Detecting Malicious Websites by Learning IP Address Features

Daiki Chiba*, Kazuhiro Tobe*, Tatsuya Mori† and Shigeki Goto*
*Department of Computer Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555 JAPAN
Email: {chiba, tobe, goto}@goto.info.waseda.ac.jp
†NTT Service Integration Laboratories, NTT Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo 180-8585 JAPAN
Email: mori.tatsuya@lab.ntt.co.jp

***Abstract*—Web-based malware attacks have become one of the most serious threats that need to be addressed urgently. Several approaches that have attracted attention as promising ways of detecting such malware include employing various blacklists. However, these conventional approaches often fail to detect new attacks owing to the versatility of malicious websites. Thus, it is difficult to maintain up-to-date blacklists with information regarding new malicious websites. To tackle this problem, we propose a new method for detecting malicious websites using the characteristics of IP addresses. Our approach leverages the empirical observation that IP addresses are more stable than other metrics such as URL and DNS. While the strings that form URLs or domain names are highly variable, IP addresses are less variable, i.e., IPv4 address space is mapped onto 4-bytes strings. We develop a lightweight and scalable detection scheme based on the machine learning technique. The aim of this study is not to provide a single solution that effectively detects web-based malware but to develop a technique that compensates the drawbacks of existing approaches. We validate the effectiveness of our approach by using real IP address data from existing blacklists and real traffic data on a campus network. The results demonstrate that our method can expand the coverage/accuracy of existing blacklists and also detect unknown malicious websites that are not covered by conventional approaches.**

***Keywords*-IP address; Blacklist; Web-based malware; Drive-by-download; Machine learning;**

I. Introduction

Web-based malware attacks have become one of the most serious threats that need to be addressed urgently. Some malicious websites steal users' confidential information, which may include login IDs, passwords, and personal information. Other malicious websites enforce users to download malicious software (malware).

Web-based malware attacks target vulnerabilities that exist in web browsers and several plugins such as Flash players, Java VMs, and PDF plugins [1]. These vulnerabilities are exploited to compromise the browser to download and run a malware on the targeted system. Such attacks are often called *drive-by-download* attacks [2].

Computers are subjected to conventional attacks when they are connected to the Internet or external devices such as a USB memory that is infected with malware. In contrast, drive-by-download attacks are triggered by users' access to certain websites. Fig. 1 illustrates the procedure of a typical drive-by-download attack. When a browser accesses a compromised *landing* site, the HTTP connection is redirected to a *hopping* site. A hopping site is a website that contains a redirect instruction code that redirects an HTTP connection to the next hopping site or an *exploit* site. After a connection has been redirected to an exploit site, the browser is forced to download malware from a *malware distribution site*. An exploit site is a website that actually exploits vulnerabilities of users' web browsers.

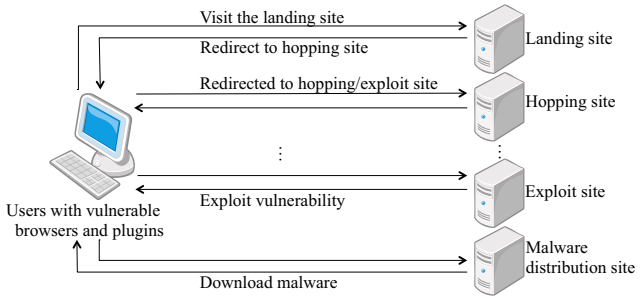


Figure 1. Procedure of a Drive-by-Download Attack.

Owing to the complexity of the infection procedure shown above, the detection of infection by web-based malware is often complex. Authors of malware use several techniques that redirect users to actual malware distribution sites by masquerading them as exciting and fun themes on social networking websites or e-mails [3]. The redirection sites can be easily updated. Several intermediate redirection URLs are effective for one-time access only. Moreover, they employ various obfuscation techniques such as encryption, polymorphism, and tunneling to evade detection. Therefore, conventional approaches often fail to detect new attacks owing to the versatility of malicious website deployment, which is discussed in Section II. To resolve this problem, this paper presents a new method for detecting malicious websites using the characteristics of IP addresses.

This paper proposes a method for detecting communication associated with web-based malware on the basis of

978-0-7695-4737-4/12 $26.00 © 2012 IEEE
DOI 10.1109/SAINT.2012.14

features extracted from structures of IP addresses in order to prevent users from accessing even *unknown* malicious websites. Our approach leverages the empirical observation that IP addresses are more stable than other metrics such as URLs and DNS. While the strings that form URLs or domain names are highly variable, IP addresses are less variable, i.e., IPv4 address space is mapped onto 4-bytes strings. We develop a lightweight and scalable detection scheme that is based on the machine learning technique. Note that the goal of this paper is not to provide a single solution that effectively detects web-based malware but to develop a technique that compensates for the limitations of existing approaches. We validate the effectiveness of our approach by using real IP addresses data obtained from existing blacklists and real traffic data on a campus network. The results demonstrate that our method accurately differentiates between the IP addresses used for benign websites and malicious websites. In addition, we show that our method expands the coverage/accuracy of existing blacklists and detects even *unknown* malicious websites that are not covered by conventional approaches.

The rest of this paper is organized as follows. We first review related works and discuss their limitations in Section II. Then, we present our detection methodology in Section III. Section IV illustrates the experimental results using actual web traffic data collected on a large-scale campus network. Finally, we conclude our study in Section V.

II. Related Work

This section reviews related works and discusses their limitations. We divide the systems proposed in these works into three categories: blacklists and reputation systems, intrusion detection systems (IDS), and client honeypots.

A. Blacklists and Reputation Systems

So far, blacklists and reputation systems have been the most popular solutions that prevent users from accessing malicious websites. Blacklists can be applied to both network-side and client-side filtering. Network-side filtering can be used with DNS block lists (DNSBL) [4], [5] and commercial security appliances. Client-side filtering can be included in current web browsers, as in [6], [7].

In reputation systems, reputation is based on various types of information present in each IP address or domain and is applied to prevent users from accessing malicious websites. Criteria for reputation include features retrieved from domains, WHOIS information, and link structures. Antonakakis et al. [8] developed a dynamic reputation system called *Notos*. The system collects information from multiple sources such as DNS zones, border gateway protocol prefixes, and AS information to model network and zone behaviors of benign and malicious domains. Then, it applies these models to calculate a reputation score for each domain name. Felegyhazi et al. [9] proposed domain-based proactive blacklisting. It utilizes a small set of known malicious domains to predict other malicious domains on the basis of registration and name server information. Ma et al. [10] proposed a supervised learning approach for classifying URLs as benign or malicious on the basis of both lexical structure of URLs and host-based features such as WHOIS records and geographical information.

Although these approaches are widely employed, they often fail to keep up with the transient nature of malicious activities. For instance, it was demonstrated that maintaining up-to-date blacklists is not easily accomplished because new malicious domain names can be easily and continuously generated [11]–[13]. Furthermore, it has been reported that attackers frequently change domain names to evade detection by reputation systems [1]. In addition, Shin et al. [14] showed that only 17% of worm/bot victims are covered by several blacklists and they pointed out that better ways to detect future emerging malware are needed.

In summary, existing blacklist-based approaches are prone to failure with respect to detecting versatile web-based malware.

B. Intrusion Detection Systems

IDS can be used to block users from accessing malicious websites. It can be classified into two types: signature-based IDS and anomaly-based IDS. Signature-based IDS such as Bro [15] and Snort [16] monitor network traffic and search packets that correspond to predefined attack signatures. Because attack signatures are generated by known attacks, signature-based IDS cannot detect unknown attacks. Anomaly-based IDS learns normal network behavior and uses this information to detect attacks. Therefore, it has potential for detecting unknown attacks [17]. However, there is a high probability of anomaly-based IDS falsely regarding normal traffic as attacks, namely false positives. Moreover, malicious websites often contain *obfuscated* scripts to evade such IDS. As we will show later in Section IV-F, detecting malicious websites with IDS is not readily performed in real environments. Therefore, it is recognized that IDS has limitations with respect to blocking all kinds of malicious websites.

In summary, existing IDS-based approaches may fail to detect obfuscated malware.

C. Client Honeypots

Recent studies have proposed *client honeypot systems* that aim to detect and analyze drive-by-download attacks [2], [18], [19]. A client honeypot is a system that crawls websites and detects malicious websites. Client honeypots can be classified into two types: low-interaction honeypots and high-interaction honeypots. Low-interaction honeypots such as HoneyC [18] include an emulator of a browser to crawl websites. Therefore, there is no risk that honeypots

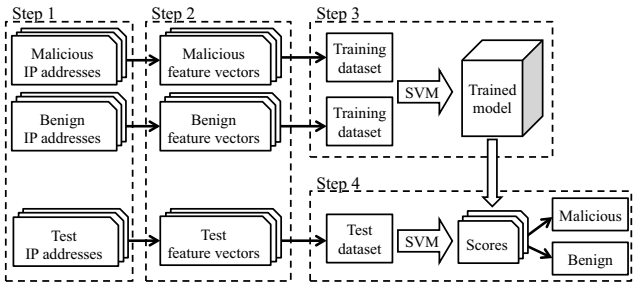


Figure 2. Overview of Our Method.

themselves will be infected with malware. High-interaction honeypots such as Capture-HPC [19] include a real web browser and system; therefore, they can collect more information such as malware behavior after exploitation. One clear drawback of high-interaction honeypots is the risk of malware infection because they actually run exploit codes. Akiyama et al. [2] proposed a state-of-the-art high-interaction honeypot called *Marionette* that overcomes the risk of malware infection. Marionette has a real vulnerable web browser and plugins that can detect attacks without being infected with malware.

However, client honeypots still have three major problems that affect their detection of all malicious websites. One problem is the lack of scalability. At present, attackers generally deploy a large number of malicious websites [13], whose URLs change within a short time period [12]. Thus, it is not feasible to study the entire set of malicious URLs with client honeypot systems. Another problem is that even if websites are benign while being investigated, they may be attacked afterwards and vice versa. Therefore, it is necessary to shorten the intervals between investigations. However, because of time and cost limitations, it is not feasible to investigate all URLs several times. In addition, malicious websites utilize *cloaking* techniques to hide their malicious contents from particular IP addresses used by honeypots [20]. This cloaking makes it more difficult for client honeypots to crawl and detect malicious websites [1].

In summary, existing client honeypot-based approaches have some problems: lack of scalability and versatility and failure to detect cloaking techniques.

III. Detection Methodology

In this section, we first present a high-level overview of our detection methodology. Next, we present several feature extraction techniques. Finally, we show how a machine learning approach can be applied to our detection methodology.

A. High-level overview

Our method is based on two intrinsic characteristics of IP addresses: (1) stability with time [21] and (2) address space skewness [22], [23]. First, although attackers can change URLs and DNS at a low cost, it is much more difficult to change IP addresses, which are essentially associated with malicious activity. Thus, the characteristics of an IP address should be more stable compared with other metrics. Second, IP addresses associated with malicious activity are likely to be concentrated in certain network address spaces, as reported by previous measurement studies [22], [23]. In Section IV, we will show that IP addresses of web-based malware also have this characteristic. To the best of our knowledge, the approach presented here is the first IP address-based approach against malicious websites that employ drive-by-download attacks. On the other hand, approaches against botnets, phishing, and spam mails have been widely studied, as in [21]–[23].

Basically, our method blocks users' access to malicious websites by extending existing blacklists. The unique feature of our method is that it can evaluate IP addresses that are not listed in existing blacklists. To complicate analysis and detection attempts, drive-by-download attacks lead users to an actual attacking website via multiple stepladder sites by redirecting users' browsers many times. Therefore, blocking access to the IP address of a destination malicious website can protect users from malware infection.

Our method involves the following four steps: 1) collecting IP addresses, 2) extracting feature vectors, 3) building a trained model, and 4) detecting malicious IP addresses. Our key technical contribution is building a novel feature extraction scheme (step 2), which is described in the next subsection. We employ supervised machine learning techniques for steps 3 and 4. Fig. 2 outlines our method.

B. Feature extraction

Our task is to project an IP address space onto a feature space that reflects the relationship between IP addresses and the network structure and can be interpreted by machine learning algorithms. This paper proposes three methods to extract a feature vector from an IP address: *octet-based extraction (Octet)*, *extended octet-based extraction (ExOctet)*, and *bit string-based extraction (Bit)*.

Octet-based Extraction (Octet):

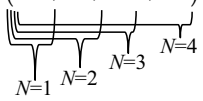
Octet constructs an $M = 2^8 \times N$-dimensional feature vector represented as a sparse bit sequence $\{b_0, \cdots, b_{M-1}\}$ from the most significant N octets of an IPv4 address, where N is a natural number, ranging between one and four that is used as a parameter. The initial value for each bit $\{b_0, \cdots, b_{M-1}\}$ is zero. A feature vector is represented by the following equation:

$$\begin{cases} b_k = 1 & (k \ in \ \bigcup_{n=1}^{N} \{2^8 \cdot (n-1) + X_n\}) \\ b_k = 0 & (otherwise), \end{cases}$$

where k is an index set of feature vector.

IPv4 address: 198.51.100.88

$(X_1, X_2, X_3, X_4) = (198, 51, 100, 88)$

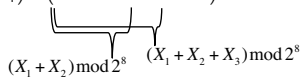


N	n-th octet	Feature vector	M
1	1	$b_k = 1\ (k = 198)$ $b_k = 0\ (otherwise)$ $(0 \le k \le 255)$	256
2	1, 2	$b_k = 1\ (k = 198, 307)$ $b_k = 0\ (otherwise)$ $(0 \le k \le 511)$	512
3	1, 2, 3	$b_k = 1\ (k = 198, 307, 612)$ $b_k = 0\ (otherwise)$ $(0 \le k \le 767)$	768
4	1, 2, 3, 4	$b_k = 1\ (k = 198, 307, 612, 856)$ $b_k = 0\ (otherwise)$ $(0 \le k \le 1023)$	1024

Figure 3. Example of Octet-Based Extraction.

IPv4 address: 198.51.100.88

$(X_1, X_2, X_3, X_4) = (198, 51, 100, 88)$



N	n-th octet	Feature vector	M
3	1, 2, 3	$b_k = 1\ (k = 198, 307, 612)$ $b_k = 1\ (k = 2^8 \cdot 3 + (198 + 51) \bmod 2^8)$ $b_k = 1\ (k = 2^8 \cdot 4 + (198 + 51 + 100) \bmod 2^8)$ $b_k = 0\ (otherwise)$ $(0 \le k \le 1279)$	1280

Figure 4. Example of Extended Octet-Based Extraction.

Let $b_{2^8(n-1)+X_n} = 1$ when the n-th $(1 \le n \le N)$ octet of an IPv4 address is represented as X_n $(0 \le X_n \le 2^8 - 1)$ in decimal notation. Fig. 3 shows an example of feature vectors corresponding to IPv4 address 198.51.100.88 with Octet. The upper half of the figure illustrates the correspondence relationship between parameter N and feature extraction coverage. A feature vector uses the first octet of an IP address when $N = 1$; the first and second octets when $N = 2$; the first, second, and third octets when $N = 3$; and the first, second, third, and fourth octets when $N = 4$. The lower half of Fig. 3 lists the feature vectors extracted by parameter N. For example, when $N = 3$, k consists of 198, 307 $(= 256 + X_2)$, and 612 $(= 512 + X_3)$.

Extended Octet-based Extraction (ExOctet):

ExOctet extends Octet's feature vector to construct an $M = 2^8 \times (N + 2)$-dimensional feature vector represented as a sparse bit sequence $\{b_0, \cdots, b_{M-1}\}$ from the most significant N octets of an IPv4 address, where N is a natural number greater than or equal to three. The initial value for each bit $\{b_0, \cdots, b_{M-1}\}$ is zero. A feature vector is represented according to the following equation:

$$\begin{cases} b_k = 1 & (k\ in \bigcup_{n=1}^{N} \{2^8 \cdot (n-1) + X_n\}) \\ b_k = 1 & (k\ in \bigcup_{m=N \ge 3}^{N+1} \{2^8 \cdot m + (\sum_{i=1}^{m-1} X_i) \bmod 2^8\}) \\ b_k = 0 & (otherwise). \end{cases}$$

Let $b_{2^8(n-1)+X_n} = 1$ when the n-th $(1 \le n \le N)$ octet of an IPv4 address is represented as X_n $(0 \le X_n \le 2^8 - 1)$ in decimal notation. Exoctet adds $b_{2^8 \cdot 3 + (X_1 + X_2) \bmod 2^8} = 1$ and $b_{2^8 \cdot 4 + (X_1 + X_2 + X_3) \bmod 2^8} = 1$ when $N = 3$. Fig. 4 shows an example of feature vectors corresponding to IPv4 address 198.51.100.88 with ExOctet. The upper half of the figure illustrates the correspondence relationship of extended coverage that is different from that of Octet. A feature vector uses the first, second, and third octets of an IP address when $N = 3$, which is the same as Octet. Moreover, a feature vector is extended with a combination of the first and second octets, and that of the first, second, and third octets.

Bit String-based Extraction (Bit):

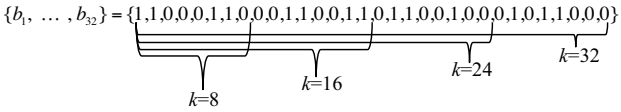
Bit constructs a k-dimensional feature vector represented as a bit sequence $\{b_1, \cdots, b_k\}$ from a 32-bit binary form of IPv4 address, where k is a natural number used as a parameter. The value for each bit $\{b_1, \cdots, b_k\}$ is equivalent to first k bits of a binary-formatted IPv4 address. A feature vector is represented according to the following equation:

$$\begin{cases} b_k = 1 & (\text{when the } k\text{-th bit value of IPv4 address is 1}) \\ b_k = 0 & (otherwise). \end{cases}$$

Let $b_k = 1$ when the k-th bit value of binary-formatted IPv4 address is 1. Fig. 5 shows an example of feature vectors corresponding to IPv4 address 198.51.100.88 with Bit. The

IPv4 address: 198.51.100.88

$\{b_1, \ldots, b_{32}\} = \{1,1,0,0,0,1,1,0,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,0,0,0,1,0,1,1,0,0,0\}$



k	Feature vector
8	$b_k = 1\ (k = 1,2,6,7)$ $b_k = 0\ (otherwise)$ $(1 \le k \le 8)$
16	$b_k = 1\ (k = 1,2,6,7,11,12,15,16)$ $b_k = 0\ (otherwise)$ $(1 \le k \le 16)$
24	$b_k = 1\ (k = 1,2,6,7,11,12,15,16,18,19,22)$ $b_k = 0\ (otherwise)$ $(1 \le k \le 24)$
32	$b_k = 1\ (k = 1,2,6,7,11,12,15,16,18,19,22,26,28,29)$ $b_k = 0\ (otherwise)$ $(1 \le k \le 32)$

Figure 5. Example of Bit String-Based Extraction.

Table I
EXAMPLE OF A TRAINING DATASET

Label t_n	IP address	Feature vector $\boldsymbol{x}_n$
$+1$	198.51.100.88	$\{1,1,0,0,0,1,1,0,0,0,1,1,0,0,1,1,0,1,1,0,0,1,0,0\}$
-1	192.0.2.1	$\{1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0\}$
-1	203.0.113.24	$\{1,1,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,1\}$
$\cdots$	$\cdots$	$\cdots$

upper half of Fig. 5 illustrates the correspondence relationship between parameter k and feature extraction coverage, and the lower half lists the feature vectors extracted by parameter k.

C. Application of Machine learning

We apply a supervised binary classifier to the feature vectors extracted in Step 2. Several options exist for a supervised binary classifier such as the logistic regression model, neural network, Naive Bayes classifier, and support vector machine (SVM). Because it has been reported that SVM works very well for various problems in many areas [24], we adopt it to demonstrate the effectiveness of our approach. As demonstrated in the subsequent sections, our detection scheme with SVM works successfully for real datasets. Note that our scheme is generic and can leverage any type of classifier that fits our problem formulation. A key contribution of this paper is the building of effective feature vectors that can be input to supervised classifiers.

Table I shows an example of a training dataset used in this study. The feature vectors in this example are extracted using Bit when $k = 24$. The training dataset comprises N input vectors $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N$ with corresponding target label values $t_1, \cdots, t_N$, where $t_n \in \{-1(\text{benign}), +1(\text{malicious})\}$.

We now briefly show how we train SVM classifiers using the training dataset. SVM uses the concept of *margin*, which is defined to be the smallest distance between the decision boundary and any of the samples. The decision boundary is chosen to be the one which the margin is maximized. The discrimination function of SVM is defined as follows:

$$y(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\phi(\mathbf{x}) + \beta,$$

where $\mathbf{w}$ denotes the parameter to move the decision boundary, $\mathbf{w}^{\mathbf{T}}$ is transposed matrix of $\mathbf{w}$, $\phi(\mathbf{x})$ denotes the feature space transformation function, and β is bias parameter. Using the discrimination function above, the label C of a sample with feature vector $\mathbf{x}$ can be inferred as

$$\widehat{C} = \begin{cases} 1 & (y > 0) \\ -1 & (y < 0). \end{cases}$$

In SVM, parameters $\mathbf{w}$ and β are trained so that margins are maximized. The optimization problem is formulated as follows:

$$\text{argmax}_{\mathbf{w},\beta} \left\{ \frac{1}{|\mathbf{w}|} \min_n \left[t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + \beta) \right] \right\}.$$

To numerically derive the solutions, we use sequential minimal optimization [25]. For the kernel function, we adopt the Gaussian kernel: $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T\phi(\mathbf{y}) = e^{-\gamma||\mathbf{x}-\mathbf{y}||}$. To obtain the best parameters, γ and β, we perform grid search with cross-validation tests. To estimate the probability of SVM's binary output, we apply the technique proposed by Platt in [26]. The key idea is to apply the logistic sigmoid function to the discrimination function of SVM, $y(\mathbf{x})$. The *score* is calculated as follows:

$$score = P(t = 1|\mathbf{x}) = \sigma(Ay(\mathbf{x}) + B), \qquad (1)$$

where $\sigma(a)$ is the logistic sigmoid function defined by $\sigma(a) = 1/(1 + \exp(-a))$. The values for parameters A and B are found by minimizing the cross-entropy error function defined by a training dataset consisting of pairs of values $y(\mathbf{x}_n)$ and t_n. We can now use the scores assigned to each IP address for controlling the risk of various errors, which are discussed later.

IV. EXPERIMENTS

This section illustrates the experimental results using real IP addresses data obtained from existing blacklists and actual web traffic data collected on a large-scale campus network.

A. Dataset

To evaluate our method, we collected URLs of both benign and malicious websites and web traffic data captured on a campus network. Table II shows the training dataset for building the trained model, and Table III shows the test dataset used to evaluate our method.

Our benign training dataset TRN_B comprises URLs of the top 10,000 websites on the Alexa traffic ranking list [27] on April 30, 2011. From these URLs, we resolved 10,372 IP addresses. Domain names in the ranking list include those to which multiple IP addresses are assigned for load balancing, namely DNS round robin and the content delivery network (CDN). Therefore, the number of IP addresses in the ranking list exceeds the number of URLs, which are related to domain names. The Alexa ranking is calculated from a combination of the average daily visitors and page views during the month immediately preceding this study [27]. Therefore, it contains both benign sites and less benign sites such as pornographic and file-sharing sites.

Our benign test dataset TST_B consists of HTTP traffic data captured on a campus network for two weeks in

Table II
TRAINING DATASET

Data	Period	#URLs	#IP addresses
TRN_B	Apr 30, 2011	10,000	10,372
TRN_M	Jan 1, 2009–Apr 30, 2011	63,694	14,171

Table III
TEST DATASET

Data	Period	#URLs	#IP addresses
TST_B	May 1, 2011–May 14, 2011	96,597	57,190
TST_M_ACTIVE	May 1, 2011–May 14, 2011	11,223	2,450
TST_M_NEW	May 1, 2011–May 14, 2011	455	161

May 2011. The campus network is a production network with /16 prefix lengths and is used by more than 50,000 students and faculty members. The average throughput of the campus traffic is approximately 300-400 Mbps, and that of HTTP traffic is up to about 25 Mbps. To minimize the probability that malicious websites are contained in TST_B, we checked all URLs in the captured data using Google Safe Browsing API [7]. Google Safe Browsing is constantly updated with blacklists of suspected phishing and malware related websites. As a result, we excluded 2,515 URLs that were suspected of being malicious sites, i.e., phishing or malware sites.

Our malicious data consists of IP addresses selected from the malware domain list (MDL) [28], which contained 64,152 malicious website entries as of May 14, 2011. Note that MDL contains some malicious websites on web hosting, which utilize the same IP address for multiple domain names. Therefore, the number of URLs is greater than that of IP addresses. We divided MDL into two subsets as of May 1, 2011 and used them as training dataset TRN_M and test data. Then, we divided the test data into two subsets: TST_M_ACTIVE and TST_M_NEW. TRN_M contains 14,171 unique malicious IP addresses collected over a period of more than two years from January 1, 2009 to April 30, 2011. TST_M_ACTIVE contains 2,450 unique active malicious IP addresses observed for two weeks from May 1 to May 14, 2011. TST_M_NEW consists of 161 malicious IP addresses that are unknown to the trained model. To evaluate the generalization ability of the trained model, from TST_M_NEW, we eliminated any IP addresses that are contained in TRN_M. Moreover, from TST_M_NEW, we eliminated any IP addresses for which the top three octets are equal to those of an IP address contained in TRN_M. Therefore, TST_M_NEW contains IP addresses that have *never* been observed. They can be seen as the IP addresses that are *not* covered by blacklists, yet.

B. Evaluation Methodology

This paper defines the correct classification of an actually malicious IP address into a malicious category as a true positive (TP), the incorrect classification of an actually benign IP address into a malicious category as a false positive (FP), the incorrect classification of an actually malicious IP address into a benign category as a false negative (FN), and the correct classification of an actually benign IP address into a benign category as a true negative (TN). Table IV shows the relationships among these terms.

Table IV
RELATIONSHIPS AMONG TERMS

		Actual Value	
		Malicious	Benign
Prediction Outcome	Malicious	True Positive (TP)	False Positive (FP)
	Benign	False Negative (FN)	True Negative (TN)

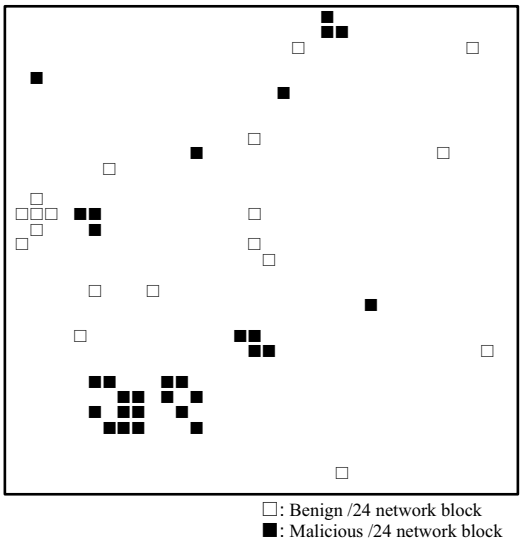


Figure 6. Visualization of IP addresses on x.0.0.0/14.

We evaluated our method using the following criteria: accuracy, false positive rate (FPR), false negative rate (FNR), receiver operator characteristic (ROC) curve, and area under the curve (AUC). Accuracy, FPR, and FNR are calculated as follows:

$$\begin{aligned} Accuracy &= (TP+TN)/(TP+FP+FN+TN) \\ FPR &= FP/(FP+TN) \\ FNR &= FN/(FN+TP). \end{aligned}$$

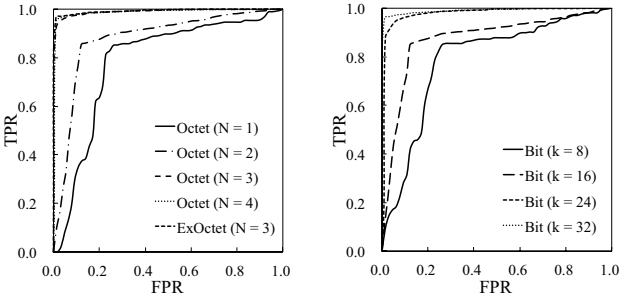
The ROC curve is a graphical plot of the true positive rate ($TPR = TP/(TP+FN)$) vs. FPR for every possible cut-off points. The cut-off point relates to $score$, as described in Section III-C. Each point on the ROC curve represents a (FPR, TPR) pair corresponding to a particular decision cut-off point. Therefore, if the curve rises rapidly towards the upper left corner, it means that overall test result is good. AUC is an area under an ROC curve that is used to score a

Table V
DETECTION PERFORMANCE WITH THE TEST DATASET COMBINATION TST_B AND TST_M_ACTIVE

Method	Accuracy	AUC	FPR	FNR
Octet ($N = 1$)	0.751	0.788	0.253	0.151
Octet ($N = 2$)	0.862	0.878	0.138	0.140
Octet ($N = 3$)	0.885	0.989	0.119	0.024
Octet ($N = 4$)	0.860	0.988	0.145	0.016
ExOctet ($N = 3$)	0.905	0.994	0.098	0.020
Bit ($k = 8$)	0.751	0.790	0.253	0.151
Bit ($k = 16$)	0.847	0.874	0.154	0.136
Bit ($k = 24$)	0.857	0.979	0.148	0.026
Bit ($k = 32$)	0.835	0.991	0.172	0.017

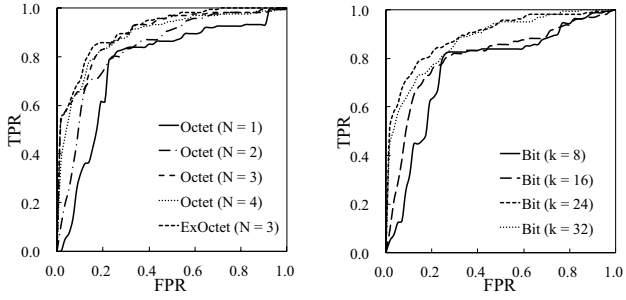
Table VI
DETECTION PERFORMANCE WITH THE TEST DATASET COMBINATION TST_B AND TST_M_NEW

Method	Accuracy	AUC	FPR	FNR
Octet ($N = 1$)	0.747	0.768	0.253	0.180
Octet ($N = 2$)	0.861	0.834	0.138	0.304
Octet ($N = 3$)	0.880	0.897	0.119	0.335
Octet ($N = 4$)	0.855	0.885	0.145	0.217
ExOctet ($N = 3$)	0.902	0.914	0.098	0.292
Bit ($k = 8$)	0.747	0.762	0.253	0.180
Bit ($k = 16$)	0.846	0.804	0.154	0.304
Bit ($k = 24$)	0.851	0.893	0.148	0.205
Bit ($k = 32$)	0.828	0.878	0.172	0.261



(a) Octet and ExOctet (b) Bit

Figure 7. ROC Curves of Detection Performance with the Test Dataset Combination TST_B and TST_M_ACTIVE.



(a) Octet and ExOctet (b) Bit

Figure 8. ROC Curves of Detection Performance with the Test Dataset Combination TST_B and TST_M_NEW.

binary classifier. AUC is calculated as follows:

$$AUC = \sum_{i \in \Omega} \delta_i FNR(i),$$

where δ_i is $FPR(i+1) - FPR(i)$, and $FPR(i)$ and $FNR(i)$ are the false positive rate and false negative rate for the i th parameter setting, respectively. Ω is the parameter space to be explored. Note that $FPR(i)$ is sorted in ascending order. AUC ranges from 0.0 (worst) to 1.0 (best).

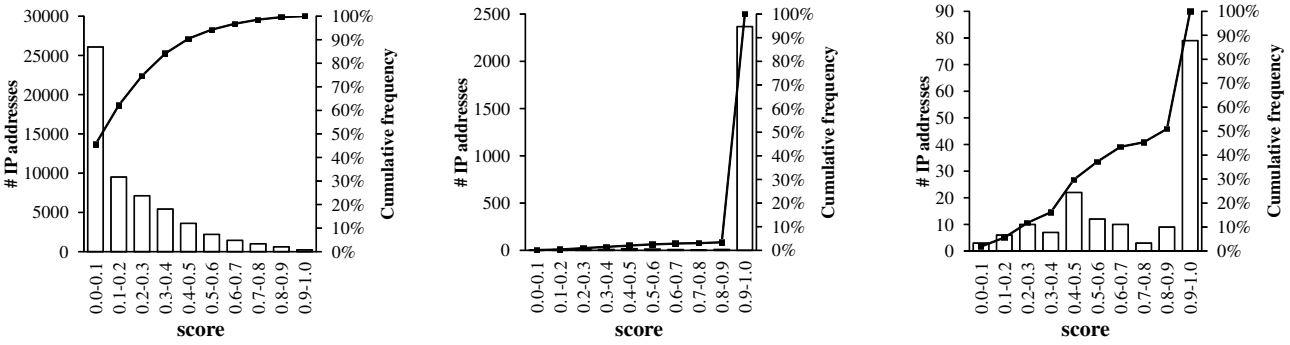
C. Experiment 1: Confirming the Locality of Malicious IP Addresses

We project IP addresses on a Hilbert curve [29] to visually confirm the locality of malicious IP addresses. The Hilbert curve is a recursively defined space-filling curve. A space-filling curve maps points into a d-dimensional space so that the closeness among the points is preserved. In this study, any consecutive IP addresses will be projected onto a single contiguous part on the curve [22], [30]. This technique has also been applied in some existing research to visualize IP address positions of the entire IPv4 address space. For example, Hao et al. [22] utilized the Hilbert curve to display the IP addresses of spam mail senders and Cai et al. [30] showed block-level IP address usage patterns using the Hilbert curve. Fig. 6 visualizes the IP addresses of x.0.0.0/14, the first octet of which is masked with x for security and projected into the Hilbert space using the training dataset in Table II. The white square represents a benign /24 network block, and the black square represents a malicious /24 network block. Fig. 6 demonstrates that the IP addresses of malicious websites are concentrated in certain network blocks.

D. Experiment 2: Comparing the Detection Performance of Feature Extraction Methods

This experiment evaluates the detection performance of our feature extraction methods. As an implementation of SVM, we use LIBSVM [24], which is currently one of the most widely used SVM software tool in many areas. In this experiment, we consider $0.0 \leq score < 0.5$ as benign and $0.5 \leq score \leq 1.0$ as malicious, which is equivalent to the cut-off point 0.5. This is the default configuration of binary classification using SVM. We evaluated our method with two kinds of test dataset combinations.

The first set of tests was conducted with the test dataset combination TST_B and TST_M_ACTIVE. Table V shows



(a) TST_B (b) TST_M_ACTIVE (c) TST_M_NEW

Figure 9. Histograms of Scores with the Test Dataset TST_B, TST_M_ACTIVE, and TST_M_NEW.

the evaluation results and Fig. 7 shows the ROC curves. In the Octet method, the accuracy increases to 0.885, AUC increases to 0.989, and FPR decreases to 0.119, as N increases to 3, where N is the parameter for the Octet method. However, when $N = 4$, the accuracy and AUC decrease and FPR increases. FNR decreases linearly with increasing N. ROC curves improve as N increases to 3. In particular, the ROC curves for both $N = 3, 4$ are closer to the upper left corner and have the potential to be ideal for binary classification. In the ExOctet method, which prioritizes the upper octets of IP addresses, the best results obtained are as follows: accuracy of 0.905, AUC of 0.994, and FPR of 0.098. The obtained ROC curve is also better than that with all other methods. In the Bit method, the accuracy increases to 0.857 and FPR decreases to 0.148 as k increases to 24, where k is the parameter for the Bit method. However, when $k = 32$, the accuracy decreases and FPR increases. AUC increases linearly with increasing k, whereas FNR decreases linearly with increasing k. The ROC curves improve as k increases and the curves obtained when $k = 24, 32$ are excellent results. These test results indicate that the top three octets of IP addresses provide an effective discriminative ability. The reason is believed to be that IP address allocation policies [31] provide locality to IP addresses and most of the networks are based on the /24 subnet.

The second set of tests was conducted with the test dataset combination of TST_B and TST_M_NEW. Table VI shows the evaluation results and Fig. 8 shows the ROC curves. Note that TST_M_NEW dataset does *not* contain any IP addresses that are used in the training stage as *known* IP addresses. We first notice that our methodology successfully detects *unknown* malicious IP addresses that could be missed by the existing blacklists. Now we look into the details of the differences among the feature selections. In the Octet method, the accuracy increases to 0.880, AUC increases to 0.897, FPR decreases to 0.119, and FNR increases to 0.335 as parameter N increases to 3. However, when $N = 4$, the accuracy, AUC, and FNR decrease and FPR increases. The ROC curves improve as N increases to 3. However, when $N = 4$, the ROC curve worsens slightly. In the ExOctet method, an accuracy of 0.902, AUC of 0.914, and FPR of 0.098 are the best results. The ROC curve is also the best result among all the methods. In the Bit method, the accuracy increases to 0.851, AUC increases to 0.893, and FPR decreases to 0.148 as parameter k increases to 24. However, when $k = 32$, the accuracy and AUC decrease and FPR increases. FNR is the lowest when $k = 24$, whereas the ROC curve is the best when $k = 24$. The reason for the best results when $N = 3$ or $k = 24$ is considered to be the same as that of the first set of tests, i.e., most of the networks are based on the /24 subnet.

For comparing the evaluation results of the two kinds of test dataset combinations, FNR with tests using TST_M_NEW is higher than that using TST_M_ACTIVE. Furthermore, the ROC curves for the test using TST_M_NEW is worse than that obtained using TST_M_ACTIVE. This is because TST_M_NEW contains only IP addresses that are sufficiently valid for the evaluation of our method, as described in Section IV-A. It should be noted that our method can classify unknown malicious IP addresses with high accuracy and low FPR. These results prove that IP addresses are effective indicators for determining whether a website is malicious.

E. Experiment 3: Evaluating the Distribution of the Score

This experiment evaluates the distribution of the scores assigned to IP addresses by our method. Fig. 9 shows histograms with cumulative frequency curves illustrating the distribution of scores for each test dataset: TST_B, TST_M_ACTIVE, and TST_M_NEW, with the ExOctet ($N = 3$) model showing the best result in Experiment 2.

Fig. 9 shows that for TST_B, our method assigns a low score to benign IP addresses. Moreover, for TST_M_ACTIVE, our method gives a significantly high score to malicious IP addresses that were active during

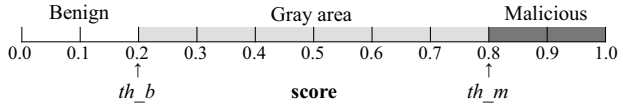


Figure 10. Example of the Relationship between Score and Two Kinds of Thresholds.

Table VII
RESULT WHEN SETTING TWO KINDS OF THRESHOLDS WITH THE TEST DATASET COMBINATION TST_B AND TST_M_ACTIVE

th_b	th_m	# IP addresses in gray area	# IP addresses of False Positives	# IP addresses of False Negatives
0.5	0.5	0	5,476	50
0.4	0.6	5,824	3,282	33
0.3	0.7	12,696	1,847	24
0.2	0.8	20,840	842	9
0.1	0.9	30,965	233	3

Table VIII
RESULT WHEN SETTING TWO KINDS OF THRESHOLDS WITH THE TEST DATASET COMBINATION TST_B AND TST_M_NEW

th_b	th_m	# IP addresses in gray area	# IP addresses of False Positives	# IP addresses of False Negatives
0.5	0.5	0	5,476	48
0.4	0.6	5,829	3,282	26
0.3	0.7	12,701	1,847	19
0.2	0.8	20,839	842	9
0.1	0.9	30,964	233	3

the observation term. This means that our method can accurately determine whether an IP address is malicious. TST_M_NEW shows that our method can assign a high score to even unknown malicious IP addresses. As described in Section IV-A, the first, second, and third octets of IP addresses in TST_M_NEW do not match any IP addresses in the training dataset TRN_M. This indicates that the IP addresses in TST_M_NEW first appeared during the observation period.

It is possible to decrease FNR and FPR by setting two kinds of thresholds: th_b and th_m. th_b determines whether an IP address is benign and th_m determines whether an IP address is malicious. An IP address whose score is lower than or equal to th_b is considered benign, whereas an IP address whose score is greater than or equal to th_m is considered malicious. An IP address whose score is between th_b and th_m is classified as *gray* area. Fig. 10 shows an example of the relationship between the score and the two thresholds when $th\_b = 0.2$ and $th\_m = 0.8$, i.e., $0.2 < score < 0.8$ is gray area.

Table VII shows the result when setting the two kinds of thresholds for the test dataset combination TST_B and TST_M_ACTIVE. Table VIII shows the result with the two kinds of thresholds for the test dataset combination TST_B and TST_M_NEW. Tables VII and VIII show that there were 5,476 false positives when $th\_b = th\_m = 0.5$. Extending the gray area decreases the number of IP addresses that result in false positives or false negatives. As an example, consider the case that $th\_b = 0.2$ and $th\_m = 0.8$. Our method does not determine whether the IP address for which the score is $0.2 < score < 0.8$ is benign or malicious but classifies it as gray area. This decreases the likelihood of false positives and negatives; however, there are a number of gray IP addresses that cannot be determined as benign or malicious. These results indicate that our method can control these two kinds of thresholds to decrease the number of false positives and negatives.

Table IX
SNORT ALERTS IN BENIGN DATASET TST_B

Snort alert	# alerts
SHELLCODE x86 inc ecx NOOP	272,180
SHELLCODE x86 NOOP	811
SHELLCODE base64 x86 NOOP	585
SHELLCODE x86 inc ebx NOOP	425
SHELLCODE x86 setuid 0	18
SHELLCODE x86 setgid 0	11
# total Snort alerts	274,030
# flows inspected by Snort	145,985,724

Table X
SNORT ALERTS IN MALICIOUS DATASET D3M

Snort alert	# alerts
SHELLCODE x86 NOOP	40
SHELLCODE x86 inc ebx NOOP	1
SHELLCODE x86 inc ecx NOOP	1
POLICY Suspicious .cn DNS query	2
# total Snort alerts	44
# URLs in D3M data	840
# flows inspected by Snort	11,393

F. Experiment 4: Applying Conventional IDS to Our Data

In this experiment, we apply conventional IDS to two kinds of data. The purpose of this experiment is to compare the detection performance of our method with that of conventional IDS to demonstrate the superiority of the proposed method. We choose the latest Snort version [16] and its rule sets. Snort is one of the most famous open source signature-based IDS. The rule sets consisted of only *Priority 1* rules, which indicate the most serious attacks.

The first test was conducted with our benign dataset TST_B. As mentioned in Section IV-A, TST_B contains benign HTTP traffic data. However, as shown in Table IX, a number of false positive alerts were output from Snort. Table IX lists many shellcode-related alerts. The reasons for these alerts are broadly classified into two categories: upload of executable files and HTTP compression. Upload of executable files to online file storage services or web-

based e-mail services is considered a shellcode-related alert by Snort. HTTP compression, which reduces the file size of HTTP data by using gzip and deflate algorithms [32], also outputs shellcode-related alerts.

The second test was conducted with *D3M 2010* in *MWS 2010 Datasets* [33] and *D3M 2011* in *MWS 2011 Datasets* [34]. These datasets were provided by the Malware Workshop Datasets project in Japan to facilitate data analysis in the security research area. The D3M 2010 and D3M 2011 datasets contained malicious communication data when the client honeypot Marionette [2] accessed URLs selected from MDL [28]. The 840 accessed URLs are a subset of the 10,000 URLs in our malicious training dataset TRN_M. Table X lists three types of shellcode-related alerts and suspicious DNS query alert output from Snort.

These two kinds of test results indicate that Snort IDS is practically incapable of detecting malicious websites because there were too many false positive alerts. Comparison of Table IX with Table X reveals that the same types of alerts are output for both benign and malicious data. In real environments, there is access to both benign and malicious websites. Therefore, the detection of malicious websites with Snort IDS is not readily performed, as demonstrated in this experiment. In contrast, our method can detect malicious websites with high accuracy and has a low false positive rate, as described in Sections IV-D and IV-E.

G. Analysis of False Positives in Our Method

To analyze why our method incorrectly determines some of the actually benign IP addresses as malicious, namely false positives, we consider the test dataset TST_B that consists of benign IP addresses. The targets of the analysis are the IP addresses in TST_B, whose score is in the top 100. The analysis shows that 83 IP addresses were used for websites on hosting services, 2 IP addresses were used for CDN, and the other 18 IP addresses were not used at that time and could not be investigated. This result indicates that because it uses only IP addresses, our method is likely to falsely regard benign IP addresses used on hosting services as malicious. When at least one of the websites deployed on a hosting service is malicious, the other benign websites are falsely considered as malicious, namely FP, although they are benign.

V. Conclusion

In this study, we have developed and evaluated a new method to detect malicious websites by learning the IP address features of both benign and malicious websites. Our experimental results have indicated that features extracted from only IP addresses are distinct indicators that enable us to compensate for the limitation of existing approaches; i.e., our scheme can detect even *unknown* malicious websites with low errors. However, our method incorrectly considers some benign websites as malicious mainly because they are on the same web hosting services that are utilized by malicious websites. This warrants future work on our method for evaluating malicious websites on hosting services more precisely. Nonetheless, the newly developed method will allow us to significantly enhance the detection performance when applied to existing network security systems.

References

[1] M. A. Rajab, L. Ballard, N. Jagpal, P. Mavrommatis, D. Nojiri, N. Provos, and L. Schmidt, "Trends in circumventing web-malware detection," Google, Google Technical Report, Jul. 2011.

[2] M. Akiyama, M. Iwamura, Y. Kawakoya, K. Aoki, and M. Itoh, "Design and implementation of high interaction client honeypot for drive-by-download attacks," *IEICE Transactions on Communications*, vol. E93.B, no. 5, pp. 1131–1139, 2010.

[3] Sharon Vaknin, "How to avoid, remove Facebook malware," http://howto.cnet.com/8301-11310_39-20070931-285/how-to-avoid-remove-facebook-malware/, Jun. 2011.

[4] Uribl. [Online]. Available: http://www.uribl.com/

[5] Opendns. [Online]. Available: http://www.opendns.com/

[6] Smartscreen filter. [Online]. Available: http://windows.microsoft.com/en-US/internet-explorer/products/ie-9/features/smartscreen-filter/

[7] Google safe browsing api. [Online]. Available: http://code.google.com/intl/en/apis/safebrowsing/

[8] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for dns," in *Proceedings of the 19th USENIX conference on Security*, ser. USENIX Security'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 18–18.

[9] M. Felegyhazi, C. Kreibich, and V. Paxson, "On the potential of proactive domain blacklisting," in *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, ser. LEET'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 6–6.

[10] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 1245–1254.

[11] K. Sato, K. Ishibashi, T. Toyono, and N. Miyake, "Extending black domain name list by using co-occurrence relation between dns queries," in *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, ser. LEET'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 8–8.

[12] C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert, "Zozzle: fast and precise in-browser javascript malware detection," in *Proceedings of the 20th USENIX conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 3–3.

[13] M. Akiyama, T. Yagi, and M. Itoh, "Searching structural neighborhood of malicious urls to improve blacklisting," in *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on*, Jul. 2011, pp. 1 –10.

[14] S. Shin and G. Gu, "Conficker and beyond: a large-scale empirical study," in *Proceedings of the 26th Annual Computer Security Applications Conference*, ser. ACSAC '10. New York, NY, USA: ACM, 2010, pp. 151–160.

[15] V. Paxson, "Bro: a system for detecting network intruders in real-time," in *Proceedings of the 7th conference on USENIX Security Symposium - Volume 7*. Berkeley, CA, USA: USENIX Association, 1998, pp. 3–3.

[16] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX conference on System administration*, ser. LISA '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 229–238.

[17] M. Ishida, H. Takakura, and Y. Okabe, "High-performance intrusion detection using optigrid clustering and grid-based labelling," in *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on*, Jul. 2011, pp. 11 –19.

[18] C. Seifert, I. Welch, P. Komisarczuk *et al.*, "Honeyc-the low-interaction client honeypot," *Proceedings of the 2007 NZCSRCS, Waikato University, Hamilton, New Zealand*, 2007.

[19] Capture-hpc. [Online]. Available: https://projects.honeynet.org/capture-hpc/

[20] Y. Niu, Y. min Wang, H. Chen, M. Ma, and F. Hsu, "A quantitative study of forum spamming using contextbased analysis," in *In Proc. Network and Distributed System Security (NDSS) Symposium*, 2007.

[21] A. Ramachandran and N. Feamster, "Understanding the network-level behavior of spammers," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '06. New York, NY, USA: ACM, 2006, pp. 291–302.

[22] S. Hao, N. A. Syed, N. Feamster, A. G. Gray, and S. Krasser, "Detecting spammers with snare: spatio-temporal network-level automatic reputation engine," in *Proceedings of the 18th conference on USENIX security symposium*, ser. SSYM'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 101–118.

[23] M. P. Collins, T. J. Shimeall, S. Faber, J. Janies, R. Weaver, M. De Shon, and J. Kadane, "Using uncleanliness to predict future botnet addresses," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 93–104.

[24] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, May. 2011.

[25] J. C. Platt, *Fast training of support vector machines using sequential minimal optimization*. Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.

[26] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *ADVANCES IN LARGE MARGIN CLASSIFIERS*. MIT Press, 1999, pp. 61–74.

[27] Alexa. [Online]. Available: http://www.alexa.com/topsites/

[28] Malware domain list. [Online]. Available: http://malwaredomainlist.com/

[29] T. Asano, D. Ranjan, T. Roos, E. Welzl, and P. Widmayer, "Space-filling curves and their use in the design of geometric data structures," *Theoretical Computer Science*, vol. 181, no. 1, pp. 3 – 15, 1997.

[30] X. Cai and J. Heidemann, "Understanding block-level address usage in the visible internet," in *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 99–110.

[31] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg, and J. Postel, "Internet registry ip allocation guidelines," RFC 2050, 1996.

[32] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol – http/1.1," RFC 2616, 1999.

[33] M. Hatada, Y. Nakatsuru, M. Akiyama, and S. Miwa, "Datasets for anti-malware research mws 2010 datasets," in *Proceedings of IPSJ Computer Security Symposium 2010 (CSS2010)*, vol. 2010, Oct. 2010, pp. 19–21, (in Japanese).

[34] M. Hatada, Y. Nakatsuru, and M. Akiyama, "Datasets for anti-malware research mws 2011 datasets," in *Proceedings of IPSJ Computer Security Symposium 2011 (CSS2011)*, vol. 2011, Oct. 2011, pp. 1–5, (in Japanese).