figure aims to visualize the Internet Host Reputation System's structure and the relevance of our terminology, i.e the characteristics we use to categorize the state of the art, to these systems.

The works that we choose to review in this study are selected in a systematic way. First, we searched in google scholar based on Reputation system, Predictive blacklisting, Network clustering, Proactive spam detection, Internet bad neighborhood and Internet host aggregation keywords. We removed the results older than 10 years unless the work has been a break through. Next, we selected the works that have been cited more than 10 times. Afterwards, we shortly read the papers and selected the most relevant works to our research. In the rest of this section, we compare the literature based on *reputation oracle*, *reputation function*, *reputation output*, *aggregation feature*, and *Benchmarking*.

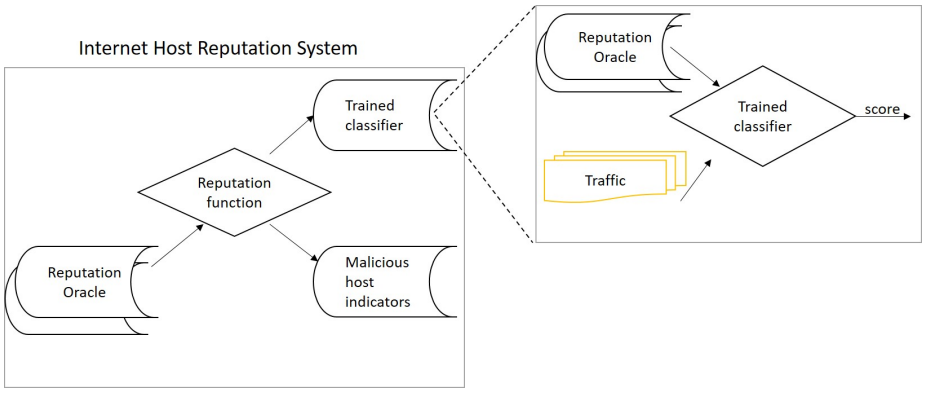


Figure 2.1: Internet host reputation systems

2.1 Reputation Oracle

In Internet host reputation systems, the reputation is measured based on previous activities of hosts. We use the term reputation oracle to refer to a database that stores the reputation of Internet hosts based on their malicious activities in past. A majority of the literature use public blacklist datasets such as SURBL[13], SBL [14], CBL [15], XBL [16], Spamcop [17], Malware Domain List [18], DNS-BH [19], Zeus tracker [20], JWSDB [21], URIBL [22], SORBS [23], DSheild [24], Phishtank [25], hpHosts [26], OpenBL [27], VERIS [28], WHID [29], WPBL [30], Support Intelligence [31], UCEPROTECT[32], APWG [33], Viruswatch [34], Malware Patrol [35] and Bot Command and Control IP addresses from ShadowServer Foundation [36]. A few works use Malware generated traffic in a controlled environment or the traffic to a Honeypot [2][11] or Spamtrap [7][8][37] as their reputation oracle. These works record the hosts that the malicious program tries to connect to. Finally, a few works use the reputation computed by another product, such as IDS, to track the reputation of a host [38].

The nature of the datasets and how data is collected is different. Several datasets are providing the IP and DNS of spammers: SBL [14], CBL [15], [17], JWSDB [21] and SORBS [23]. Some other datasets identify phishing websits: SURBL[13], Phishtank [25] and APWG [33]. Datasets such as Zeus tracker [20] and Bot Command and Control IP addresses from ShadowServer Foundation [36] identify Botnet's command and control servers. DSheild [24] provides the attackers' addresses based on firewall and IDS logs.

Different studies use different datasets based on the aim of the study and the type of malicious hosts the dataset would reveal. Nonetheless, there is a tradeoff between malicious detection capability and detection latency while choosing various datasets. Increasing the dataset size would possibly increase malicious detection performance but at the same time it would increase the processing time. Table 2.1 reports the datasets and approaches that different

Dataset	The literatures that used the dataset as their reputation Oracle								
[1]	[9]	[39]	[2]	[40]	[10]	[12]	[7]	[8]	[37]
SURBL					✓	✓			
SBL			✓		✓	✓		✓	✓
CBL						✓		✓	
XBL					✓				
Spamcop						✓		✓	
Malware Domain List			✓						
Zeus tracker			✓						
JWSDB				✓					
URIBL				✓					
SORBS								✓	
DSheild	✓	✓				✓			
Phishtank					✓	✓			✓
hpHosts						✓			
OpenBL						✓			✓
UCEPROTECT						✓			
VERIS						✓			
WHID									
WPBL						✓			
Malware Domains			✓						
Support Intelligence					✓				
APWG					✓				
Viruswatch					✓				
Malware Patrol					✓				
ShadowServer					✓				
Honeypot traffic			✓						
Spamtrap							✓	✓	✓

Table 2.1: Datasets used by literature

works use. We note that few studies use the same datasets.

2.2 Reputation Function

Reputation function computes the reputation of an Internet host based on some historical features of the sender. Many of the state of the art works use classifiers in order to compute the reputation based on a set of features. [1] [8] [7] [37] are simple linear classifiers that label malice based on a malicious activity threshold. The threshold is usually based on a number of historical malicious activities e.g. the number of sent spams, the number of blacklists that list an IP address or DNS, the number of IP addresses in a network that are blacklisted, or spam ratio i.e. the percentage of spams to total sent emails. Several works use sophisticated Machine Leaning (ML) classifiers and label maliciousness based on an extensive set of features [38][2][12]. [38] uses RuleFit classifier and uses 13 network and traffic related features such as message length and time of the day for classification. [2] uses Logit-Boost strategy (LAD) decision tree based on 16 domain related statistical features. These features span from number of related IP addresses and domains to the number of related malicious domains

[1]The authors of the works, in order, are Soldo et al, Zhang et al, Antonakakis et al, Felegyhazi et al, Shue et al, Liu et al, Venkataraman et al, Qian et al, Moura et al

literature	category	algorithm	Number of features
Soldo et. al. [9]	Recommendation	Exponential Weighted Moving Average - Cross Validation (CA) clustering algorithm - K-Nearst neighbors	2
Zhang et. al. [39]	Recommendation	similar to Google's page rank algorithm	1
Hao et. al.[38]	Classification	RuleFit	13
Antonakakis et. al. [2]	Classification	Logit-Boost strategy (LAD	16
Liu et. al. [12]	Classification	Random Forest classifier	256
Felegyhazi et. al.[40]	Clustering	Manual	˜10
Shue et. al. [10]	Classification	linear classifier	1
Venkataraman et. al. [7]	Classification	linear classifier	1
Qian et. al. [8]	Classification	linear classifier	1
Moura et. al. [37]	Classification	linear classifier	1

Table 2.2: Reputation function characterization of state of the art

and lexical features of the domain name. [12] uses Random Forest classifier and a list of 258 features that are mainly related to mismanagement issues such as open resolver in the network or mis-configured HTTPS certificate.

Other works use recommendation ML algorithms to output the reputation of an Internet host [39][9]. These systems aim to build a customized blacklist for a victim. Their input is a matrix that shows which attackers attacked which victims. They aim to find the most relevant attackers to a victim. While [39] only searches a two-dimensional matrix to find the relevance of different attackers to a victim, [9] also takes into account the temporal behavior of the attacks and searches in a 3-dimensional space. Table 2.2 reports the reputation function category, the algorithm and the number of features that different works use.

2.3 Reputation output

Internet Host Reputation Systems (IHRSs) usually output either of two general artifacts: blacklist; or a trained classifier. Regardless, the state of the art may use the blacklist or the classifier in a detection engine later on. For instance, [7, 8] deploy the result of reputation system they develop in SpamAssassin. Many public IHRS and state of the art works output a blacklist. Other IHRSs based on reputation usually output a trained classifier that may be used in a malicious detection engine. Detection engines that solely use blacklists usually have a less detection latency in comparison to the trained classifiers that work with multiple features because the processing would be limited to only querying the reputation of the node from the blacklist. On the other hand, the trained classifier output of an IHRS usually has a better detection performance accuracy; similar to solving a crime case by law enforcement, more evidences facilitate a better decision. In summary, detection engines based on blacklist artifact have lower detection performance accuracy while maintaining shorter detection latency in comparison to trained classifiers.

A blacklist would assign reputation to an Internet host based on its address

output artifact	The literatures that output such artifact										
[2]	[40]	[8]	[37]	[12]	[9]	[39]	[10]	[2]	[38]	[7]	[1]
blacklist	✓	✓	✓		✓	✓	✓			✓	✓
trained classifier				✓				✓	✓		

Table 2.3: Internet host reputation systems taxonomy based on output artifact

whereas a classifier is trained based on many attributes, including the Internet host address. In addition, a compiled blacklist is independent of the final implementation and it can be used in numerous systems e.g. firewall, IDS etc; however, a trained classifier can hardly be used anywhere else except in the designed detection engine.

A majority of state of the art output a reputation blacklist [40, 8, 37, 9, 39, 10, 7, 1]. [2, 38, 12] train a classifier based on a reputation oracle. [2] and [38] use the trained classifier in a detection engine while [12] uses the classifier to predict future cyber incidents. In addition to the previous works, there are few researches that can not be strictly characterized in one category. Table 2.3 reports the characterization of the literature based on the decision time.

2.4 Aggregation feature

Reputation is assigned to an Internet host based on a metric by which the host can be distinguished from the others. We use the general term aggregation feature to such host distinguisher even when there is no aggregation in practice. In such case, we consider aggregation size equal to one. Reputation is commonly assigned to Internet hosts based on *IP address* [37] [9][39][7][8]. Except [39], the rest use an aggregation of IP addresses based on IP prefix. One popular alternative to IP address is *DNS address*. There is usually a mapping between DNS host name and IP address through A record and vice versa through DNS PTR record. Since IP addresses are often dynamically assigned, DNS name sometimes maintains more stability. [2] and [40] use DNS records. [8] uses reverse Domain authoritative reverse DNS server (rANS) and reverse DNS(rDNS) as aggregation feature. In addition to DNS, *Autonomous System Number (ASN)* can be used as an aggregation feature. The mapping between IP address and ASN is not usually intuitive; the routing BGP broadcasts need to be examined for the mapping. [10] and [41] use ASN number as aggregation feature. Finally, some works use Geo location of an Internet host such as country or city to assign reputation [42, 11].

Aggregation metric may represent an individual host or a group. IP address, and DNS A records represent an individual host while the rest of the indicators represent a group of hosts and the assigned reputation applies to all the group's

[2]The authors of the works, in order, are Soldo et al, Felegyhazi et al, Qian et al, Moura et al, Liu et al, Soldo et al, Zhang et al, Shue et al, Antonakakis et al, Hao et al, Venkataraman et al, Wanrooij and Pras

Literature	IP	DNS	ASN	Geo location	Organization	Aggregated	Network-Aware
Soldo et. al.[9]	✓					✓	
Zhang et. al.[39]	✓						
Antonakakis et. al.[2]		✓					
Felegyhazi et. al.[40]		✓					
Shue et. al.[10]			✓			✓	✓
Van Polen et. al.[11]				✓		✓	
Venkataraman et. al.[7]	✓					✓	✓
Qian et. al.[8]	✓					✓	✓
Moura et. al.[37]	✓					✓	
Liu et. al.[12]					✓	✓	✓

Table 2.4: IHRSs taxonomy based on aggregation feature

members. Aggregating the Internet nodes based on a metric may be carried out in different ways. For instance, IP prefix is a common method to aggregate Internet nodes. That said, the IP prefix based on an IP addresses can be calculated either statically and based on a fixed prefix or dynamically and based on BGP prefix. The latter is called a network aware cluster because it represents a network in real world. Several studies suggest that network aware clusters have better granularity, and hence the precision derived from these clusters is better [7, 8, 43]. On the other hand, granularity increases the entries' size that need to be stored, and hence the detection latency may increase.

Table 2.4 reports the result of categorizing the state of the art based on their aggregation feature. We note that systems which output a trained classifier are not mentioned in the table. The reason is that these works, as already mentioned, are taking a holistic approach; they don't use only one metric to track the reputation. For instance, [2] uses both th IP address prefix and the Domain name of a host for training the classifier.

2.5 Benchmarking

The literature use different benchmarking to validate the proposed reputation system. The first category of works use their reputation oracle to validate their results. In other words, they split the data from reputation oracle to two sets: training data set; and the testing data set. The testing data set, used for validation, contains the malicious activity records of later time, hereafter as *future versions* of blacklist. For instance, the records of SBL from the October of one year may be used for detection, and the records of November of the same year may be used for validation. [40] reports false positive and true positive based on presence or absence of the predicted malicious domains in JWSDB, URIBL and McAfee SiteAdvisor. Sophisticated classification solutions follow a similar approach but in a more systematic way. [38, 2, 12] use multi fold cross validation to report false positive and true positive. The second category of works use another well-known malicious detection system to compare their results. [8, 1, 7] use SpamAssassin as the ground truth to evaluate their system. What they report is basically the improvement they obtain by employing their compiled list in SpamAssassin. For instance, Qian et. al. report that their

	Ground truth	Accuracy metric	Measurement method	TP/FP value	compared work
[9]	Reputation Oracle	Hit Rate	experimental analysis	———	[39]
[39]	Reputation Oracle	Hit rate	experimental analysis	———	None
[38]	Reputation Oracle	TP/FP	10-fold cross validation	70%/0.3%	None
[2]	Reputation Oracle	TP/FP	10-fold cross validation	96.8%/0.38%	None
[12]	Reputation Oracle	TP/FP	10-fold cross validation	90%/10%	None
[40]	Reputation Oracle	TP/FP	cross validation	75%/5%	None
[8]	SpamAssassin	FN/FP	experimental analysis	10%/1% [3]	None
[37]	———	information lost - error	mathematical calculation		None
[1]	SpamAssassin	FP/FN	experimental analysis	———	None
[7]	SpamAssassin	server goodput	experimental analysis	———	None

Table 2.5: Benchmarking of the state of the art

aggregation method improves the false positive rate of SpamAssassin by 50%. Venkataraman et. al. report the percentage of the Spams that were detected when the server was overloaded. Such measurements are not comprehensive enough to compare the malicious detection performance of their underlying reputation systems. The third category, Recommendation-based solutions, do not report based on FP and NP; they concentrate on hit count [9][39]. Hit count is calculated based on the number of blacklist entries that are seen in the actual traffic. The reason to choose this measurement is the goal of such works i.e. optimizing the size and effectiveness of the public blacklists. Table 6.1 reports the characterization of the literature based on their benchmarking.

[3]The reported accuracy in this work differs based on threshold and the aggregation feature. Our reported value is an approximation of the optimum balance of FP/NP of DNS and IP BGP prefix clusters

Chapter 3

Problem Formulation and Definitions

The objective of this paper is to compare aggregation of IP reputation based on the BGP announcements and fixed value masking. In order to measure reputation, we collect the malicious activity of the IP addresses; we will explain our data collection approach in the next section. Using the reputation of IP addresses, we compute the probability of having malicious activity from a group of IP addresses. In this section, we formally formulate the terms and research questions. In section 3.1, we define the basic terms that we will use in the rest of this paper. In section **??**, with recourse to the basic definitions, we define several metrics that allow us compare the two aggregation features and answer the research questions we earlier presented.

3.1 Basic definitions

We define two constants BGP and $Fixed$ that will be used as labels and indexes to successively reference to aggregation based on BGP and fixed prefixes. We define a set of IP indicators as:

$$I_{s,t} = \{i_1, i_2, i_3, ..., i_n\}$$

s represents the start date of collecting indicators and t represents the number of days that the collection lasted. I, in simple terms, represents our dataset of indicators. We define an aggregation feature value domain D^f as the the set of all the values that aggregation based on a feature, $F(x)$ can have. We define $|.|$ operator that outputs the size of a set. Formally speaking:

$$D^f = \{a_1, a_2, a_3, ..., a_n\},\ f \in \{BGP, Fixed\},\ |D| = n$$

$$F(x) \in D^f$$

We define the set of all the prefixes derived from an indicator set by $A^f_{s,t}$:

$$a_i \in A^f_{s,t} \; only \; if \; \exists x \in I_{s,t} \; such \; that \; F(x) = a_i$$

A prefix a_i represents a group of IP addresses. We name this set U_{a_i}. The size of U_{a_i}, $|U_{a_i}|$, depends on the length of prefix. For Fixed prefix, the prefix length is always 24, however, for BGP the length can be anything greater or equal to 8 and less than or equal to 24. Given a prefix size l for prefix a:

$$|U_a| = 2^{32-l}$$

Similarly, we define set M_{a_i}:

$$x \in M_{a_i} \; only \; if \; x \in I \; and \; F(x) = a_i$$

We define the probability of having malicious activity from an IP address x by $P(x)$. This probability can be different in different times t since Internet hosts may be infected and cleaned. Henceforth:

$$P(x) \approx \sigma(x, t)$$

$\sigma(x, t)$ is a function that is unknown to us. By aggregation, we generalize the reputation of few hosts in a group to th whole group. Formally speaking, we are modeling $\sigma(t)$:

$$P(x) \approx P_{s,t}(a) \; such \; that \; F(x) = a$$

$P_{s,t}(a)$ is in fact the probability that an IP address from a prefix a is indeed malicious based on our training data collected from s up to t days. For the sake of simplicity, we refer to $P_{s,t}(a)$ by simply using $P(a)$ notation unless we strictly say otherwise. We define score of prefix a as S_a which is simply $|M_a|$, and we define $P(a)$ based on S_a:

$$P(a) = \frac{S_a}{|U_a|}$$

3.1.1 Example

In order to clarify the notations, we give an example using all the notations. We assume our dataset has the following entries and it has been collected from 2017-Mar-18 for 7 days:

$$I_{2017-Mar-18,10} = \{15.14.13.10, 15.14.13.11, 37.3.0.1, 156.147.2.1\}$$

D^{Fixed} in this case is all the following entries:

$$D^{Fixed} = \{0.0.0.0/24, 0.0.1.0/24, ..., 255.255.255.0/24\} \; |D^{Fixed}| = 16777216$$

D^{BGP}, however, depends on the BGP announcements between 2017-Mar-18 and 2017-Mar-25. An interesting reader can refer to ftp://archive.routeviews.org/ in

order to download the appropriate dataset or can use our BGP collector script in the Appendix to generate D^{BGP}. $F(x)$ in case of fixed aggregation is a simple masking with `0xffffff00` value. In case of BGP, $F(x)$ should be implemented by checking the routing snapshots of the given period. Below is the result of aggregation:

$$A^{Fixed}_{2017-Mar-18,10} = \{15.14.13.0/24, 37.3.0.0/24, 156.147.2.0/24\}$$

$$A^{BGP}_{2017-Mar-18,10} = \{15.0.0.0/8, 37.2.0.0/15, 156.147.0.0/16\}$$

$$M_{15.14.13.0/24} = M_{15.0.0.0/8} = \{15.14.13.10, 15.14.13.11\}$$

$$M_{37.3.0.0/24} = M_{37.2.0.0/15} = \{37.3.0.1\}$$

$$M_{156.147.2.0/24} = M_{156.147.0.0/16} = \{156.147.2.1\}$$

Based on the above data, $P(15.14.13.0/24)$ is $\frac{2}{256}$. This probability means that if we observe traffic from any IP address that its masking with `0xffffff00` is 15.14.13.0/24 there is $\frac{2}{256}$ chance that this traffic is malicious. In contrast, with BGP aggregation, the probability of having malicious activity from any equivalent IP address is $\frac{2}{65536}$. The reader should notice that our example is not representative of the real world since we are only considering a dataset of only 4 IP addresses; we only gave this example to clarify the definitions and not to compare Fixed with BGP aggregation.

3.2 Metrics definition

3.2.1 BPL˜

In order to see if BGP aggregation has a significant effect and in order to answer the sub research question 1, we will examine *BGP Prefix Length Distribution (BPL˜)*. If all or majority of prefix population has a length of 24, then further analysis in not required since two aggregation features lead to an almost equal A set. This in turn leads to equal P values and hence two approaches are basically the same.

3.2.2 Precision

In order to measure precision of BGP and fixed aggregation, and answer sub research question 2, we first split our datasets to two set: training; and testing. Our training set spans from date s to $s + t$ and our testing set spans from s' to $s' + t$ such that $s' > s + t$. We, then, measure the probability of malice for the observed prefixes in the training set and the testing sets. Afterwards, we extract $(P^f_{s,t}(a), P^f_{s',t}(a)), f \in \{BGP, Fixed\}$ points from the prefixes $a \in J$:

$$J = A^f_{s,t} \bigcap A^f_{s',t} \quad f \in \{Fixed, BGP\}$$

$P^f_{s,t}(a)$, hereafter as $P(a)$ unless stated otherwise, measures the probability that an IP address from prefix a sends malicious traffic based on our training dataset. $P^f_{s',t}(a)$, hereafter as $P'(a)$ unless stated otherwise, measures the probability that an IP address from prefix a sends malicious traffic based on our testing dataset. In simple words, $P(a)$ is our predicted probability and $P'(a)$ is our observed probability. After constructing J, we plot it to examine the relation between P and P'. We then investigate if there is any correlation through *regression modeling*. We compare the precision of BGP and Fixed aggregation based on the standard error.

3.2.3 Granularity

By definition, fixed aggregation is more granular than BGP; the reputation derived from fixed aggregation is attributed to a smaller group of hosts. In other words, we have more probability records in our database and this can give us a more granular view of the Internet host reputations. Formally speaking:

$$|A^{BGP}_{t,s}| <= |A^{Fixed}_{t,s}|$$

The above relation exists because all advertised prefixes in wild are less than or equal to 24, and all the prefixes in the Fixed approach are exactly 24. Although the fixed aggregation is more granular than BGP, the probabilities distribution in BGP can still neutralize the effect if:

$$P^{Fixed}_{s,t}(a) = P^{BGP}_{s,t}(a) \forall a \in A^{Fixed}_{t,s}$$

However there is not guarantee that all prefixes a exist in $A^{BGP}_{t,s}$. As a matter of fact, with BPL metric we ensure this is not the case; otherwise, comparison of BGP and fixed prefix is meaningless since $A^{Fixed}_{t,s} = A^{BGP}_{t,s}$. Nevertheless, $P^{BGP}_{s,t}(a)$ can still be derived from an a' that encompass the IP addresses that a encompass. This is because, by definition, we generalize the reputation of M_a to $x \in U_a$. Since a' encompasses a, $U_a \subset U'_a$. Henceforth, the reputation of U_a members is represented by the reputation a' and $[P(x)x \in U_a$ is represented by $P(a')$. In order to clarify the concept, assume that we want to lookup the reputation of 9.50.10.1 based on aggregation. In fixed approach, we need to lookup the maliciousness probability of 9.50.10.0\24 in our database. Assume that based on the routing snapshots 9.50.10.1 belongs to 9.50.10.0\23 prefix. In the BGP approach, in order to lookup the probability of 9.50.10.1 we need to look for 9.50.10.0\23 entry. If P^{Fixed}(9.50.10.0\24) = P^{BGP}(9.50.10.0\23), there hasn't been indeed any granularity loss. This can only happen if the adjacent \24 prefixes that comprise a BGP prefix have almost the same probability of maliciousness. For instance, in the former example, if 9.50.10.0\24 and 9.50.11.0\24 have probability y. P^{BGP}(9.50.10.0\23) would also have the probability of y. In order to compare the granularity of BGP and fixed aggregation, and answer research question 3 we investigate if the above phenomenon exists i.e. the adjacent \24 prefixes based on the BGP view have the same

probabilities. To perform such analysis, we define $\Delta(a)$:

$$Delta(a) = P^{Fixed}(a) - P^{BGP}(a)$$

We compute $P^{BGP}(a)$ in the same manner that we explained in the above paragraphs. We then analyze the distribution of *Delta* and analyze its statistical characters.

3.2.4 Detection Rate

In order to compare the detection performance of BGP and fixed aggregation, we employ the detection rate metric. We define the hits as the number of records in $A^f_{s,t}$ that also appear in $A^f_{s',t}$. Based on this definition:

$$D = A^f_{s,t} \bigcap A^f_{s',t} \quad Hit(A^f_{s,t}, A^f_{s',t}) = \sum_{i=1}^{|D|} |M_{a_i}| \;\; \forall a_i \in D$$

Similarly, we define *Detection Rate*:

$$Detection\ Rate = \frac{Hit(A^f_{s,t})}{|A^f_{s',t}|}$$

In simple words, this metric shows how much of the malicious IP addresses our prefix set based on an aggregation can detect. Of course this metric does not say anything about the confidence of the prediction; detection, here, simply means that we can have an estimation on the probability of having malicious traffic from a prefix. In order to better understand the nature ot the detection that a prefix set may have we define Cumulative Distribution Function(CDF) of Hits based on the malice probability x:

$$CDF(x) = \sum_{i=1}^{n} |M'_a| \; \forall a \; s.t. \; P(a) <= x$$

CDF graph gives us an insight about the probabilities that we report for the hits and our confidence about the malice chance. This insight will help the designer of a malicious detection solution to better understand how to employ the aggregated reputation and what to expect from the reputation.

By the detection rate metric, we aim to see the potential of each aggregation feature to identify the entire malicious IP addresses. In this regard, it is expected that BGP identifies more malicious IP addresses since:

$$\forall a \in D^Fixed \; , \forall b \in D^BGP, U_a \subset U_b$$

The above relation holds because the derived BGP prefixes from our IP indicators set are either of size 24 or larger size that encompass the /24 prefix. Since the IP space that a BGP aggregated reputation database covers is always bigger

than its Fixed /24 counterpart, the Detection Rate of BGP is always equal to or greater than /24 Fixed aggregation. Larger IP space coverage, however, has a downside; the chance of having higher false positives increases by an IP space coverage growth. That said, it is the tradeoff between the Detection Rate and the space growth that can justify the usage of BGP. If the Detection Rate of BGP is meaningfully different, we can use our probability metric to signify a chance of malicious activity and expect other malicious detection features to distinguish false positives from the true positives. Otherwise, the Fixed aggregation is preferred since it will have smaller false positive rate.

3.2.5 Lookup performance

In order to compare the lookup performance of BGP and fixed aggregation, we employ two metrics. Firstly, we compare the processing overhead of each aggregation feature based on the lookup algorithm *Order value*. We present two lookup algorithms that are independent of the employed aggregation feature. Yet, since the number of prefixes derived from each aggregation is different, $O(x)$ that is dependent on the number of records can be different. For instance, if the lookup algorithm has $O(n)$ the performance would be two times more if n becomes $\frac{n}{2}$.

Secondly, we compare the *footprint* of each aggregation feature. Again, footprint depends on the implementation of the lookup algorithm and the underlying data structure; we take this into consideration. We report the footprint based on the number of records and the size of the lookup algorithm data structure in Bytes on disk and in memory.

Chapter 4

Data Collection Methodology

In this chapter, we explain what dataset we use for our analysis. Our data comes from Redsocks company that is a cyber security solution provider in Netherlands. The data that they shared with us is used for malicious detection from network traffic. We processed the raw data that we received and a build a dataset of IP prefixes with an assigned score showing the number of malicious IP addresses within that prefix. In the rest of this chapter, in Section 4.1, we explain in detail what our raw dataset from Redsocks Security (RS) contains. In Section 4.2, we discuss the structure of the prefix dataset we would generate from raw data. In Section 4.3, we present our preprocessing steps on the raw data to construct the prefix dataset we will use later for experimental analysis. In section 4.4, we give some statistics about our prepared prefix dataset.

4.1 Redsocks Raw Dataset

Redsocks Security(RS) collects Indicators Of Compromise (IOC) on a daily basis. These IOCs are used for behavioral detection of malicious activity. RS gave us access to this database for a period of 1 month starting from 2017-04-18. Since they store IOCs with a timestamp, we fetched only the IOCs that are collected within the aforementioned period.

4.1.1 Indicators Of Compromise

An indicator Of Compromise (IOC) within the dataset that we had access to has many attributes. Both because of confidentiality and also lack of relevance we do not discuss all the attributes. The relevant attributes to our work are *indicator, source and timestamp*. The indicator is a string value of different length that contains the valuable data to indicate a malicious activity. IP address, DNS name, URL and file hash value, among others, are some types of IOC that one

can expect from the raw dataset we had access to. Source stores the information about the source that the IOC has been collected from e.g. a malware lab. In the next subsection we elaborate on some of the sources. Timestamp says about the date that the IOC is collected. This attribute is important to us since an IOC is not always valid; for instance an infected workstation IP address may be an IOC in the dataset but after the workstation is cleaned the corresponding IP must also be removed.

4.1.2 IOC Sources

Due to business confidentiality, we can not mention all the sources that RS uses to collect IOCs. That said, we mention the tops sources ,based on count analysis, that the majority of the IOCs come from. Our count analysis shows that manual input from RS malware analysts, VirusTotal and two malware labs that RS has are the main providers of IOCs. RS also uses public datasets such as Zeus Tracker and Phishtank. However, the data from such sources is one order of magnitude less than the other sources that we mentioned.

4.1.3 How Big is our raw data?

The number of indicators for Four weeks of data collection that we had are reported in Table 4.1. These values report the indicators from all types e.g. file hash, URL etc. In Section 4.3, we explain how we process these indicators and extract the data we need.

4.2 Prefix Dataset Structure

In order to analyze /24 Fixed aggregation with BGP aggregation, we need to analyze the distribution of IP addresses in their corresponding prefixes. Henceforth, we build a dataset of prefixes from indicators showing the number of malicious indicators (IPs) in each prefix. Each row in our dataset has the following attributes:

Start Date	End Date	Indicators
2017-04-18	2017-04-24	336037
2017-04-25	2017-05-01	349656
2017-05-02	2017-05-08	757027
2017-05-09	2017-05-15	329075

Table 4.1: RS IOC statistics

- *NetID*: a 32 bit integer that is numeric representation of a net block ID. For instance, 188.201.133.0 is represented with 3167323392 value

- *PrefixLength*: Prefix length is the number of bits used to represent the NetID of an IP address. For instance, in CIDR representation 188.201.133.0/24, 24 is the PrefixLength.

- *IndicatorsCount*: This attribute reports the number of indicators that their IP addresses has this NetID.

- *BGPID*: This attribute reports the CIDR that the NetID belongs to. For instance, our previous example belongs to 188.200.0.0/14.

- *timestamp*: Since indicators and BGPIDs are valid only for a period, we store the date that we compile the record.

4.3 Preprocessing

We process the raw data from RS and then compile our prefix dataset. We perform this preprocessing in three steps (see Figure 4.1). Steps in details are:

1. *IP extraction*: Since not all the indicators are IP, we need to extract the IP indicators or try to map an indicator to an IP address. A mapping can be drawn between URL and domain indicators and an IP address. For URLs, we extract the host name part of the URL and then resolve it to an IP address. For domains, we again query it and try to resolve it to an IP address. We note that not all host names are resolvable; in such cases, we discard the indicator. We also note that an IP address may be pointed by multiple indicators i.e. different URLs, domain names etc. In our analysis, such IP addresses are treated same as other IP addresses; considering the weight for such IP addresses and analyzing the effect is beyond the scope of the current work. Finally, we note that not the same IP addresses can be derived at different points of time from the same dataset of indicators; domains and URLs may point to different IP addresses at different points of time. That said, the closest time to the time when an indicator is collected is the best time to query it for the malicious IP address.

2. /24 *prefix mapping*: In our analysis, a /24 prefix is the smallest prefix (from the number of hosts point of view) that an IP address can belong to; for Fixed aggregation, this prefix is indeed the one and for BGP the prefix is either of size 24 or larger. Henceforth, we first mask every IP address with $0xffffff00$ value and derive the prefix it belongs to. If it is not already listed in our dataset, we include that prefix. Otherwise, we increase the IndicatorsCount value by one. The PrefixLength as the name of this step implies is always 24; analysis based on other sizes would change this value.

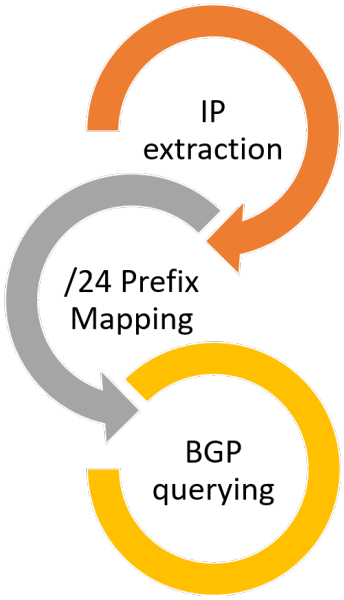


Figure 4.1: Preprocessing

3. *BGP querying*: To compare Fixed aggregation with that of BGP, we need to know how IP addresses are distributed in BGP prefixes. We map each /24 Fixed prefix to a BGP one. The mapping is done using [44]. To have an accurate mapping, we load the relevant BGP announcements according to indicators date from [45] into *pyasn* script. To compute the number of malicious IP addresses in a BGP prefix, we have to aggregate all the /24 prefixes that it encompasses. To expedite analysis, we store the result of such aggregation.

4.4 Generated Prefix Datasets

The result of our data collection and the final dataset that we base our analysis is accessible from [46]. This dataset contains the prefixes that we observe malicious activities from the Indicators of Compromises Redsocks Security shared with us from 2017-04-18 to 2017-05-18. Each BGP and /24 prefix has an assigned score that reflects the number of malicious IP addresses from that prefix. The dataset is in SQL format and it has been implemented in a MySQL database. The data from each week and based on the aggregation feature (fixed or BGP) is reported in a different table. Furthermore, aggregation based on different training lengths (1,2 and 3 weeks) is also reported in a different table.

Table 4.2 reports the data that the database that we will use[46] has. The number of IP addresses, /24 prefixes and BGP prefixes that we observed in each

week is reported in Table 4.3.

Start Date	End Date	name	aggregation feature
2017-04-18	2017-04-24	fixed_week_0418	Fixed
2017-04-18	2017-04-24	bgps_week_0418	BGP
2017-04-25	2017-05-01	fixed_week_0425	Fixed
2017-04-25	2017-05-01	bgps_week_0425	BGP
2017-05-02	2017-05-08	fixed_week_0502	Fixed
2017-05-02	2017-05-08	bgps_week_0502	BGP
2017-05-09	2017-05-15	fixed_week_0509	Fixed
2017-05-09	2017-05-15	bgps_week_0509	BGP
2017-04-18	2017-05-01	fixed_2weeks_20170418	Fixed
2017-04-18	2017-05-01	bgps_2weeks_20170418	BGP
2017-04-18	2017-05-08	fixed_3weeks_20170418	Fixed
2017-04-18	2017-05-08	bgps_3weeks_20170418	BGP

Table 4.2: [46] dataset that we will use for analysis

Start Date	End Date	Unique IP addresses	/24 prefixes	BGP prefixes
2017-04-18	2017-04-24	113357	61222	26197
2017-04-25	2017-05-01	103298	71866	30274
2017-05-02	2017-05-08	154939	83018	34782
2017-05-09	2017-05-15	128475	62557	26768

Table 4.3: Processed IOC and prefixes statistics

Chapter 5

Aggregated Reputation Lookup Implementation

After compiling the reputation of a group of IP addresses, clustered based on a Fixed or BGP size, the database will be used for reputation lookup of single IP addresses. We explain in this chapter that there is a tradeoff between the lookup performance and the footprint of the lookup implementation. Since one of the criteria for our selection of an aggregation feature is the lookup performance and footprint, we need to consider the underlying implementation. Henceforth, in this chapter we give two implementations that work based on a database of prefix reputation.

Our implementations in this chapter are independent of the aggregation feature that we use to build the reputation database. In other words, we assume that the reputation lookup algorithm is unaware of the reputation compilation algorithm. In order to achieve this, we abstractly assume that the database keeps the reputations in a $NetID, PrefixLength, Probability$ tuple format. For instance, 2.16.196.0/23 prefix with a malice probability of 79% is stored as $34653184, 23, 0.79$. Therefor, in the Fixed aggregation all the entries in our database have length /24 while for BGP the length can be anything between 8 and 24,

In the rest of this chapter, in Section 5.1, we present an algorithm that can do the lookup in $O(1)$ regardless of the aggregation compilation feature. In Section 5.2, we present another lookup algorithm that can do the search in $O(log(n))$ but is significantly more efficient from footprint perspective.

5.1 Indexing Search Algorithm

The IP reputation lookup can be implemented in a fast manner via indexing. Since IP space can be presented as a finite series, 2^32 members, we can exploit this feature for indexing. We may use the IP address integer value as a pointer to a memory location. This memory location would store the reputation of

the IP addresses. As prefixes also can be represented by integer, we can split the IP space to the prefixes of same length. Given a prefix length of l, the IP space would be split to 2^l chunks. We assign a reputation to each of these chunks and then later we can query the reputation of an IP address by finding its corresponding chunk. This approach is very similar to Hash Table concept. Following this analogy, the chunks are indeed the defined buckets in a Hash table and finding the bucket of an IP is through a simple division of the IP by 2^{32-l} value.

5.1.1 Data Structure

The implement the O(1) algorithm, we simply use an array (see Figure 5.1). This array has 2^24 entries; we split the IP space to buckets of length 256, or prefixes of size 24. Each element stores the reputation of the corresponding /24 Net block i.e. the start of the array address plus the result of the mask of the IP address with `0xffffff00`. For instance, the reputation of 0.0.5.57 IP is stored in the 6 element of the array. For the /24 entries that we haven't recorded any malicious activity, we store 0. This value is logically correct since based on historical records, there is 0 chance that an IP from this space exposes malicious activity.

5.1.2 Initialization

In order to materialize the Indexing Search algorithm, we first need to initialize a byte array of size 2^24 with the corresponding probabilities of the indexes i.e. /24 prefixes. Since the reputation database may contain larger net block entries than /24, $l < 24$, we first needs to process the reputation database entries and split large net blocks to /24 entries. This processing is done using Algorithm 1. The inputs of this algorithm as mentioned earlier in this chapter come from the aggregated reputation compilation. After that, Algorithm 2 initializes the Indexing array in the memory. We use by default one byte to store the reputation of a net block. In order to realize storing a float as a byte, we multiply the probability by 256. If this precision is not sufficient, the reputation can be stored even as a float; the footprint, however, increases as a

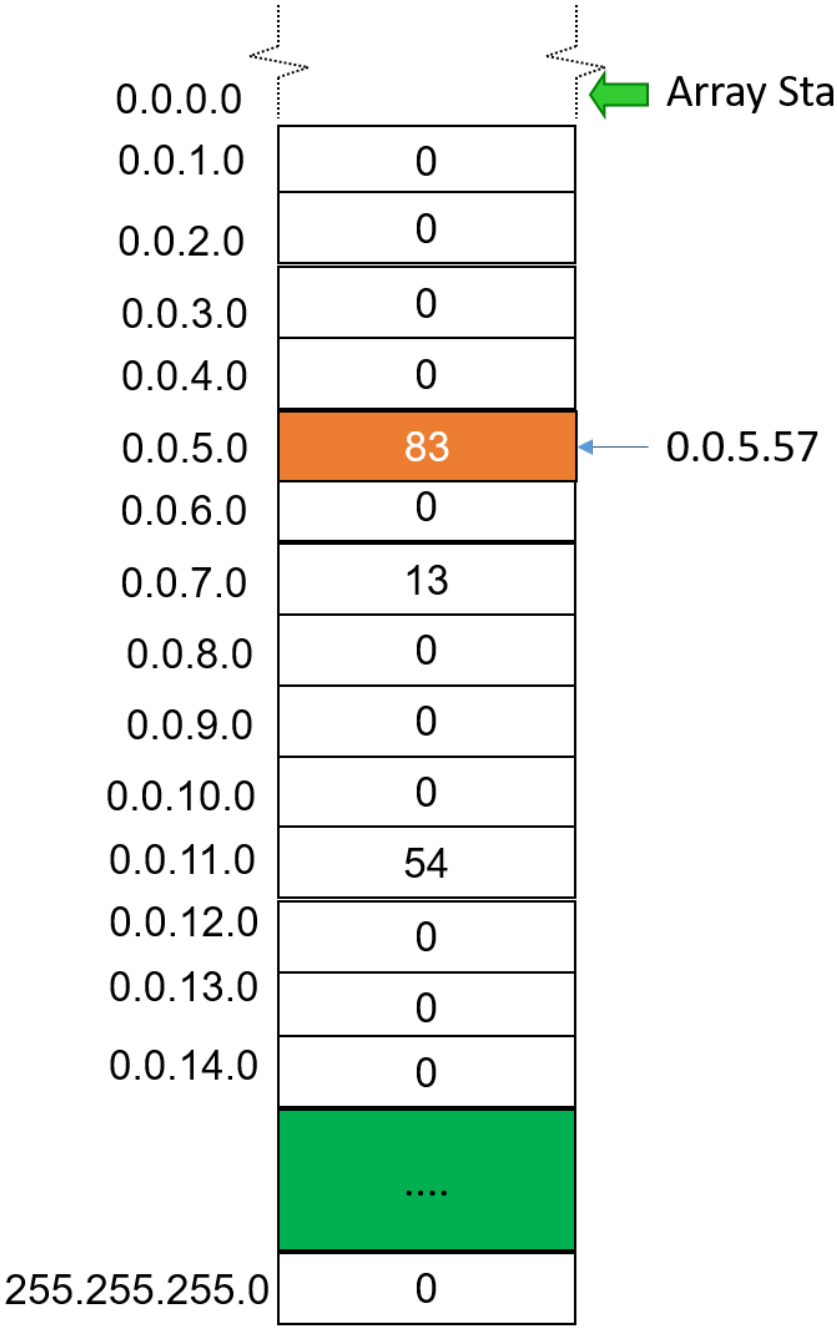


Figure 5.1: Array data structure to use for O(1) algorithm

result.

Algorithm 1: Covert a reputation database with entries' prefix larger than 24 to 24

```
Data: Three arrays of NetIDs, Lengths and Probabilities
Result: Two derived lists of Fixed_NetIDs and Fixed_Probabilities
Fixed_NetIDs:=List();
Fixed_Probabilities:=List();
for i:=1 to len(NetIDs) do
    start_NetID = NetID[i];
    Num__Of_Prefixes = 2^(24−Lengths[i]) ;
    for j:=1 to Num__Of_Prefixes do
        m:= (j-1) * 256;
        Fixed_NetID := start_NetID + m;
        Fixed_NetIDs.append(Fixed_NetID);
        Fixed_Probabilities.append(Probabilities[i]);
    end
end
```

Algorithm 2: Initialize the Indexing Array with the corresponding probabilities

```
Data: Two lists of Fixed_NetIDs and Fixed_Probabilities
Result: Array Indexed_NetIDs with corresponding probabilities
Indexed_NetIDs := Byte[2^24];
for i:=1 to len(Indexed_NetIDs) do
    Indexed_NetIDs := 0;
end
for j:=1 to len(Fixed_NetIDs) do
    PrefixID := Fixed_NetID.ElementAt(j);
    Index := PrefixID/256;
    Indexed_NetIDs[Index] := Fixed_Probabilities.ElementAt(j);
end
```


5.1.3 Searching

Using the Indexed array, the reputation of any IP address can be easily fetched by converting it to an index to its reputation. The reputation searching is illustrated in Algorithm 3.

Algorithm 3: Initialize the Indexing Array with the corresponding probabilities

```
Data: Indexed_NetIDs and IP
Result: Reputation
Index := IP / 256 ;
Reputation := Indexed_NetIDs[Index] ;
```

5.2 Binary Search Algorithm

We employ the classic Binary Search algorithm with a small adjustment. In order to employ Binary Search, we initialize an array with the start and the end addresses of every net block. Then, in order to find a reputation, we find the array index of the start address of an input IP, and use it as an index to the probabilities array.

5.2.1 Data structure

Our data structure for this algorithm is a sorted array. This sorted array contains the initial and end addresses of every net block for which we have a reputation. Our goal in the binary search implementation is not to find a value in the array but to find the relevant range for an IP address. For instance, given a set of $\{1.1.1.0/24, 1.1.24.0/24, 2.16.196.0/20\}$, our data structure to start the search is an array of $[16843008, 16843520, 16848896, 16849152, 34653184, 34654208]$

5.2.2 Initialization

To build our data structure, we need to load all the initial end end addresses of a net block in a sorted array. To achieve this, we read each NetID from a sorted array, and according to its prefix size, we compute the end address of the prefix. We store the initial addresses at Odd indexes and the end addresses at even in indexes of the array. It goes unsaid that this array is as twice as the initial NetID array. Algorithm 4 illustrates the process to construct our data structure.

Algorithm 4: Initializing range array data structure for the binary search

Data: Sorted NetIDs and NetID_Lengths
Result: Sorted range array R_NetIDs
R_NetIDs_size := 2 * len(NetIDs) ;
R_NetIDs = Byte[R_NetIDs_size] Reputation := Indexed_NetIDs[Index] ;
for *i:=1 to len(NetIDs)* **do**
    odd := NetIDs[i];
    even := odd + $2^{32-NetID\_Lengths[i]}$;
    R_NetIDs[2i-1] := odd;
    R_NetIDs[2i] := even;
end

5.2.3 Searching

Our binary search algorithm has a very similar structure to the classic binary search algorithm. That said, we adjusted the algorithm to search for the range an IP address belongs to. If we have an entry for the reputation of the net block the IP address belongs to, then the IP address value falls between an odd (lower band) and an even (higher band) value in our data structure. Otherwise, the IP address value falls between an even (lower band) and an

odd (higher band) value. Following the example we presented in the beginning of this section, the result for searching 1.1.25.53 must return 0. Looking at $[16843008, 16843520, 16848896, 16849152, 34653184, 34654208]$, we learn that 16849205, integer value of 1.1.25.53, falls between element 4 and 5 of the array. Since the lower band, 4, is even we return 0. Algorithm 5 illustrates our binary search algorithm.

Algorithm 5: Binary search algorithm based on our data structure

```
Data: R_NetIDs, Probabilities and IP
Result: Reputation
low := 1 ;
high = len(R_NetIDs) ;
while low + 1 ¡= high do
    middle := ⌈(low+high)/2⌉;
    even := odd + 2^(32−NetID_Lengths[i]) ;
    if R_NetIDs[middle]¿=IP then
        high := m;
    else
        low := m;
    end
end
if low%=1 then
    index := ⌊low/2⌋ + 1 ;
    Reputation := R_NetIDs[index];
else
    Reputation := 0;
end
```

Chapter 6

Experimental Results

For all the experiments, testing period is one week immediately after the training end date.

6.1 BGP Prefix Length Distribution

We measure the distribution of the prefixes and IP indicators over different lengths. Figure 6.1 shows such distributions. This graph gives us two insights. First, BGP aggregation leads to a significant difference in the length of the prefixes. Second, there is not a a correlation between BGP prefix length and the number of hits; it may be assumed that since a larger prefix length encompasses a larger space, it should also have higher hits while this is not the case in reality. Third, there is not any correlation between the number of prefix entries and the number of hits. In other words, more entries for a specific length does not result in the identification of more malicious IP addresses. In conclusion, we learn from the distribution of the prefixes with malicious activity over different lengths in BGP aggregation that aggregation based on BGP would lead to a different view of the prefixes and grouping of Internet hosts.

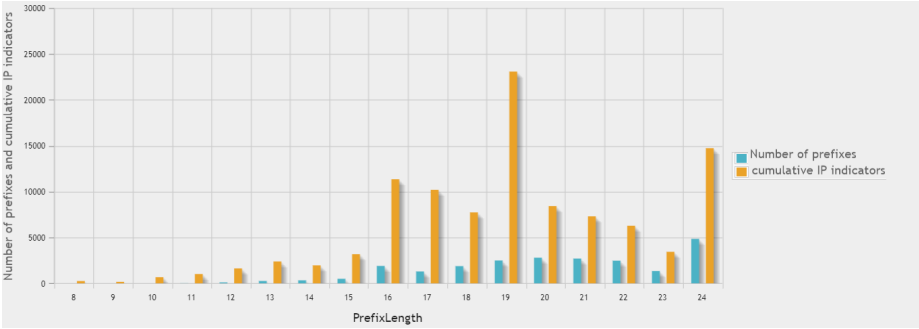


Figure 6.1: Distribution of IP indicators and prefixes over different lengths

training length	Coefficient	Standard Error	P value	Residual std er	Multiple R squared
1 week	0.8808146	0.0105130	$< 2 * 10^{-16}$	0.03193	0.4109
2 weeks	0.8857	0.001517	$< 2 * 10^{-16}$	0.01493	0.8602
3 weeks	0.9692774	0.0020672	$< 2 * 10^{-16}$	0.02034	0.7987

Table 6.1: Linear Regression modeling of P' based on P for Fixed aggregation

training length	Coefficient	Standard Error	P value	Residual std er	Multiple R squared
1 week	1.0140342	0.0084435	$< 2 * 10^{-16}$	0.01912	0.5872
2 weeks	0.7635	0.002823	$< 2 * 10^{-16}$	0.01354	0.7416
3 weeks	0.9429	0.002527	$< 2 * 10^{-16}$	0.01285	0.8501

Table 6.2: Linear Regression modeling of P' based on P for BGP aggregation

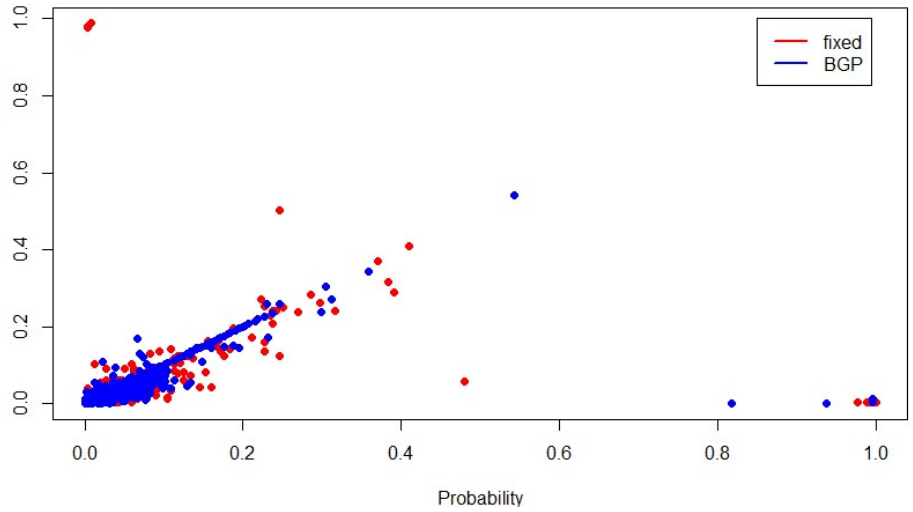
6.2 Precision

As we discussed in **??**, we plot the points in J to see if there is a correlation between P, the probability of malice based on the training set on the x axis, and P', the probability of malice based on the testing set on the y axis. The graphs show a strong correlation for both approaches. In order to measure the precision we will rely on numeric values.

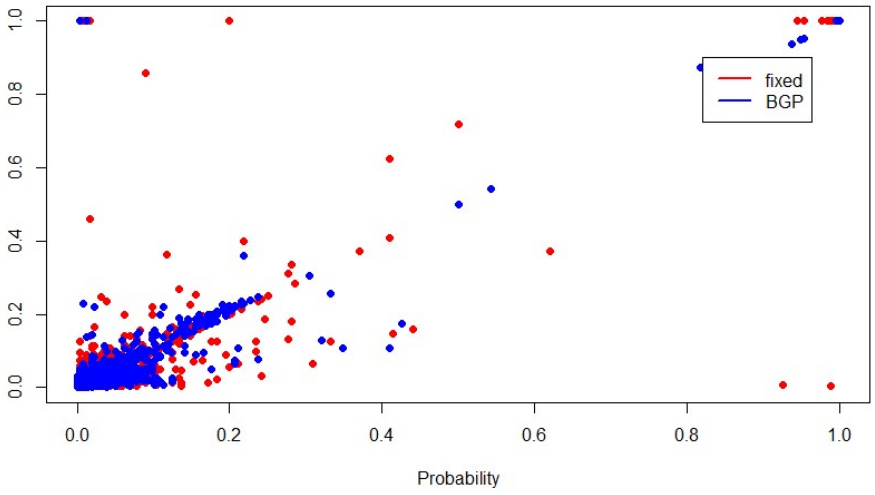
Table 6.1 and 6.2 respectively illustrate the result of linear modeling of P' based on P for Fixed and BGP aggregation. By analyzing the values we conclude the followings:

- In both cases, there is a very strong correlation between P and P'. This means that we can safely use P to predict the probability of maliciousness with a small error based on either approaches
- Two weeks of training for both aggregations lead a finer modeling o P'. Longer training has diminutive effect on both approaches both relatively better effect on BGP
- Based on the R squared value, we can see that BGP will have a better modeling for larger portion of prefixes as the training time increases
- Based on 1 week of training, BGP can better predict P'

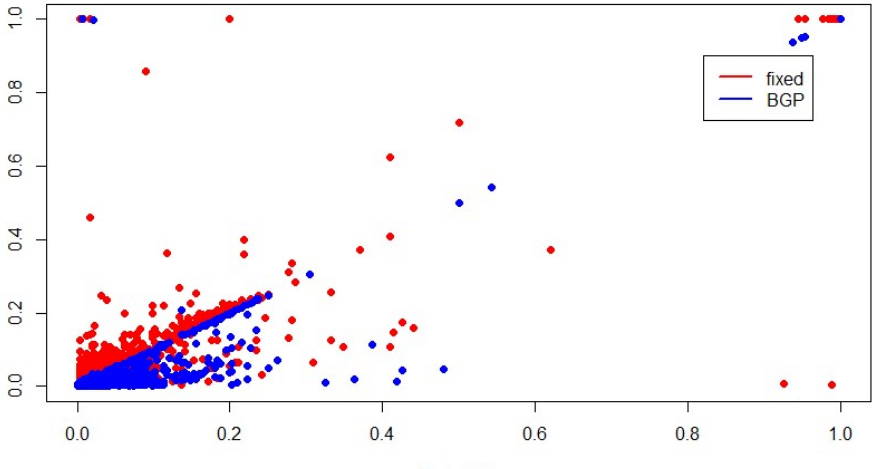
The result of our analysis, in this section, shows that BGP and Fixed aggregation are both adequately precise in the prediction of malice probability. This analysis, however, does not say anything about the probability values effect on our final prediction. To illustrate the concept, let's review an example. Assume that /20 prefix in BGP has 20 malicious indicators based on our training dataset. The probability of compromise for all the members in this /20 prefix is $\frac{20}{2^{12}}$. If all these indicators are accumulated in a /24 prefix X, the probability of malice for members of X is $\frac{20}{2^8}$. Since for the adjacent prefixes of X, we don't have any record the probability of malice for the those prefixes is zero. What our analysis revealed now is that based on the testing set, the probability for all the members of the set (either BGP or fixed) remains almost the same; in this



(a) Training(P) and Testing(P') Probabilities based on 1 week of training



(b) PP' graph based on two weeks of training



3rd Quartile	Mean	Standard Deviation	Variance
0	0.04401	0.8731013	0.7623059

Table 6.3: Normalized Delta distribution statistics

example, the probability for the /20 prefix does not grow or drop significantly, and the same holds for the /24 prefix

6.3 Granularity

What our precision analysis misses to consider is the granularity of the probabilities. Following our previous example, Fixed aggregation gives a probability of $\frac{20}{2^8}$ to all the X members while BGP gives the probability of $\frac{20}{2^{12}}$ to these members that is 16 times larger! Of course this is an example however if in reality BGP aggregation indeed leads to such scenario then Fixed aggregation accuracy is preferred. In order to compare the granularity of the two aggregations and see if the above phenomenon exists we analyze the distribution of $Delta(p)$. Since P' can be small, due to the BGP prefix size, we normalize $Delta(p)$ by dividing it to P'. This normalization allows to understand how larger the probability can be if we use fixed aggregation instead of BGP. Figure 6.3 shows the distribution of normalized $Delta(p)$ value. We can instantly notice that most probabilities have the same value for both approaches. Furthermore, except some outliers the normalize $Delta(p)$ is less than 10. In order to precisely analyze the granularity we reported the statistical features of this distribution, excluding the outliers, in Table 6.3. As illustrated 6.3 75% of the data has no difference in probability and the mean of difference is 4% change with a small variance. The conclusion is that although Fixed aggregation is inevitably more granular, the loss of information by using BGP is negligible. Analyzing the reason behind this phenomenon is beyond the time capacity of this research but an explanation could be homogeneous distribution of maliciousness in the /24 prefixes of a BGP.

6.4 Detection Rate

In order to answer sub question 5 and see if BGP has a meaningful higher Detection Rate, we compare the number of hits and the Detection Rate in of Fixed and BGP aggregation successively in Table 6.4 and 6.5. A quick glance reveals that Detection Rate of BGP is meaningfully higher than Fixed aggregation. We note that after three weeks of training, the Detection Rate of fixed aggregation enhances and becomes very close to that of BGP. This can be explained by the growth in the number of stored fixed prefixes in Fixed aggregation; the coverage of IP addresses by the two methods become very similar. The number of stored BGP prefixed, however, is 40% of Fixed prefixes.

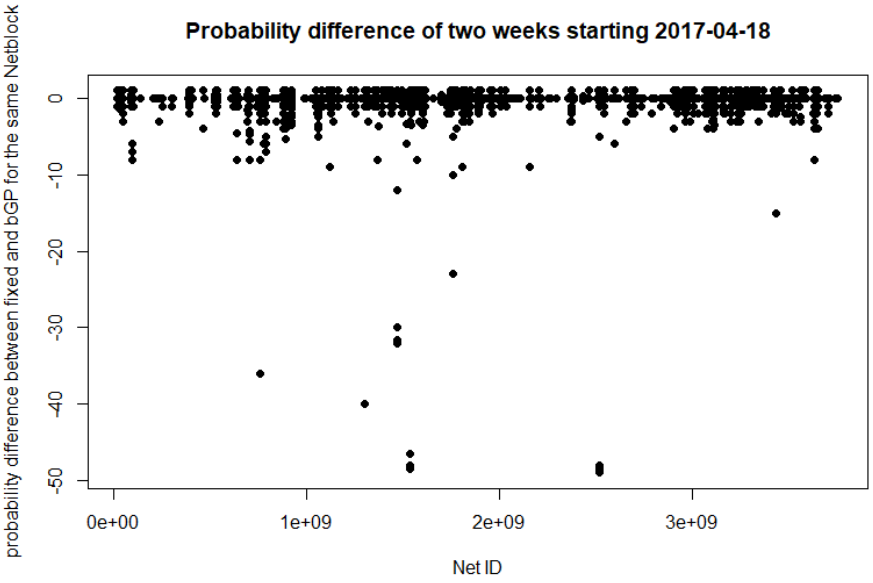


Figure 6.3: Normalized $Delta(p)$ distribution

Although longer training enhances the detection rate, we note that the number of entries without any hit also increases (see Table 6.4 and 6.5 last column). The increase in number of such entries can result in false positives if there is traffic from such prefixes to a monitored source. Therefore, based on the high detection rate and also the low percentage of unused entries(entries without hit) of BGP, we conclude that BGP has a better detection rate; it can faster learn and predict the net blocks that indeed expose malicious activity.

The detection concept that we presented above only says that we can report the reputation of the IP address that communicates through the reputation of its net block. It goes unsaid that the higher the probability that we report, the

Training length	# of hit prefixes	# of hit indicators	Detection rate	Testing prefixes Size	Testing indicators Size	Training prefixes size	% of prefixes without hit
1 week	53014	83445	80%	71866	103293	61222	14%
2 weeks	55396	115562	74%	83018	154930	80072	31%
3 weeks	54835	107655	83%	62557	128470	107692	49%

Table 6.4: Detection rate and values for Fixed aggregation

Training length	# of hit prefixes	# of hit indicators	Detection rate	Testing prefixes Size	Testing indicators Size	Training prefixes size	% of prefixes without hit
1 week	23430	91291	88%	30274	100061	26197	11%
2 weeks	25494	133202	86%	34782	149771	33041	23%
3 weeks	24535	110365	86%	26768	122207	42328	42%

Table 6.5: Detection rate and values for BGP aggregation

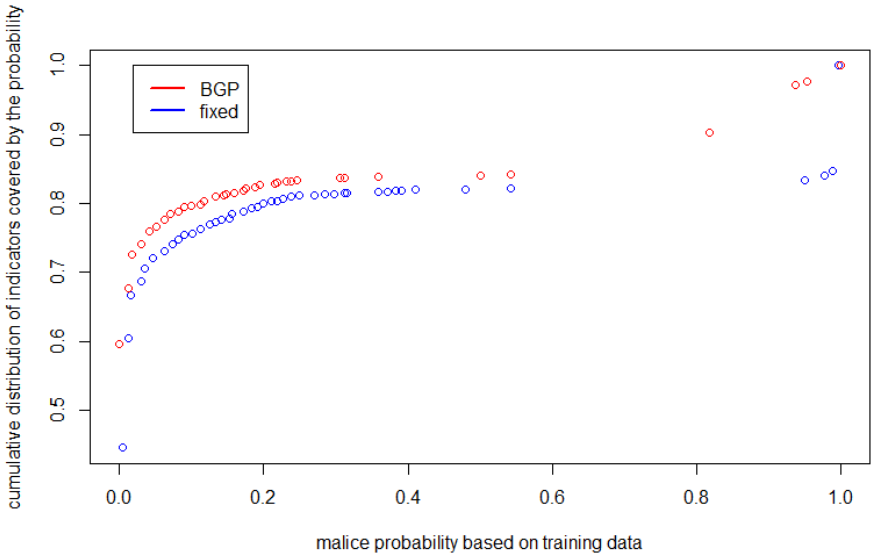


Figure 6.4: CDF of hits based on probability of malice

higher the chance that the traffic is indeed malicious. This implicitly means that for lower probabilities we need to rely more on other features to detect malicious activity. In order to give an insight about the probabilities that we would report based on the hits we employ CDF metric that we defined in Section **??**. Figure 6.4 illustrates the CDF of hits for both aggregation features. We note that for almost 80% of the data we report probabilities less than 20%. Furthermore, for around 60% of hits we report numbers close to zero. This implies that a designer of an IDS must treat such traffic with low probabilities of malice with discretion; on one hand, networks with low probabilities can not be treated as benign since there are indeed malicious traffic from such networks. On the other hand, if other detection features can not distinguish harmful from benign traffic in these networks, a lot of false positives will be raised that is not desired. For other 20% of the hits, we report relatively high probabilities. For the entire data, Fixed aggregation reports higher probabilities and the difference is especially significant for the top left of Figure 6.4.From Section **??** results, we know the reported probabilities are adequately precise for both aggregation features. *We can conclude that fixed aggregation can better identify the malicious traffic for 20% of the data in case probability is the only feature for malicious activity; in such case, traffic from a network with 85% of malicious activity can be labeled malicious with a high confidence.*

b

Training length	Performance metric	/24 Fixed aggregation	BGP aggregation
1 week	Number of prefixes(n)	61222	26197
	Lookup order value	O(log(61222))=O(15.90)	O(log(26197))=O(14.68)
	Footprint	0.55MB	0.24MB
2 weeks	Number of prefixes(n)	80072	33041
	Lookup order value	O(log(80072))=O(16.29)	O(log(33041))=O(15.01)
	Footprint	0.72MB	0.30MB
3 weeks	Number of prefixes(n)	107692	42328
	Lookup order value	O(log(107692))=O(16.72)	O(log(42328))=O(15.37)
	Footprint	0.97	0.38MB

Table 6.6: Lookup performance comparison based on BSA with $O(log(n))$

6.5 Lookup performance

In Chapter 4 we presented two algorithms that are the base for our comparison in this section. Indexing Search Algorithm 3 is optimized for fast processing with $O(1)$. There is, however, a memory footprint penalty with this algorithm. In contrast, Binary Search Algorithm 5 is optimized for memory footprint with a processing overhead penalty, $O(log(n))$. In this section, we compare Fixed with BGP aggregation based on each of this algorithm.

For Indexing Search Algorithm, the lookup order value and footprint for both approaches are the same. Since the Order of this algorithm is $O(1)$ the lookup order value for both approaches is 1. The memory (and Disk) footprint for both Fixed and BGP aggregation is $16MB$ (based on $2^{24}/24$ prefixes and $1B$ for score storage). This may come as a surprise to the reader since BGP has less entries than /24 Fixed prefixes. In 2, however, we processed the BGP entries and derived the probabilities of child /24 prefixes. Therefore, for both cases our point of reference for fetching the malice probability of an IP address is its /24 prefix that is computed by masking the IP value with $ffffff00$. This entails that we have the same values for all the adjacent /24 prefixes in parent BGP prefix.

For Binary Search Algorithm (BSA), the lookup order value and footprint is different for each approach since n (the number of prefixes) is different. Table 6.6 reports the result of our computation for each approach. We elaborate that footprint column is calculated based on the 5. In the implementation of this algorithm, two arrays are required; one for storing the lower and higher band of each prefix and another for storing the scores. Since IP is 32 bits long we assume that the size of the first array is $4 * n * 2 = 8nbytes$ where n is the number of prefixes. 4 in the calculation represents a $4B$ integer and 2 represents the lower and higher band for each prefix. The second array stores a score of one byte long for each prefix. Hence, the footprint for n prefixes is $9n$ Bytes.

According to Table 6.6, the lookup order value for BGP is $O(1)$ faster for the order of n that our dataset has. It is hard to measure the exact computation time performance since it depends on the CPU power and also the number of IPs that are looked up in a unit of time. The footprint for BGP is 2 to 2.5 times less than Fixed aggregation given n based on our datasets.

6.6 Discussion

In summary, our analysis that aggregation of reputation based on BGP announcements is better than aggregation based on Fixed /24 prefixes. Firstly, BGP is as precise and sometime preciser than fixed aggregation in predicting the probability of malice from the parent network of an IP address. Secondly, although Fixed aggregation is always more granular by definition, the loss of granularity by BGP aggregation is negligible. Thirdly, BGP has a better detection rate give a short training time that possibly leads to less false positives. Finally, in terms of lookup performance BGP has equal or better lookup performance in terms of lookup Order and footprint.

Chapter 7

Conclusion and future work

[to be written]

Bibliography

[1] Ward Van Wanrooij and Aiko Pras. Filtering spam from bad neighborhoods. *International Journal of Network Management*, 20(6):433–444, 2010.

[2] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a dynamic reputation system for dns. In *USENIX security symposium*, pages 273–290, 2010.

[3] Giovane CM Moura, Carlos Ganán, Qasim Lone, Payam Poursaied, Hadi Asghari, and Michel van Eeten. How dynamic is the isps address space? towards internet-wide dhcp churn estimation. In *IFIP Networking Conference (IFIP Networking), 2015*, pages 1–9. IEEE, 2015.

[4] Yinglian Xie, Fang Yu, Kannan Achan, Eliot Gillum, Moises Goldszmidt, and Ted Wobber. How dynamic are ip addresses? In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 301–312. ACM, 2007.

[5] Sushant Sinha, Michael Bailey, and Farnam Jahanian. Shades of grey: On the effectiveness of reputation-based "blacklists". In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 57–64. IEEE, 2008.

[6] M Patrick Collins, Timothy J Shimeall, Sidney Faber, Jeff Janies, Rhiannon Weaver, Markus De Shon, and Joseph Kadane. Using uncleanliness to predict future botnet addresses. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 93–104. ACM, 2007.

[7] Shobha Venkataraman, Subhabrata Sen, Oliver Spatscheck, Patrick Haffner, and Dawn Song. Exploiting network structure for proactive spam mitigation. In *Usenix Security*, 2007.

[8] Zhiyun Qian, Zhuoqing Morley Mao, Yinglian Xie, and Fang Yu. On network-level clusters for spam detection. In *NDSS*, 2010.

[9] Fabio Soldo, Anh Le, and Athina Markopoulou. Blacklisting recommendation system: Using spatio-temporal patterns to predict future attacks. *IEEE Journal on Selected Areas in Communications*, 29(7):1423–1437, 2011.

[10] Craig A Shue, Andrew J Kalafut, and Minaxi Gupta. Abnormally malicious autonomous systems and their internet connectivity. *IEEE/ACM Transactions on Networking (TON)*, 20(1):220–230, 2012.

[11] Matthijs GT Van Polen, Giovane CM Moura, and Aiko Pras. Finding and analyzing evil cities on the internet. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*, pages 38–48. Springer, 2011.

[12] Yang Liu, Armin Sarabi, Jing Zhang, Parinaz Naghizadeh, Manish Karir, Michael Bailey, and Mingyan Liu. Cloudy with a chance of breach: Forecasting cyber security incidents. In *USENIX Security*, pages 1009–1024, 2015.

[13] Surbl. `http://www.surbl.org/`. Accessed: 2017-03-22.

[14] Spamhaus project, spamhaus block list (sbl). `https://www.spamhaus.org/sbl/`. Accessed: 2017-03-22.

[15] Composite blocking list. `http://www.abuseat.org/`. Accessed: 2017-03-24.

[16] Spamhaus project, exploits block list (xbl). `https://www.spamhaus.org/xbl/`. Accessed: 2017-03-22.

[17] Ironport systems' spamcop blacklists. `https://www.spamcop.net/bl.shtml/`. Accessed: 2017-03-24.

[18] Malware domain list. `https://www.malwaredomainlist.com/`. Accessed: 2017-03-24.

[19] Dns-bh – malware domain blocklist. `http://www.malwaredomains.com/`. Accessed: 2017-03-24.

[20] Zeus tracker. `https://zeustracker.abuse.ch/`. Accessed: 2017-03-24.

[21] Joe wein spam domain blacklist. `http://www.joewein.de/sw/blacklist.htm/`. Accessed: 2017-03-24.

[22] Uribl. `http://uribl.com/`. Accessed: 2017-03-24.

[23] Sorbs. `http://www.sorbs.net/`. Accessed: 2017-03-24.

[24] Dsheild. `https://www.dshield.org`. Accessed: 2017-03-24.

[25] Opendns, phishtank. `http://www.phishtank.com/`. Accessed: 2017-03-22.

[26] hphosts for your pretection. `http://hosts-file.net/`. Accessed: 2017-03-29.

[27] Openbl. `http://www.openbl.org/`. Accessed: 2017-03-24.

[28] Veris community database. `http://veriscommunity.net/vcdb.html/`. Accessed: 2017-03-24.

[29] Web-hacking-incident-database. `http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database`. Accessed: 2017-03-24.

[30] Wpbl: Weighted private block list. `http://www.wpbl.info/`. Accessed: 2017-03-29.

[31] Support intelligence, llc. `http://www.support-intelligence.com/`. Accessed: 2017-03-22.

[32] Uceprotector network. `http://www.uceprotect.net/`. Accessed: 2017-03-29.

[33] Apwg, anti-phishing working group. `http://www.antiphishing.org`. Accessed: 2017-03-22.

[34] Netpilot gmbh, viruswatch mailing list. `http://lists.clean-mx.com/cgi-bin/mailman/listinfo/viruswatch/`. Accessed: 2017-03-22.

[35] Malware patrol, malware block list. `http://www.malwarepatrol.net/`. Accessed: 2017-03-22.

[36] Shadowserver foundation. `https://www.shadowserver.org/`. Accessed: 2017-03-22.

[37] Giovane CM Moura, Ramin Sadre, Anna Sperotto, and Aiko Pras. Internet bad neighborhoods aggregation. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 343–350. IEEE, 2012.

[38] Shuang Hao, Nadeem Ahmed Syed, Nick Feamster, Alexander G Gray, and Sven Krasser. Detecting spammers with snare: Spatio-temporal network-level automatic reputation engine. In *USENIX security symposium*, volume 9, 2009.

[39] Jian Zhang, Phillip A Porras, and Johannes Ullrich. Highly predictive blacklisting. In *USENIX Security Symposium*, pages 107–122, 2008.

[40] Mark Felegyhazi, Christian Kreibich, and Vern Paxson. On the potential of proactive domain blacklisting. *LEET*, 10:6–6, 2010.

[41] Michel Van Eeten, Johannes M Bauer, Hadi Asghari, Shirin Tabatabaie, and David Rand. The role of internet service providers in botnet mitigation an empirical analysis based on spam data. In *Proceedings of the 9th Workshop on the Economics of Information Security (WEIS)*, 2010.

[42] Niels Provos Panayiotis Mavrommatis and Moheeb Abu Rajab Fabian Monrose. All your iframes point to us. In *USENIX Security Symposium*.

[43] Jing Zhang, Zakir Durumeric, Michael Bailey, Mingyan Liu, and Manish Karir. On the mismanagement and maliciousness of networks. In *NDSS*, 2014.

[44] Hadi Asghari. Python ip address to autonomous system number lookup using archived bgp dumps. `https://github.com/hadiasghari/pyasn`. Accessed: 2017-07-13.

[45] RouteViews Project University of Oregon Advanced Network Technology Center. Route views project. `http://www.routeviews.org/`. Accessed: 2017-03-22.

[46] Ali Davanian. Prefix reputation dataset. `https://github.com/adavanian/badhood`. Accessed: 2017-07-13.