Regular Paper

Analyzing Spatial Structure of IP Addresses for Detecting Malicious Websites

Daiki Chiba[1,a)] Kazuhiro Tobe[1,b)] Tatsuya Mori[2,c)] Shigeki Goto[1,d)]

Received: October 15, 2012, Accepted: March 1, 2013

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 one of several 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 for new malicious websites. To tackle this problem, this paper proposes a new scheme 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 URLs and DNS records. While the strings that form URLs or DNS records are highly variable, IP addresses are less variable, i.e., IPv4 address space is mapped onto 4-byte strings. In this paper, a lightweight and scalable detection scheme that is based on machine learning techniques is developed and evaluated. 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. The effectiveness of our approach is validated by using real IP address data from existing blacklists and real traffic data on a campus network. The results demonstrate that our scheme can expand the coverage/accuracy of existing blacklists and also detect unknown malicious websites that are not covered by conventional approaches.

Keywords: Web/Mail security, computer viruses, machine learning, IP address, drive-by-download attacks

1. 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 [2]. These vulnerabilities are exploited by compromising the browser so that malware is downloaded and run on the targeted system. Such attacks are often called *drive-by-download* attacks [3].

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. **Figure 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.

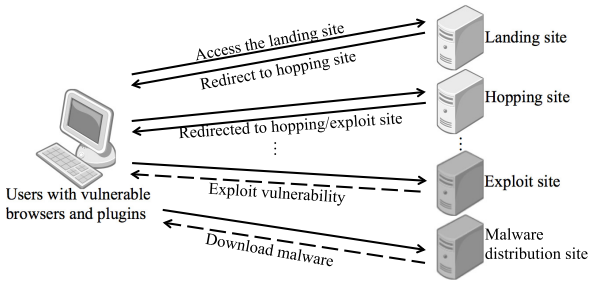


Fig. 1 Procedure of a drive-by-download attack.

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 [4]. 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,

[1] Department of Computer Science and Engineering, Waseda University, Shinjuku, Tokyo 169–8555, Japan
[2] NTT Network Technology Laboratories, NTT Corporation, Musashino, Tokyo 180–8585, Japan
a) chiba@goto.info.waseda.ac.jp
b) tobe@goto.info.waseda.ac.jp
c) mori.tatsuya@lab.ntt.co.jp
d) goto@goto.info.waseda.ac.jp
An earlier version of this paper was presented at IEEE/IPSJ 12th International Symposium on Applications and the Internet (SAINT 2012) [1].

which is discussed in Section 2. To resolve this problem, this paper presents a new scheme for detecting malicious websites using the characteristics of IP addresses.

This paper proposes a scheme for detecting communication associated with web-based malware using the 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 records. While the strings that form URLs or DNS records are highly variable, IP addresses are less variable, i.e., IPv4 address space is mapped onto 4-byte strings. In this paper, a lightweight and scalable detection scheme that is based on machine learning techniques is developed and evaluated. 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. The effectiveness of our approach is validated by using real IP address data from existing blacklists and real traffic data on a campus network. The results demonstrate that our scheme accurately differentiates between the IP addresses used for benign websites and malicious websites. In addition, this paper illustrates that our scheme 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. First, Section 2 reviews related works and discusses their limitations. Next, our detection scheme is presented in Section 3. Then, Section 4 illustrates the experimental results using actual web traffic data collected on a large-scale campus network. Finally, Section 5 concludes our study.

2. Related Work

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

2.1 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 blacklist or blocklist (DNSBL) [5], [6], [7] and commercial security appliances. Client-side filtering can be included in current web browsers [8], [9].

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. [10] developed a dynamic reputation system called *Notos*. The system collects information from multiple sources such as DNS zones, border gateway protocol prefixes, and Autonomous System (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. [11] proposed domain-based proactive blacklisting. It utilizes a small set of known malicious domains to predict other malicious domains using registration and name server information. Ma et al. [12] proposed a supervised learning approach for classifying URLs as benign or malicious using both lexical structure of URLs and host-based features such as WHOIS records and geographical information. Yadav et al. [13] focused on algorithmically generated malicious domain names and found that such domains were quite different from legitimate ones. They utilized this characteristic to develop their detection method based on statistical learning.

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 [14], [15], [16]. Furthermore, it has been reported that attackers frequently change domain names to evade detection by reputation systems [2]. In addition, Shin et al. [17] 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.

2.2 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 [18] and Snort [19] 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 the potential for detecting unknown attacks [20]. 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 shown later in Section 4.5, 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.

2.3 Client Honeypots

Recent studies have proposed *client honeypot systems* that aim to detect and analyze drive-by-download attacks [3], [21], [22], [23], [24]. 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 [21] include an emulator of a browser to crawl websites. Therefore, there is no risk that honeypots themselves will be infected with malware. High-interaction honeypots such as Capture-HPC [22] and BLADE [23] 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. proposed a state-of-the-art high-interaction honeypot called *Marionette* [3], [24] 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 [16], whose URLs change within a short time period [15]. 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 [25]. This cloaking makes it more difficult for client honeypots to crawl and detect malicious websites [2].

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

3. Detection Scheme

In this section, a high-level overview of our detection scheme is presented first in Section 3.1. Next, the effectiveness of our approach is shown in Section 3.2. Then, Section 3.3 illustrates several feature extraction techniques. Finally, Section 3.4 shows how a machine learning approach can be applied to our detection scheme.

3.1 High-level Overview

Our detection scheme is based on two intrinsic characteristics of IP addresses: (1) stability with time [26] and (2) address space skewness [27], [28]. First, although attackers can change URLs and DNS records at a low cost, it is much more difficult to change IP addresses, which are essentially associated with malicious activities. Thus, the characteristics of an IP address should be more stable compared with other metrics. Second, IP addresses associated with malicious activities are likely to be concentrated in certain network address spaces, as reported by previous measurement studies [27], [28]. In the following Section 3.2, our preliminary experiments show that IP addresses of web-based malware also have these characteristics. 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 Refs. [26], [27], [28].

Basically, our approach blocks users' access to malicious websites by extending existing blacklists. The unique feature of our scheme 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.

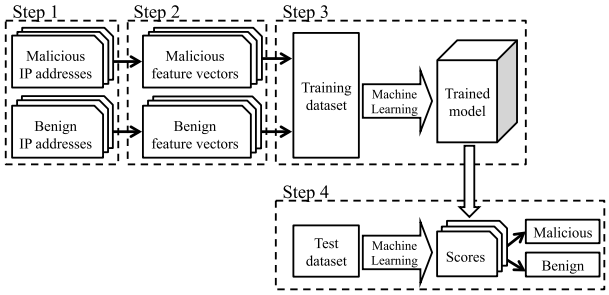


Fig. 2 Overview of our detection scheme.

Table 1 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

Our detection scheme 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 methods (step 2), which is described in Section 3.3. Section 3.4 shows our supervised machine learning technique that is employed for step 3 and step 4. **Figure 2** outlines our detection scheme.

3.2 Effectiveness of IP Address-based Approach

This section illustrates the effectiveness of our IP address-based approach to detect malicious websites. Two preliminary experiments using real IP address data are conducted to show stability with time in malicious networks and address space skewness.

3.2.1 Training Dataset

To evaluate the effectiveness of IP address-based approaches, IP addresses of both benign and malicious websites are collected. **Table 1** shows the collected training dataset. Note that this training dataset is also used for building a trained model in step 3, which is described later in Section 3.4.

Our benign training dataset TRN_B comprises URLs of the top 10,000 websites on the Alexa traffic ranking list [29] on April 30, 2011. From these URLs, 10,372 IP addresses are resolved. Domain names in the ranking list include those to which multiple IP addresses are assigned for load balancing, using DNS round robin and content delivery networks (CDNs). Therefore, the number of IP addresses in the ranking list exceeds the number of URLs, which correlate with domain names. The Alexa ranking is calculated from a combination of the average number of daily visitors and page views during the month [29]. Therefore, it contains both benign sites and less benign sites such as pornographic and file-sharing sites.

Our malicious training dataset TRN_M consists of IP addresses selected from the malware domain list (MDL) [30]. Note that MDL contains some malicious websites on web hosting servers,

Table 2 Top 5 malicious AS in TRN_M.

AS	#URLs	#IP addresses
AS #1	1,389	482
AS #2	2,422	400
AS #3	1,061	355
AS #4	761	280
AS #5	1,047	275

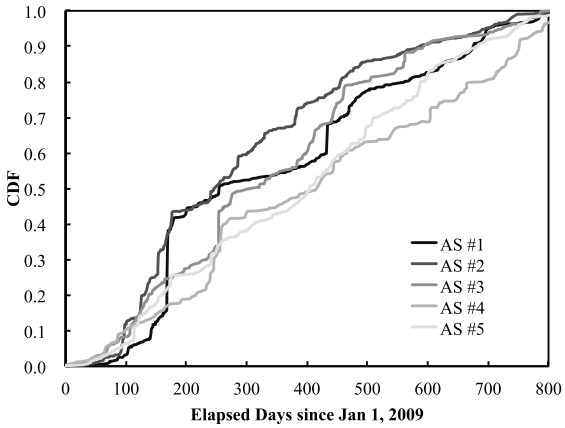


Fig. 3 Lifetime of malicious AS in TRN_M.

which utilize the same IP address for multiple domain names. Therefore, the number of URLs is greater than that of IP addresses. 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.

3.2.2 Stability with Time in Malicious Networks

To evaluate stability with time in malicious networks, malicious IP addresses of TRN_M as shown in Table 1 are analyzed. In this preliminary experiment, IP addresses are treated per AS. **Table 2** shows analyzed AS, whose AS numbers are masked for security. From the day when IP addresses are observed in TRN_M, the elapsed days since Jan 1, 2009 are calculated. **Figure 3** represents the cumulative distribution function (CDF) of IP addresses observed in each AS. This result illustrates that malicious IP addresses tend to be contained in certain AS. Moreover, such AS is continuously used for malicious activities for over two years.

Unlike IP addresses, malicious URLs change within a short time period [15]. For example, more than 60% of URLs contained in TRN_M or the malware domain list (MDL) were active for less than one month [16]. Furthermore, attackers create a lot of new malicious URLs one right after the other to avoid being blacklisted [2], [16]. Our analysis indicates that IP addresses are more stable than URLs in terms of time, i.e., IP addresses have less variability over time.

3.2.3 Address Space Skewness

In this preliminary experiment, IP addresses are projected on a Hilbert curve [31] to visually confirm the locality of malicious IP addresses. Hilbert curve is a recursively defined space-filling curve. A space-filling curve maps points on to 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 [27], [32]. **Figure 4** shows an example of IP address mapping on a Hilbert curve. In Fig. 4, the

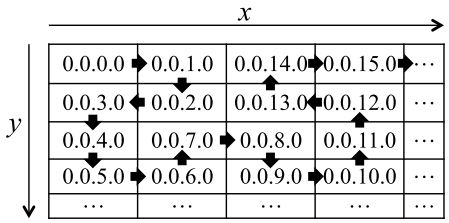


Fig. 4 Example of IP address mapping on a Hilbert curve.

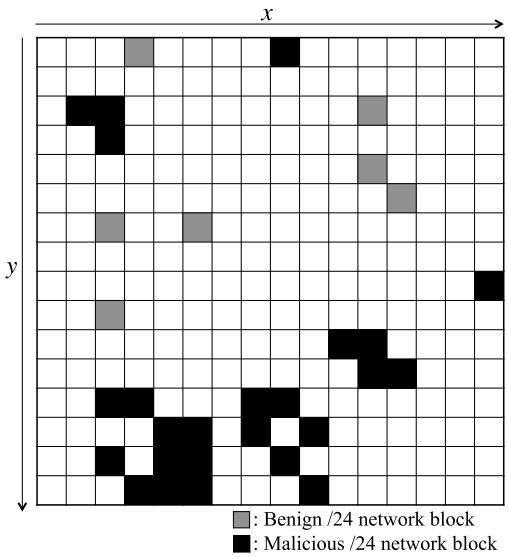


Fig. 5 Visualization of IP addresses on A.B.0.0/16.

squares are ordered according to the Hilbert curve where a single square represents a /24 network block. 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. [27] utilized the Hilbert curve to display the IP addresses of spam mail senders and Cai et al. [32] showed block-level IP address usage patterns using the Hilbert curve. **Figure 5** visualizes the IP addresses of A.B.0.0/16, the first and the second octets of which are masked with A and B for security and projected into the Hilbert space using the training dataset in Table 1. The gray square represents a benign /24 network block, and the black square represents a malicious /24 network block. Figure 5 demonstrates that the IP addresses of malicious websites are concentrated in certain network blocks.

3.3 Feature Extraction Methods

As described in Fig. 2, step 2 projects an IP address space onto a feature space that reflects the relationship between IP addresses and the network structure for interpretation 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

IPv4 address: 198.51.100.88

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

N=1 N=2 N=3 N=4

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

Fig. 6 Example of Octet-based Extraction (Octet).

is represented by the following equation:

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

where k is an index set of feature vectors.

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. **Figure 6** shows an example of feature vectors corresponding to the IPv4 address 198.51.100.88 with Octet. The upper half of Fig. 6 illustrates the correspondence between the 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. 6 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:

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

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 $N = 3$. **Figure 7** shows an example of feature vectors corresponding to the IPv4 address 198.51.100.88 with ExOctet. The upper half of Fig. 7 illustrates the correspondence of the extended coverage that is different from that of Octet. The lower half of Fig. 7 lists feature vectors extracted when $N = 3$. A feature vector uses the first, second, and third octets of an IP address when

IPv4 address: 198.51.100.88

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

$(X_1 + X_2) \bmod 2^8$ $(X_1 + X_2 + X_3) \bmod 2^8$

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

Fig. 7 Example of Extended Octet-based Extraction (ExOctet).

IPv4 address: 198.51.100.88

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

k=8 k=16 k=24 k=32

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

Fig. 8 Example of Bit String-based Extraction (Bit).

$N = 3$, which is the same as Octet. Moreover, a feature vector is extended with a combination of the first and second octets, and 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 the 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 the first k bits of a binary-formatted IPv4 address. A feature vector is represented according to the following equation:

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

Let $b_k = 1$ when the k-th bit value of binary-formatted IPv4 address is 1. **Figure 8** shows an example of feature vectors corresponding to the IPv4 address 198.51.100.88 with Bit. The upper half of Fig. 8 illustrates the correspondence between parameter k and feature extraction coverage, and the lower half lists the feature vectors extracted by parameter k.

3.4 Application of Machine Learning

As shown in Fig. 2, step 3 applies machine learning to the feature vectors extracted in step 2. Several options exist for machine learning classifiers such as k-nearest neighbors (k-NN), neural networks, and support vector machines (SVMs) [33]. K-NN are methods for classifying test data based on proximity between test data and training data in the feature space. Due to their high calculation complexity, k-NN are not suitable for high-speed classification [33]. Also, typical k-NN cannot accurately handle high-

dimensional data such as our IP address-based feature spaces. Neural networks, especially feedforward multi-layer neural networks, are computational models that can be used for data classification. Neural networks need a lot of parameters to perform and they cannot find global optimum solutions [33]. SVM is a popular machine learning method for classification. Unlike k-NN, SVM can appropriately handle high-dimensional feature spaces. Moreover, the number of parameters in SVM is small and SVM can find a global optimum solution [33], [34]. In addition, it has been reported that SVM works very well for various problems in a lot of areas [34]. Our scheme is intended to be an accurate and lightweight detection method. Therefore, SVM is selected to demonstrate the effectiveness of our approach. As shown in Section 4, 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 into supervised classifiers.

Table 3 shows an example of a training dataset used in this study. The feature vectors in this example are extracted using Bit when $k = 16$. 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 $t_n \in \{-1(\text{benign}), +1(\text{malicious})\}$.

Now the way to train SVM classifiers using the training dataset is described below. SVM uses the concept of a *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 whose margin is maximized. **Figure 9** illustrates a conceptual image of SVM's feature space. 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 the transposed matrix of $\mathbf{w}$, $\phi(\mathbf{x})$ denotes the feature space transformation function, and β is a 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, sequential minimal optimization [35] is applied. For the kernel function, our scheme selects 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, our scheme performs grid search with cross-validation tests. To estimate the probability of SVM's binary output, the technique proposed by Platt in Ref. [36] is applied. 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:

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

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. The scores assigned to each IP address are used for controlling the risk of various errors, which are discussed in Section 4.4.

Table 3 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}
−1	192.0.2.1	{1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
−1	203.0.113.24	{1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0}
...	...	...

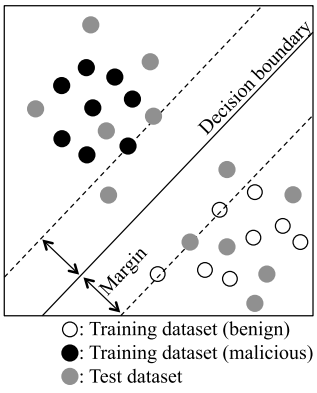


Fig. 9 Conceptual image of SVM's feature space.

4. 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.

4.1 Test Dataset

Test dataset is created from both benign and malicious websites, which include web traffic data captured on a campus network. **Table 4** shows the test dataset used to evaluate our scheme.

Our benign test dataset TST_B consists of HTTP traffic data captured on a campus network for two weeks in 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, all URLs in the captured data are checked with the Google Safe Browsing API [9]. Google Safe Browsing is constantly updated with blacklists of suspected phishing and malware related websites. As a result, 2,515 URLs that are suspected of being malicious sites are excluded.

Our malicious test dataset consists of IP addresses selected from the malware domain list (MDL) [30], which is the same

Table 4 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

data collection source as our malicious training dataset TRN_M. Note that the collection period of our malicious test dataset is different from that of TRN_M. The malicious test dataset are divided into two subsets: TST_M_ACTIVE and TST_M_NEW. 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, any IP addresses that are also contained in TRN_M are eliminated from TST_M_NEW. Moreover, from TST_M_NEW, any IP addresses for which the top three octets are equal to those of an IP address contained in TRN_M are eliminated. 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.

4.2 Evaluation Method

In the following experiments, the training dataset shown in Table 1 is used for the SVM's trained model as explained in Section 3.4. Then, our scheme is evaluated with the test dataset of Table 4.

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 5** shows the relationships among these terms.

Our scheme is evaluated 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 point. The cut-off point relates to *score*, as described in Section 3.4. 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 the overall test result is good. AUC is the area under the ROC curve that is used to score a 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).

Table 5 Relationships among terms.

		Actual value	
		Malicious	Benign
Prediction outcome	Malicious	True Positive (TP)	False Positive (FP)
	Benign	False Negative (FN)	True Negative (TN)

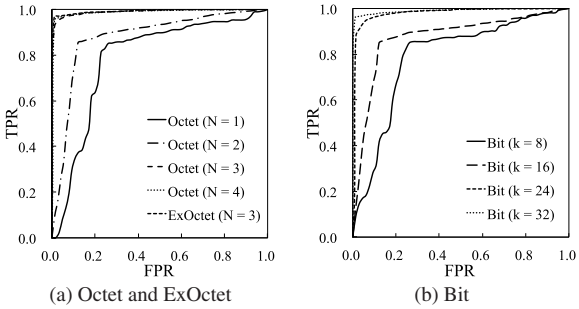
4.3 Experiment 1: Comparing the Detection Performance of Feature Extraction Methods

This experiment evaluates the detection performance of our feature extraction methods. As an implementation of SVM, our scheme uses LIBSVM [34], which is currently one of the most widely used SVM software tool for many disciplines. In this experiment, $0.0 \leq score < 0.5$ is considered 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. Our scheme is evaluated 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 6** shows the evaluation results and **Fig. 10** 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 those of 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 increas-

Table 6 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



(a) Octet and ExOctet (b) Bit

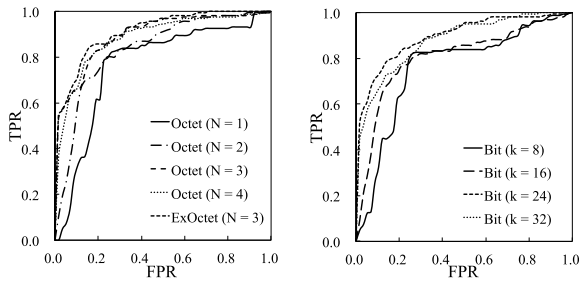
Fig. 10 ROC curves of detection performance with the test dataset combination TST_B and TST_M_ACTIVE.

ing 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. These test results indicate that the top three octets of IP addresses prove to be effective in discriminating between benign and malicious websites. We believe that the reasons are (1) IP address allocation policies [37] provide locality to IP addresses and (2) 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 7** shows the evaluation results and **Fig. 11** shows the ROC curves. Note that the TST_M_NEW dataset does *not* contain any IP addresses that are used in the training stage as *known* IP addresses. This paper first notices that our scheme successfully detects *unknown* malicious IP addresses that could be missed by existing blacklists. Now this test evaluates the performance in detecting unknown malicious data. 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.

Table 7 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

Fig. 11 ROC curves of detection performance with the test dataset combination TST_B and TST_M_NEW.

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.

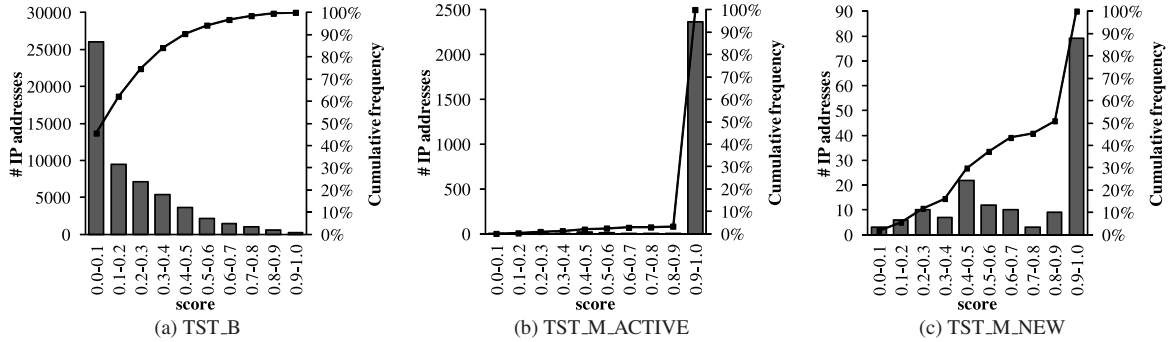
For comparing the evaluation results of the two kinds of test dataset combinations, FNR with tests using TST_M_NEW is higher than that using TST_M_ACTIVE. Furthermore, the ROC curves for the test using TST_M_NEW is worse than that obtained using TST_M_ACTIVE. This is because TST_M_NEW contains only IP addresses that are sufficiently valid for the evaluation of our scheme, as described in Section 4.1. It should be noted that our scheme 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.

4.4 Experiment 2: Evaluating the Distribution of the Score

This experiment evaluates the distribution of the scores assigned to IP addresses by our scheme. **Figure 12** 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 results in Section 4.3.

Figure 12 shows that for TST_B, our scheme assigns low scores to benign IP addresses. Moreover, for TST_M_ACTIVE, our scheme gives significantly high scores to malicious IP addresses that were active during the observation term. This means that our scheme can accurately determine whether an IP address is malicious. TST_M_NEW shows that our scheme can assign high scores to even unknown malicious IP addresses. As described in Section 4.1, 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.

Now, our scheme introduces multiple thresholds to control the risk of incorrect classification. In a real environment, the risk



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

Fig. 12 Histograms of scores with the test dataset TST_B, TST_M_ACTIVE, and TST_M_NEW.

should be configurable under various security policies such as minimizing the number of false positives and false negatives. This paper discusses a way to follow these security policies. For example, it is possible to decrease the number of false positives and false negatives 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 being in a *gray* area. **Figure 13** 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 the gray area.

Table 8 shows the result when setting the two kinds of thresholds for the test dataset combination TST_B and TST_M_ACTIVE. **Table 9** shows the result with the two kinds of thresholds for the test dataset combination TST_B and TST_M_NEW. Table 8 and Table 9 show that there were 5,476 false positives when $th\_b = th\_m = 0.5$. In this case, $th\_b = th\_m$ is equivalent to setting a single threshold without any gray areas. Extending the gray area decreases the number of IP addresses that result in false positives or false negatives. As an example, compare the cases between multiple thresholds ($th\_b = 0.2$ and $th\_m = 0.8$) and a single threshold ($th\_b = th\_m = 0.5$). When using multiple thresholds, our scheme 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 being in the gray area. This decreases the likelihood of false positives and false negatives compared to the case of a single threshold; however, there are a number of gray IP addresses that cannot be determined as being benign or malicious. These results indicate that our scheme can control these two kinds of thresholds to decrease the number of false positives and false negatives.

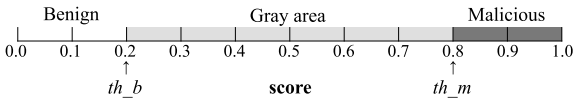


Fig. 13 Example of the relationship between score and two kinds of thresholds.

Table 8 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 9 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

4.5 Experiment 3: Applying Conventional IDS to Our Data

In this experiment, conventional IDS is applied to two kinds of data. The purpose of this experiment is to compare the detection performance of our scheme with that of conventional IDS to demonstrate the superiority of the proposed scheme. As a conventional IDS, the latest Snort [19] and its rule sets are selected. Snort is one of the most famous open source signature-based IDS. The Snort signatures applied in this experiment consisted of only eight *Priority 1* signatures as shown in **Table 10** and **Table 11**.

The first test was conducted with our benign dataset TST_B. As mentioned in Section 4.1, TST_B contains benign HTTP traffic data. However, as shown in Table 10, a number of false positive alerts were output from Snort. In this case, false positive means the incorrect classification of actually benign traffic into the malicious category by Snort. Table 10 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 [38], also outputs shellcode-related alerts.

The second test was conducted with the *D3M 2010* dataset from *MWS 2010* [39] and the *D3M 2011* dataset from *MWS 2011* [40]. These datasets were provided by the *anti Malware engineering WorkShop* (*MWS*) [41] project in Japan to facilitate data analysis in the security research area. The D3M datasets (D3M 2010 and D3M 2011) contained malicious communication data when the client honeypot Marionette [3] accessed URLs in MDL [30]. Marionette inspected websites corresponding to the URLs in MDL to determine whether they were malicious or not at the time of inspecting [16]. From inspected URLs, 840 URLs are selected for the D3M datasets by the MWS project [41]. Both

Table 10 Snort alerts in benign dataset TST_B.

Snort signature	# 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 11 Snort alerts in malicious dataset D3M.

Snort signature	# 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

the D3M datasets and our malicious training dataset TRN_M use the same MDL [30]. Therefore, the 840 URLs are a subset of the 63,694 URLs in TRN_M. Table 11 lists three types of shellcode-related alerts and suspicious DNS query alert output from Snort, but the number of correct detection by Snort is small.

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 10 with Table 11 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.

For reference, the 840 malicious URLs in the D3M datasets are tested by our scheme. Our scheme detects 805 URLs correctly, so the detection accuracy is 0.958 (= 805/840), and FNR is 0.042 (= 35/840). This result shows that our scheme can compensate for the limitations of conventional IDS.

4.6 Experiment 4: Comparing the Processing Time

This experiment compares the processing time between our scheme and existing approaches. The processing time is measured under the same Linux machine with an Intel Xeon 2.40 GHz CPU and 4 GB RAM. **Table 12** shows the results of this experiment.

In our scheme, the processing time is divided into training time and test time. Training time is the time for building a trained model using the training dataset shown in Table 1. Test time is the total time for checking all IP addresses in the D3M 2010 and D3M 2011 datasets shown in Section 4.5. Test speed is the number of processable addresses per second, which is calculated from test time. Note that our scheme does not need to build a trained model every time and the same trained model can be used repeatedly. In the Octet method, the training time and the test time increase linearly with increasing N. In the ExOctet method, the training time is shorter than that of the Octet ($N = 3$) method, but the test time is longer than that of the Octet ($N = 3$) method. In the Bit method, the training time and the test time increase linearly with increasing k.

Existing approaches include network-side blacklists and signature-based IDS. As a network-side blacklist, *rblcheck* [42] is used for measuring the test time for checking IP addresses in the D3M datasets. Rblcheck is a tool for performing lookups in multiple DNSBL services. Using more DNSBL services takes a longer time. Therefore, only one of the DNSBL services is selected for comparing the processing time fairly. As shown in Table 12, the test time of rblcheck is up to 41 times longer than that of our scheme. The reason is that rblcheck contacts external DNSBL servers to check IP addresses using the DNS protocol. As a signature-based IDS, Snort (see Section 4.5) was selected. The test time of Snort with the D3M datasets is longer than any of our schemes. It is because Snort is checking not only IP addresses but also the entire packet payload.

Table 12 Processing time with the test dataset D3M.

Method	Training [sec]	Test [sec]	Test speed [addresses/sec]
Octet ($N = 1$)	15.6	0.57	1486
Octet ($N = 2$)	18.1	0.59	1415
Octet ($N = 3$)	31.5	0.67	1249
Octet ($N = 4$)	44.2	1.12	752
ExOctet ($N = 3$)	24.8	0.78	1074
Bit ($k = 8$)	18.1	0.70	1194
Bit ($k = 16$)	24.7	0.73	1146
Bit ($k = 24$)	45.1	1.20	698
Bit ($k = 32$)	65.8	2.04	412
Rblcheck	–	23.60	36
Snort	–	3.98	211

For comparing the processing time, the test speed of our scheme is faster than that of existing approaches. Our scheme is based on only the IP address structure, so it is more lightweight than other approaches. Therefore, our scheme can be combined with existing approaches to compensate for their drawbacks.

4.7 Analysis of False Positives in Our Scheme

This section analyzes why our scheme incorrectly determines some of the actually benign IP addresses as malicious, namely false positives. 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 scheme 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 as FPs, although they are benign.

5. Conclusion

In this study, a new scheme to detect malicious websites by learning the IP address features has been developed and evaluated. Our experimental results have indicated that features extracted only from IP addresses are distinct indicators that enables us to compensate for the limitation of existing approaches; i.e., our scheme can detect even *unknown* malicious websites with low errors. However, our scheme 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 a more thorough investigation of our scheme for hosting services as part of our future work. Moreover, this paper considers only IPv4 addresses structure because the number of malicious websites using IPv6 addresses is much less than that of using IPv4 addresses at the present time of writing. Thus, applying our IPv4 address-based approach to IPv6 address environment will be our future work in the near future. Nonetheless, the newly developed scheme will allow us to significantly enhance the detection performance when applied to existing network security systems.

References

[1] Chiba, D., Tobe, K., Mori, T. and Goto, S.: Detecting Malicious Websites by Learning IP Address Features, *Proc. IEEE/IPSJ 12th International Symposium on Applications and the Internet* (*SAINT 2012*), pp.29–39 (2012).

[2] Rajab, M.A., Ballard, L., Jagpal, N., Mavrommatis, P., Nojiri, D.,

Provos, N. and Schmidt, L.: Trends in Circumventing Web-Malware Detection, Google Technical Report (2011).
[3] Akiyama, M., Iwamura, M., Kawakoya, Y., Aoki, K. and Itoh, M.: Design and Implementation of High Interaction Client Honeypot for Drive-by-Download Attacks, *IEICE Trans.*, Vol.E93.B, No.5, pp.1131–1139 (2010).
[4] Vaknin, S.: How to Avoid, Remove Facebook Malware (online), available from ⟨http://howto.cnet.com/8301-11310_39-20070931-285/how-to-avoid-remove-facebook-malware/⟩ (accessed 2011-07-01).
[5] Levine, J.: DNS Blacklists and Whitelists, RFC 5782 (2010).
[6] URIBL (online), available from ⟨http://www.uribl.com/⟩ (accessed 2011-05-14).
[7] OpenDNS (online), available from ⟨http://www.opendns.com/⟩ (accessed 2011-05-14).
[8] SmartScreen Filter (online), available from ⟨http://windows.microsoft.com/en-US/internet-explorer/products/ie-9/features/smartscreen-filter/⟩ (accessed 2011-05-14).
[9] Google Safe Browsing API (online), available from ⟨http://code.google.com/intl/en/apis/safebrowsing/⟩ (accessed 2011-05-14).
[10] Antonakakis, M., Perdisci, R., Dagon, D., Lee, W. and Feamster, N.: Building a Dynamic Reputation System for DNS, *Proc. 19th USENIX Conference on Security*, p.18 (2010).
[11] Felegyhazi, M., Kreibich, C. and Paxson, V.: On the Potential of Proactive Domain Blacklisting, *Proc. 3rd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More* (*LEET 2010*), p.6 (2010).
[12] Ma, J., Saul, L.K., Savage, S. and Voelker, G.M.: Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs, *Proc. 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD 2009*), pp.1245–1254 (2009).
[13] Yadav, S., Reddy, A.K., Reddy, A. and Ranjan, S.: Detecting Algorithmically Generated Malicious Domain Names, *Proc. 10th Annual Conference on Internet Measurement* (*IMC 2010*), pp.48–61 (2010).
[14] Sato, K., Ishibashi, K., Toyono, T. and Miyake, N.: Extending Black Domain Name List by Using Co-Occurrence Relation between DNS Queries, *Proc. 3rd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More* (*LEET 2010*), p.8 (2010).
[15] Curtsinger, C., Livshits, B., Zorn, B. and Seifert, C.: ZOZZLE: Fast and Precise In-Browser JavaScript Malware Detection, *Proc. 20th USENIX Conference on Security*, p.3 (2011).
[16] Akiyama, M., Yagi, T. and Itoh, M.: Searching Structural Neighborhood of Malicious URLs to Improve Blacklisting, *Proc. IEEE/IPSJ 11th International Symposium on Applications and the Internet* (*SAINT 2011*), pp.1–10 (2011).
[17] Shin, S. and Gu, G.: Conficker and Beyond: A Large-Scale Empirical Study, *Proc. 26th Annual Computer Security Applications Conference* (*ACSAC 2010*), pp.151–160 (2010).
[18] Paxson, V.: Bro: A System for Detecting Network Intruders in Real-Time, *Proc. 7th Conference on USENIX Security Symposium*, Vol.7, p.3 (1998).
[19] Roesch, M.: Snort - Lightweight Intrusion Detection for Networks, *Proc. 13th USENIX Conference on System Administration* (*LISA 1999*), pp.229–238 (1999).
[20] Ishida, M., Takakura, H. and Okabe, Y.: High-Performance Intrusion Detection Using OptiGrid Clustering and Grid-Based Labelling, *Proc. IEEE/IPSJ 11th International Symposium on Applications and the Internet* (*SAINT 2011*), pp.11–19 (2011).
[21] Seifert, C., Welch, I. and Komisarczuk, P.: HoneyC - the Low-Interaction Client Honeypot, *Proc. 2007 NZCSRCS*, Waikato University, Hamilton, New Zealand (2007).
[22] Capture-HPC (online), available from ⟨https://projects.honeynet.org/capture-hpc/⟩ (accessed 2011-05-31).
[23] Lu, L., Yegneswaran, V., Porras, P. and Lee, W.: BLADE: An Attack-Agnostic Approach for Preventing Drive-By Malware Infections, *Proc. 17th ACM Conference on Computer and Communications Security* (*CCS 2010*), pp.440–450 (2010).
[24] Akiyama, M., Kawakoya, Y. and Hariu, T.: Scalable and Performance-Efficient Client Honeypot on High Interaction System, *Proc. IEEE/IPSJ 12th International Symposium on Applications and the Internet* (*SAINT 2012*), pp.40–50 (2012).
[25] Niu, Y., Wang, Y., Chen, H., Ma, M. and Hsu, F.: A Quantitative Study of Forum Spamming Using Contextbased Analysis, *Proc. 14th Annual Network and Distributed System Security Symposium* (*NDSS 2007*) (2007).
[26] Ramachandran, A. and Feamster, N.: Understanding the Network-Level Behavior of Spammers, *Proc. 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (*SIGCOMM 2006*), pp.291–302 (2006).
[27] Hao, S., Syed, N.A., Feamster, N., Gray, A.G. and Krasser, S.: Detecting Spammers with SNARE: Spatio-Temporal Network-Level Automatic Reputation Engine, *Proc. 18th Conference on USENIX Security Symposium*, pp.101–118 (2009).
[28] Collins, M.P., Shimeall, T.J., Faber, S., Janies, J., Weaver, R., De Shon, M. and Kadane, J.: Using Uncleanliness to Predict Future Botnet Addresses, *Proc. 7th ACM SIGCOMM Conference on Internet Measurement* (*IMC 2007*), pp.93–104 (2007).
[29] Alexa Top Sites (online), available from ⟨http://www.alexa.com/topsites/⟩ (accessed 2011-05-01).
[30] Malware Domain List (online), available from ⟨http://malwaredomainlist.com/⟩ (accessed 2011-05-14).
[31] Asano, T., Ranjan, D., Roos, T., Welzl, E. and Widmayer, P.: Space-Filling Curves and Their Use in the Design of Geometric Data Structures, *Theoretical Computer Science*, Vol.181, No.1, pp.3–15 (1997).
[32] Cai, X. and Heidemann, J.: Understanding Block-Level Address Usage in the Visible Internet, *Proc. ACM SIGCOMM 2010 Conference*, pp.99–110 (2010).
[33] Bishop, C.M.: *Pattern Recognition and Machine Learning* (*Information Science and Statistics*), Springer (2006).
[34] Chang, C.C. and Lin, C.J.: LIBSVM: A Library for Support Vector Machines, *ACM Trans. Intelligent Systems and Technology* (*TIST*), Vol.2, No.3, pp.27:1–27:27 (2011).
[35] Platt, J.: Fast Training of Support Vector Machines Using Sequential Minimal Optimization, *Advances in Kernel Methods*, pp.185–208, MIT Press (1999).
[36] Platt, J.: Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods, *Advances in Large Margin Classifiers*, pp.61–74, MIT Press (1999).
[37] Hubbard, K., Kosters, M., Conrad, D., Karrenberg, D. and Postel, J.: Internet Registry IP Allocation Guidelines, RFC 2050 (1996).
[38] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1, RFC 2616 (1999).
[39] Hatada, M., Nakatsuru, Y., Akiyama, M. and Miwa, S.: Datasets for Anti-Malware Research MWS 2010 Datasets, *Proc. IPSJ Computer Security Symposium* (*CSS 2010*), pp.19–21 (2010) (in Japanese).
[40] Hatada, M., Nakatsuru, Y. and Akiyama, M.: Datasets for Anti-Malware Research MWS 2011 Datasets, *Proc. IPSJ Computer Security Symposium* (*CSS 2011*), pp.1–5 (2011) (in Japanese).
[41] Anti Malware Engineering Workshop 2012 (MWS 2012) (online), available from ⟨http://www.iwsec.org/mws/2012/en.html⟩ (accessed 2012-12-22).
[42] Rblcheck (online), available from ⟨http://sourceforge.net/projects/rblcheck/⟩ (accessed 2012-12-22).

Daiki Chiba was born in 1988. He received his B.E. degree in computer science and engineering from Waseda University in 2011. He is currently a 2nd-year master's student in the Department of Computer Science and Engineering, Waseda University. His research interest is network security.

Kazuhiro Tobe was born in 1987. He received his B.E. and M.E. degrees in computer science and engineering from Waseda University in 2010 and 2012, respectively. He had been engaged in the research of network application and network security.

Tatsuya Mori was born in 1973. He received his B.E. and M.E. degrees in applied physics, and Ph.D. from Waseda University in 1997, 1999 and 2005, respectively. He joined NTT in 1999 and became a senior researcher in 2010. He has been engaged in the research of understanding large-scale networked systems and network security. He is a member of IEICE, ACM, and IEEE.

Shigeki Goto was born in 1948. He received his M.S. degree from the University of Tokyo in 1973. He earned his Ph.D. degree from the University of Tokyo in 1991 while he worked at NTT Laboratories from 1973 to 1996. He became a professor at Waseda University in 1996. His research interest is computer networking, especially the Internet. He is the president of JPNIC. He is an IPSJ fellow and a member of IEICE, ACM, and IEEE.