Master's Thesis

Title

Detecting Drive-by Download Attacks from Proxy Log Information using Convolutional Neural Network

Supervisor
Professor Masayuki Murata

Author
Kohei Yamanishi

February 6th, 2017

Department of Information Networking
Graduate School of Information Science and Technology
Osaka University

Master's Thesis

Detecting Drive-by Download Attacks from Proxy Log Information using Convolutional Neural Network

Kohei Yamanishi

Abstract

Many hosts are still infected by drive-by download attacks despite the efforts of many security researchers and venders. In the drive-by download attacks, the attackers maliciously change popular web sites. Then, the users are redirected via the redirect URLs to the exploit URLs. At the exploit URLs, an exploit code is executed, and malware is downloaded from malware distribution URLs [1]. By using the redirections via multiple URLs, which is called a redirection chain, the attacker can separate functions such as redirections and attack. As a result, the attacker can easily change the URLs for the redirections and attack in a short time, which makes it difficult for researchers and vendors to analyize them [1].

Blacklists including URLs and domains related to malicious web sites are widely implemented [2]. The blacklists are created by using honeyclients [3], which have decoy browsers. However, the URLs used for attacks are frequently changed. Thus, there may be many web sites that are used by the drive-by download attacks but are not included in the blacklists.

When a drive-by download attack occurs, the corresponding URL sequence includes the redirection chain. In this thesis, we focus on the features of the URL sequences, including the features of malicious URLs and their order. We propose the method to detect drive-by download attacks using the convolutional neural network (CNN), which achieves high accuracy in the field of analyzing sequence data [4]. In addition to simply applying CNNs, we introduce an Event De-noising CNN (EDCNN) [5], which is a neural network extended from the CNN so as to mitigate the impact of the benign URLs included in the URL sequence.

To detect drive-by download attacks from the proxy logs, we need to consider the case that multiple web sites are accessed simultaneously. In this case, the URL sequence includes the URLs of multiple web sites. If one of the web sites included in the URL sequence is related to the drive-by download attacks, the sequence must be detected as malicious. However, such URL sequences are difficult to be identified as malicious by the CNN or EDCNN, because most of the URLs included in the sequence are not related to the drive-by download attacks. Therefore, we also propose a method to detect the drive-by download attacks even in such cases. In this method, we input the subsequences of the URL sequence to the CNN or the EDCNN. Then, if one of them is classified as malicious, the sequence is detected as malicious.

We evaluate our method by using the URL sequences collected by a honeyclient accessing popular/blacklisted web sites and the URL sequences collected at the gateway of our laboratory. The results show that the EDCNN using the subsequences achieves the true positive rate higher than 95 % even when URL sequences include URLs of multiple web sites. However, the EDCNN may cause the FPR higher than 39%. To improve the accuracy, we integrate multiple EDCNNs. The method integrating four EDCNNs achieves TPR higher than 97% and FPR lower than 20%.

Keywords

Drive-by Download

Convolutional Neural Network

URL Sequence

Proxy Log

Attack Detection

Contents

List of Figures

List of Tables

1 Introduction

Many hosts are still infected by drive-by download attacks despite the efforts of many security researchers and venders. In the drive-by download attacks, the attackers maliciously change popular web sites. Then, the users are redirected via the redirect URLs to the exploit URLs. At the exploit URLs, an exploit code is executed, and malware is downloaded from malware distribution URLs [1]. By using the redirections via multiple URLs, which is called a redirection chain, the attacker can separate functions such as redirections and attack. As a result, the attacker can easily change the URLs for the redirections and attack in a short time, which makes it difficult for researchers and vendors to analyize them [1]. Furthermore, in the redirect URLs, the client environment is identified by browser fingerprinting, then the attacker changes the next redirect URLs based on the identified client environment; if the client has the target vulnerability, it is redirected to the URLs related to the attacks. Otherwise, the client is redirected to the benign web sites [6]. This makes it difficult for the researchers and venders to collect malwares.

Blacklists including URLs and domains related to malicious web sites are widely implemented [2]. The blacklists are created by using honeyclients [3], which have decoy browsers. However, the URLs used for attacks are frequently changed. Thus, there may be many web sites that are used by the drive-by download attacks but are not included in the blacklists. Another approach to detecting the drive-by download attacks is to analyze the content of web sites [7]. However, this method requires the collection of the web content, which is unrealistic for network administrators because it requires a large amount of resources. On the other hand, access history information such as proxy logs is generally stored [8]. We can obtain a sequence of URLs accessed by a user from the logs, though the logs do not include the content of web sites. In this thesis, we call it a URL sequence. When a drive-by download attack occurs, the corresponding URL sequence includes the redirection chain. In this thesis, we focus on the features of the URL sequences, including the features of malicious URLs and their order.

A URL sequence consists of multiple URLs occurred by accessing web sites. Furthermore, a URL sequence includes many pairs of URLs related by redirections. This relation of URLs in URL sequences is similar to the relation of words in sentences. In the field of natural language processing such as analysis of sentences, the convolution neural network (CNN) achieves high accuracy [4]. Therefore, we use a CNN to detect drive-by download attacks.

In addition to simply applying CNNs, we introduce an Event De-noising CNN (EDCNN) [5], which is a neural network extended from the CNN so as to mitigate the impact of the benign URLs included in the URL sequence. To detect drive-by download attacks from the proxy logs, we need to consider the case that multiple web sites are accessed simultaneously. In this case, the URL sequence includes the URLs of multiple web sites. If one of the web sites included in the URL sequence is related to the drive-by download attacks, the sequence must be detected as malicious. However, such URL sequences are difficult to be identified as malicious by the CNN or EDCNN, because most of the URLs included in the sequence are not related to the drive-by download attacks. Therefore, we also propose a method that detects the drive-by download attacks by using the learned CNNs or EDCNNs even in such cases. In this methods, we input the subsequence to the CNNs or EDCNNs. By using the subsequences, we can detect drive-by download attacks accurately, because the URLs of the same web site tend to be close in the URL sequences. Therefore, one of the subsequences includes malicious redirections. As a result, the method using the subsequences detects the malicious URL sequences.

The rest of this thesis is organized as follows. In Section 2, we introduce the related work including the existing methods to detect malicious web sites, and the research on the neural networks. Section 3 explains details of the proposed method. Section 4 evaluates the proposed method. Section 5 concludes this thesis and future work.

2 Related Work

2.1 Malicious Website Detection

A honeyclient is a decoy system for visiting and detecting malicious web sites. There are two types of the honeyclient; high interactive honeyclients and low interactive honeyclients. High interactive honeyclients use actual browsers with vulnerabilities and detect malicious web sites by observing malicious processes and access to file systems. Capture-HPC [9] is a high interactive honeyclient, which monitors events such as access to files and registries, process control, and detects malicious content that violates predefined rules. However, most high interactive honeyclients depend on certain operating systems and applications, and they collect only the malicious web sites whose targets match the operation systems and applications installed in the honeyclients. On the other hand, low interactive honeyclients use browser emulators to detect malicious web sites using web content and various heuristics [10, 11].

Many methods to detect attacks focusing on web content have been proposed because malicious web sites includes redirection code and exploit code. Curtsinger et al. proposed Zoozle [7], which classifies malicious JavaScript using hierarchical structure features of abstract syntax trees. Canali et al. proposed Prophiler as a lightweight filter against malicious web sites. It identifies non-malicious web sites by HTML-, JavaScript-, URL-based features. Then, the web sites that are not identified as non-malicious are checked by the other method. However, this method requires the collection and analysis of web content, which requires a large amount of resources. On the other hand, our system can detect malicious URLs from only the sequence of the URLs accessed by users.

Antonakakis et al. proposed Notos [2], which finds unknown malicious domains focusing on the history of domain usage and their corresponding IP addresses. Ma et al. detected malicious web sites using the lexical structure of the URL of the phishing site [12]. On the other hand, our system uses URL sequences to detect by the latest learning method based on the redirection chain which is characteristic of the current drive-by download attacks.

There are also studies that detect malicious sites by reconstructing user traffic. Mekky et al. proposed a method to detect malicious web sites by constructing the redirection structures of the HTTP [13]. Taylor et al. proposed a system to check a large amount of traffic using a honeyclient and reduce the cost of analysis by caching web content and selecting only suspicious content [14].

WebWitness [15] is a system that traces the HTTP event of malware downloads. The purpose of this method is not to detect drive-by download attack, but it can reasonably label a malicious HTTP event. Because our system can detect malicious URL sequences without inspection of content, it plays a role differently from these studies.

2.2 Deep Neural Network

In recent years, deep neural networks have demonstrated high performance in many fields. A CNN is a feed-forward neural network composed of multiple convolution layers and max pooling layers. CNNs have demonstrated high performance in image recognition since Pereira et al. showed a lower misrecognition rate than other state-of-the-art methods at the time [16]. CNNs are also applied to sequential data for video classification [17] and natural language processing [4]. Borgolte et al. applied a CNN to the security field [18], but it was a CNN of image recognition that uses a webpage as an input image and detects alteration of the web site.

In this thesis, we applied CNNs to the URL sequences. Benign URLs exists between malicious URLs in the sequences, which is a noise that has an impact on the detection of a malicious URL sequence. Although CNNs which perform noise elimination of images have been proposed [18], they cannot be used in our study because the noise eliminated by this method is different from the noise that should be eliminated in our study; the noise of the URL sequence is inserted between the malicious URLs, while the noise is added to the true values in image. Therefore, we propose a CNN to handle such noise, which is the first CNN designed for security.

A recursive neural network (RNN) is another class of neural networks that has recurrent structures. In the computer security field, Shin et al. used an RNN for binary function recognition [19]. Because the structure of a CNN can be flexibly modified according to the characteristics of URL sequences compared to an RNN, we apply a CNN to detect malicious URL sequences.

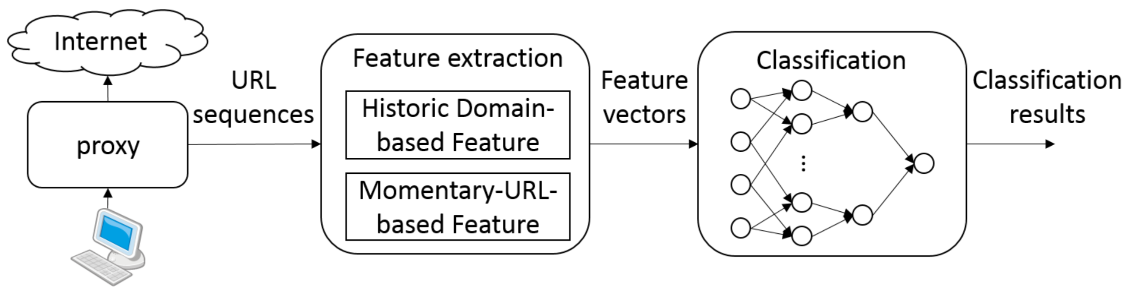

Figure 1: Overview of our system.

3 Drive-by Download Attacks Detection System

3.1 Overview

Fig. 1 shows the overview of our system. In this system, a URL sequence is extracted from the proxy log. Then, the features of the URLs in the sequence are extracted, and are used as input of the EDCNN classifier. The outputs of the classifier is malicious or benign.

Each step of our method will be explained in the rest of this section.

3.2 URL Sequence Extraction

The proxy logs the URLs accessed by the terminals, time stamps of the accesses, and the source terminals of the accesses. Our method extracts URL sequences from the proxy log. The extracted URL sequence should include all URLs of the redirection chain. However, the proxy log does not include the information related to the redirection chain each URL belongs to.

In this thesis, we divide the URLs in the proxy log into URL sequences by using the time stamp of the access to the URL. If a URL is redirected from another URL, the redirected URL is accessed soon after the first URL is accessed. On the other hand, when a user accesses a new web site, the time between the access to the web site and the access to another web site is long. Therefore, in this thesis, we regard the set of the URLs that are accessed from the same terminal without large time intervals as a URL sequence. There may be a service that continuously access the URLs. We set the maximum time length of a URL sequence, considering such cases. That is, we terminate the URL sequence when the predefined time is elapsed from the first URL access of the URL sequence.

3.3 Feature Extraction

In this thesis, we use neural networks to classify the URL sequences. The input of the neural networks should be a vector. Therefore, we vectorize the URL sequence.

The features of the URLs used by attackers have been proposed. There are two kinds of features, historic domain-based features [2,20] and momentary URL-based features [12,21]. Notos [2] uses historic domain-based features based on information extracted from Related Historic IP addresses (RHIP) and Related Historic Domain Names (RHDN). RHIP is the set of all resolved IP addresses for each fully qualified domain name (FQDN), its third-level domain (3LD) part, and its second-level domain (2LD) part in the past. The features such as the number of related BGP prefixes, the number of applicable IP addresses, the number of related ASs and so on are extracted from RHIP. On the other hand, RHDN means a set of resolving to IP addresses in the same ASN of the past IP addresses of each FQDN. The features such as the number of domains, the number of distinct TLDs, the average/standard deviation of domain name lengths, average/median/standard deviation of the frequency of N-grams, each TLD appearance, and so on are extracted from RHDN.

Various momentary URL-based features have also been proposed. Unlike the domain-based features, they are extracted from the URL. Prophiler [21] extracts the features such as presence of subdomain, presence of an IP address, TTL for the DNS A/NS record, country code, and so on.

Because these two types of features are designed from different viewpoints, they play different roles. Therefore, we use both types of features without including duplicate features. Note that our methods can use any features, though this thesis uses the features proposed by the existing work.

In our method, we vectorize the URL sequence by extracting the features for each URL included in the URL sequence.

3.4 URL Sequence Classifier

In our method, the URL sequence is classified by the neural network. In this subsection, we introduce two types of the neural networks used in our method.

3.4.1 Convolutional Neural Network

Fig. 2 shows the overall structure of the CNN. Convolution and pooling are repeated two times. Most of the redirection chains include 3 or 4 malicious URLs. Therefore, features of malicious

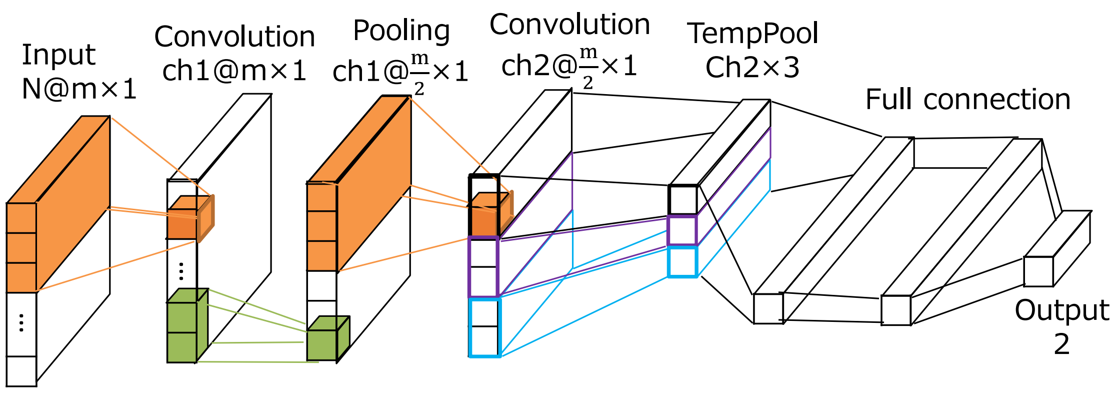

Figure 2: Architecture of the CNN; $ch1$ and $ch2$ represent number of channels in convolution layers.

URLs can be integrated with two convolution layers. After that, two fully-connected layers integrate the extracted information. The CNN has six kinds of layers, and has eight layers in all.

Input layer The input layer arranges our features extracted from URLs to input them into our CNN. Similar to the RGB values in pixels of images, our features have no order relation to each other. Similar to pixels in images, URLs are aligned in temporal order. Therefore, each URL is expressed as three-dimensional data of 1 row, 1 column, and N channels, where N is the number of the features of the URL. For convenience, three-dimensional data of i rows, j columns and k channels is written as $k@i \times j$. If the number of URLs is m, the input is represented as $N@m \times 1$. Each feature is normalized so that the average is 0 and the variance is 1 in the training data.

Convolution layer In each neuron of the convolution layer, the output is calculated from the receptive field of the previous layer. $x_{i,1,k}$ denotes the value of the i-th row, 1st column k-th channel of the previous layer. $w^c_{h,0,k}$ denotes the weight to calculate the c-th channel by multiplying the value of the neurons at the h-th row, 0th column and k-th channel from the receptive field. $y_{i,1,c}$, which is the output value of i-th row, 1st column, and c-th channel, is calculated by the following equation:

$$y_{i,1,c} = \tanh\left(\sum_{h=-H}^{H} \sum_{k=1}^{ch_{pre}} w^c_{h,0,k} x_{i+h,1,k}\right), \tag{1}$$

where H is the distance from the center of the receptive field to its boundary in the row direction, and ch_{pre} is the number of channels in the previous layer. If $x_{i+h,1,k}$ is outside the region of the previous layer, it is treated as 0. Here, letting ch_{conv} be the number of output channels of the convolution layer, if the input is $ch_{pre}@m \times 1$, the output will be $ch_{conv}@m \times 1$. This layer extracts the local relationships.

Max pooling layer In this layer, the maximum value of receptive field in the previous layer is extracted. The output $y_{i,1,k}$ is calculated as follows:

$$y_{i,1,k} = \max_{-H \leq h \leq H}(x_{i+h,1,k}). \tag{2}$$

where H is the distance from the center of the receptive field to its boundary. If $x_{i+h,1,k}$ is outside the region of the previous layer, it is treated as 0. If the input is $ch@m \times 1$, the output will be $ch@\frac{m}{2} \times 1$. As a result, even if the input position is slightly shifted, the output does not change.

Temporal pooling layer Our method accepts inputs of any length. However, the input size to the fully-connected layer must be fixed. Therefore, we introduce the temporal pooling layer so as to make the vectors of the fixed length from the vector of any length.

In the temporal pooling layer, we divide the input from the previous layer into s regions. Then the maximum value for each region is extracted. Finally, we construct the vector of the extracted maximum values.

The spatial pyramid pooling (SPP) layer [22] is the similar method to the temporal pooling layer. The SPP layer is the layer for processing the images with different sizes. The SPP divide the inputs into multiple square areas. Unlike the images, a URL sequence has features of only one direction. Thus, we use the temporal pooling layer instead of the SPP.

Fully-connected layer In this layer, the output is calculated from all values of the previous layer. Neurons in this layers are connected to all neurons in the previous layer. x_i denotes the value of the i-th neuron in the previous layer, y_j denotes the value of the j-th neuron in this layer, and w_{ij}denotes the weight between them. y_j is calculated by

$$y_j = \tanh\left(\sum_i w_{ij}x_i\right). \tag{3}$$

When learning the weight, we apply Dropout [23] that eliminates randomly selected neurons. This layer integrates all the information propagated from the previous layers.

Output layer This layer outputs the classification result. The number of neurons in the output layer is equal to the number of classes to be identified. The network structure in this layer is a complete bipartite graph. Similar to the fully-connected layer, the output value of this layer is calculated by the Softmax function; that is,

$$y_j = \frac{e^{z_j}}{\sum_i e^{z_i}}. \tag{4}$$

where z_i is defined by $z_j = \sum_i w_{ij} x_i$.

By using the Softmax function, the output values are positive and the sum of them becomes 1. Therefore, the output can be treat as the probability that the input belongs to the class.

In this thesis, the weight of each layer of the CNN is learned by the backpropagation to minimize the classification errors. We use AdaDelta [24] in this optimization, which automatically adjusts the learning rate.

3.4.2 Event De-noising Convolutional Neural Network

The URL sequences related to the drive-by download attacks also include the benign URLs. Such benign URLs may have an impact on the malicious URL sequence detection. Therefore, we introduce the Event De-noising CNN (EDCNN), which is an extended CNN to reduce the impact of benign URLs mixed in malicious URL sequences. An EDCNN performs convolution from two URLs, whose sequential order is close so that malicious URLs related to drive-by download attack can be convolved. In order to realize this convolution, we introduce an *allocation layer*.

The overall structure of EDCNN is shown in Fig. 3. The EDCNN has ten layers. There are seven kinds of layers. The rest of this subsection explains the *allocation layer*, which is newly introduced in the EDCNN, the convolution layer and the max pooling layer that are changed from the CNN.

Allocation layer Malicious URLs are highly likely to be close in the URL sequence, even if a few benign URLs are inserted. Therefore, as shown in Fig. 4, for the i-th URL in the sequence, i-th URL and $(i+1)$-th URL, i-th URL and $(i+2)$-th URL ..., and i-th URL and $(i+R)$-th URL are allocated so that they are adjacent.

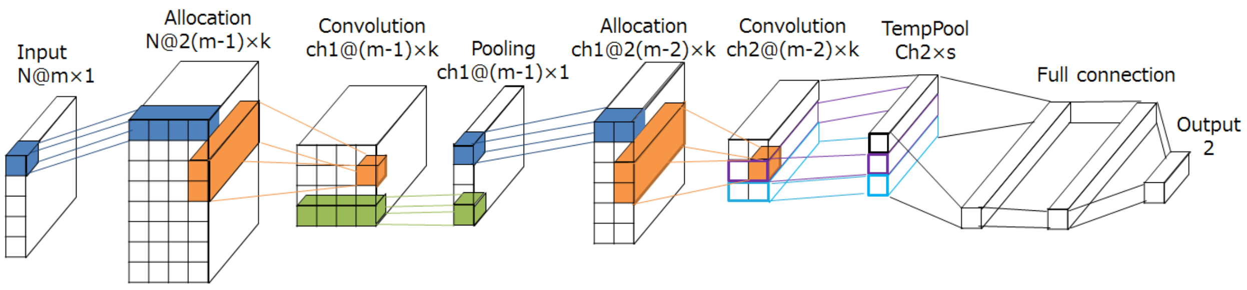

Figure 3: Architecture of the EDCNN; $ch1$ and $ch2$ represent number of channels in convolution layers.

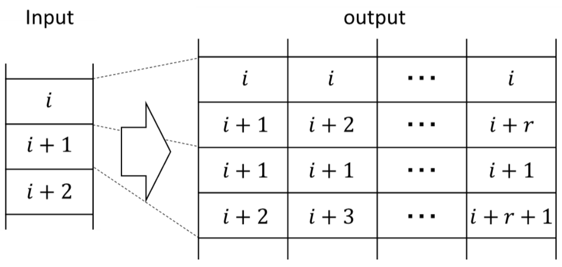
Input
output
i
$i+1$
$i+2$
i | i | ⋯ | i
$i+1$ | $i+2$ | ⋯ | $i+r$
$i+1$ | $i+1$ | ⋯ | $i+1$
$i+2$ | $i+3$ | ⋯ | $i+r+1$

Figure 4: *Allocation layer.*

The output y in the *allocation layer* is calculated by

$$y_{2i,j,k} = \begin{cases} x_{i,1,k} & (i+j \leq m) \\ 0 & (otherwise) \end{cases} \tag{5}$$

$$y_{2i+1,j,k} = \begin{cases} x_{i+j,1,k} & (i+j \leq m) \\ 0 & (otherwise), \end{cases} \tag{6}$$

where $x_{i,1,k}$ denotes the value of the i-th row, 1st column and k-th channel in the previous layer, $y_{i,j,k}$ denotes the value of the i-th row, j-th column and k-th channel in this layer, and m is the number of rows in the previous layer. If the input is $N@m \times 1$, the output will be $N@2(m-1) \times R$.

Convolution layer In the *allocation layer*, we arrange the neurons to convolute the data of the $2i$-th row, j-th column and data of the $(2i+1)$-th row, j-th column. In the convolution layer, we convolute these two neurons. $w^c_{h,0,k}$ denotes the weight to calculate the c-th channel in the convolution layer by multiplying the value of the neuron at the h-th row, 0-th column, and k-th channel from the center of the receptive field. $x_{i,j,k}$ denotes the value of the i-th row, j-th column and k-th channel in the previous layer. The output of the i row, j column, c channel ,$y_{i,j,c}$ is calculated by

$$y_{i,j,c} = \tanh\left(\sum_{h=0}^{1}\sum_{k=1}^{N} w^{c}_{h,0,k} x_{2i+h,j,k}\right). \tag{7}$$

The receptive field is moved by two neurons in the row direction. If the input is $N@2(m-1) \times R$, the output will be $N@(m-1) \times R$.

Max pooling layer Each column in the output of the convolution layer may include the convolution of the candidates of the malicious redirection. In the max pooling layer, we extract the convolution that is possible to correspond to the actual malicious redirection.

In the max pooling layer, the output $y_{i,1,k}$ is

$$y_{i,1,k} = \max_{0 \leq j \leq R}(x_{i+h,j,k}). \tag{8}$$

When x is outside the previous layer, it is treated as 0. If the input is $N@(m-1) \times R$, the output will be $N@(m-1) \times 1$.

Weight learning is performed by backpropagation similar to CNN. In the backpropagation, errors in the *allocation layer* are propagated through the connected neurons to the previous layer.

3.4.3 Malicious URL Sequence Detection

URL sequences extracted by the method in Section 3.2 may include URLs related to multiple web sites, because a user may access multiple web sites continuously. If one of such web sites is related to the drive-by download attack, most of the URLs included in the extracted sequences are benign. In this case, it is difficult to identify the malicious redirection chain from the extracted URL sequence.

Even if a user accesses multiple web sites, the accesses of the web sites are not completely simultaneous. The intervals of the redirect URLs are shorter than the interval of the accesses of the web sites. Therefore, the URLs of the same web sites tend to be close in the URL sequences.

In this method, we generate the subsequences of the URL sequence, and classify each subsequence by using CNNs. If one of the subsequences is classified as malicious, we regard the URL sequence as malicious. If all subsequences are classified as benign, we use the whole sequence as the input of the neural network.

By doing this, even if a user accesses multiple web sites continuously, we can detect the redirection chain from the subsequence of the URLs which includes multiple URLs related to the malicious redirection. If the URL sequence includes the URLs related to a single web site, we perform the classification by using the whole URL sequence.

In this thesis, we set the length of the subsequences based on the length of the URL sequence. Specifically, first the subsequence of l URLs from the beginning of the URL sequence is extracted and input into the model. Then the subsequence of the same length l shifting downward by h URLs is extracted and input into the model. Subsequently, inputting the subsequence at the same interval is continued until the end of the URL sequence. If one of the subsequences is classified as malicious, the URL sequence is classified as malicious. If all subsequences are not classified as malicious, the whole URL sequence is input last, and the output is used as the classification result.

4 Evaluation

In this section, we evaluate our method. In this evaluation, we check whether the malicious URL sequences can be detected before the honeyclient detects the sequence. We use URL sequences collected before a certain day as training data. Then, we evaluate the accuracy of identification of the URL sequences collected after the day.

4.1 Environment

4.1.1 Dataset

For this evaluation, we prepared two data sets. The first data set is collected by a honeyclient that accesses web sites listed in public blacklists [25, 26] and alexa [27].

Another data set is collected at the gateway of our laboratory. Our laboratory has about 30 users. The URL sequences collected at the gateway of our laboratory do not include the access to the malicious web sites. Therefore, we regard the URL sequences collected at our laboratory as benign. In this evaluation, URL sequences are terminated if 5 seconds are elapsed from the last accessed URL or 15 seconds are elapsed from the first URL.

We construct two sets of the training data and test data from the above data. First, in the former case, the training data is constructed based on the data collected before October 25, 2016, and the test data is constructed based on the data collected from October 26 to November 25. Second, in the later case, the training data is constructed based on the data collected before November 25, 2016, and the test data is constructed based on the data collected from November 26 to December 25. The following paragraphs explain the details of the training data and the test data.

Training Data We need the malicious and benign URL sequences for the training data. We use the malicious URL sequences collected by the honeyclient as the training data for malicious URL sequences.

The training data for the benign URL sequences is generated by mixing the URL sequences collected by the honeyclient and URL sequences collected in our laboratory. The benign URL sequence collected by the honeyclient includes the popular web sites, while the URL sequence collected in our laboratory reflects the behaviors of the people in our laboratory. That is, these data sets have different characteristics. Therefore, by mixing these data sets, classifier learns both

Table 1: Number of URL sequences in the training data for each patterns.

Label / Data source		Benign Laboratory	Benign Honeyclient	Malicious
The former case	lab	3195	0	3310
	labhc1	3195	2974	3310
	labhc2	1596	1487	3310
	labhc3	1065	992	3310
The later case	lab	3219	0	3321
	labhc1	3219	2903	3321
	labhc2	1610	1452	3321
	labhc3	1073	968	3321

of the characteristics of the benign URL sequences.

The number of collected benign URL sequences is much larger than that of malicious URL sequences. Therefore, we used the sampled benign URL sequences as the training data set. In each case of our evaluation, we generate four patterns of the training data of the benign URL sequences shown in Table 1; one is sampled from the data collected at our laboratory, and the others are sampled from the data collected at our laboratory and the data collected by the honeyclient.

Test Data We generate two kinds of the test data from the collected URL sequences. The first data is the data of URL sequences without multiple web sites. This data set includes the collected URL sequences. Some URL sequences collected for the test data may be included in the training data set. We eliminate such URL sequences from the test data.

The other data is the data including URL sequences with multiple web sites. The benign URL sequences included in this data set are the same as the previous data sets. The malicious URL sequences included in this data set are generated, considering the case that multiple web sites are accessed by a user and one of them is related to the drive-by download attacks. Such malicious URL sequences are generated by replacing the URLs related to a web site in the URL sequences collected in our laboratory with the malicious URLs collected by the honeyclient. For each malicious URL sequence, two benign URL sequences including multiple web sites are chosen randomly. In one benign sequence, the URLs of the web site accessed first are replaced with the

Table 2: Number of URL sequences in the test data.

Label Number of web sites	Benign	Malicious Single	Malicious Multiple
The former case	14,285	949	1898
The later case	15,905	461	922

URLs of the malicious URL sequence. In the other benign sequence, the URLs of a web site accessed later are replaced. Therefore, the number of the malicious URL sequences including multiple web sites is doubled from that of the collected malicious URL sequences. Table 2 shows the number of URL sequences in each test data.

4.1.2 URL Features

As described in Section 3.3, in this thesis, we use historic domain-based and momentary URL-based features. We get RHDN and RHIP of the domain of each input URL, and the features shown in the left and center of Table 3 are extracted. These features are the same as the domain-based features used in Notos introduced in Section 3.3. Then, features shown on the right of Table 3 are extracted for the input URL. Features ”Length of X” in No. 36-38 by defining the filename, path and query included in the URL. In this thesis, they for the URL ”http://host:port/folder/file?query” is

- Filename: file
- Path: folder/file
- Query: query.

In table No. 39-41, ”Presence of X” indicates whether there is a corresponding item in the URL by 0 (none) or 1 (present).

4.2 Classification Method

4.2.1 Sequence-based Approach

The CNN or EDCNN has the following parameters; the number of channels in convolution layers, the number of neurons in fully-connected layers, the number of rows in receptive field for convolu-

Table 3: Features used in this evaluation.

No.	RHIP Features	No.	RHDN Features	No.	URL Features
1	# BGP Prefixes (FQDN)	19	# FQDNs	36	Length of filename
2	# BGP Prefixes (3LD)	20	Mean (Lengths)	37	Length of path
3	# BGP Prefixes (2LD)	21	SD (Lengths)	38	Length of query
4	# Countries (FQDN)	22	Mean (1-gram)	39	Presence of IP addresses
5	# Countries (3LD)	23	Median (1-gram)	40	Presence of a subdomain
6	# Countries (2LD)	24	SD (1-gram)	41	Presence of a port number
7	# IP addresses (3LD)	25	Mean (2-grams)		
8	# IP addresses (2LD)	26	Median (2-grams)		
9	# Organizations (FQDN)	27	SD (2-grams)		
10	# ASN (FQDN)	28	Mean (3-grams)		
11	# ASN (3LD)	29	Median (3-grams)		
12	# ASN (2LD)	30	SD (3-grams)		
13	# Registries (FQDN)	31	# TLDs		
14	# Registries (3LD)	32	Ratio of .com		
15	# Registries (2LD)	33	Mean (TLD)		
16	# Dates (FQDN)	34	Median (TLD)		
17	# Dates (3LD)	35	SD (TLD)		
18	# Dates (2LD)				

tion. In this evaluation, we obtain the hyperparameters which achieves high accuracy for training data. As a result, the number of the convolution layers is set to 200 and the number of the neurons in fully-connected layers is set to 500. The number of rows in receptive field for convolution is set to 5 for the CNN, 4 for the first convolution layer of the EDCNN, 3 for the second convolution layer of the EDCNN.

As described in Section 3.4.3, we generate the subsequences of the URL sequences, and classify them by the neural network. In this evaluation, we set the length and the interval of the subsequence of the URL sequence including N URLs by

$$l = \lfloor \sqrt{N} \rfloor, h = \lfloor \sqrt{l} \rfloor.$$

4.2.2 Individual-based Approach

In this thesis, we compare our method with the method that classifies individual URLs. By this comparison, we demonstrate the effectiveness of using URL sequences. In this method, if one of the URLs in the sequence is classified as malicious, the sequence is classified as malicious.

We use Random Forest (RF) as the classifier of each URL. We train the RF by using the URLs captured in our laboratory as the benign data and the URLs that are identified as exploit URL by the honeyclient as the malicious data. The samples in the training data set of the later case includes only a small number of URLs that are identified as exploit URL. Therefore, in this section, we compare our method with RF by using only the data set of the former case in Table 1 and 2.

4.3 Classification Performance

In this subsection, we first investigate the classification performance of the trained neural networks. In this evaluation, we use the data set of URL sequences without multiple web sites. We classify the whole URL sequences without generating the subsequences. Table 4 shows true positive rate (TPR) and false positive rate (FPR).

Table 4: Classification performance for the data without multiple web sites including malicious URLs.

Model		The former case		The later case	
		TPR	FPR	TPR	FPR
CNN	lab	0.967	0.036	0.937	0.070
	labhc1	0.952	0.068	0.792	0.008
	labhc2	0.948	0.271	0.804	0.010
	labhc3	0.961	0.070	0.879	0.019
EDCNN	lab	0.952	0.041	0.954	0.129
	labhc1	0.959	0.079	0.894	0.024
	labhc2	0.959	0.152	0.915	0.021
	labhc3	0.975	0.085	0.879	0.024
RandomForest		0.946	0.003	-	-

The result shows the TPR of RF is the lowest. This is because the RF is based on the characteristics of the exploit URLs detected by the honeyclient. Thus, the RF cannot detect the malicious URL sequences that use the exploit URL whose characteristics are different from the exploit URLs detected by the honeyclient.

On the other hand, the CNN and EDCNN achieve higher TPR. This is because these methods uses the features of the URL sequences instead of the features of the individual URLs. Therefore, even if the attacker uses a new exploit URL whose characteristics are different from the past exploit URLs, the CNN or EDCNN can detect the malicious URL sequences from the features of the redirections, which hardly change.

In this result, there are no clear differences between the CNN and the EDCNN, even though the CNN does not consider the benign URLs included in the malicious URL sequences. This is because we have sufficient training data to train a CNN so that the benign URLs included in the malicious URL sequences do not affect the classification by the CNN.

We next investigate the accuracy of the classification when multiple web sites are included in a URL sequences. To handle such URL sequences, we generate the subsequences. Table 5 shows the result. For comparison, we also show the result when classifying the malicious data without subsequences. This result shows that the CNN and EDCNN without using subsequences cannot detect a large number of malicious URL sequences, if the URL sequences include URLs of multiple web sites. This is caused by that most of the URLs included in the URL sequences are benign URLs when URLs of the multiple web sites are included in the URL sequences and only one of the web sites is related to the drive-by download attacks. As a result, the URL sequences tend to be classified as the benign URL sequences.

On the other hand, the CNN and EDCNN using subsequences achieve high TPR. This is because the URLs of the same web sites tend to be close in the URL sequences. Therefore, one of the subsequences includes the redirections related to the drive-by download attacks, and such redirections are detected by the CNN and EDCNN. However, the URLs of the same web sites tend to be close in the URL sequences cause the high FPRs. This is because the CNN and EDCNN using subsequences identify the URL sequences as malicious if at least one of the subsequences is mistakenly classified as malicious. Therefore, we need a method to decrease the false positives.

The neural networks, which learn the different data sets, may learn the different characteristics of the malicious and benign URL sequences. Therefore, the classifier integrating multiple neural

Table 5: Classification performance for the URL sequences including multiple web sites.

Model		The former case		The later case	
		TPR	FPR	TPR	FPR
CNN using subsequences	lab	0.886	0.113	0.898	0.120
	labhc1	0.942	0.193	0.785	0.054
	labhc2	0.983	0.365	0.823	0.067
	labhc3	0.956	0.220	0.876	0.086
EDCNN using subsequences	lab	0.910	0.144	0.936	0.155
	labhc1	0.980	0.257	0.907	0.104
	labhc2	0.989	0.391	0.915	0.069
	labhc3	0.973	0.270	0.881	0.088
CNN without subsequences	lab	0.467	0.036	0.663	0.070
	labhc1	0.482	0.068	0.320	0.008
	labhc2	0.725	0.271	0.362	0.010
	labhc3	0.609	0.070	0.541	0.019
EDCNN without subsequences	lab	0.419	0.041	0.838	0.129
	labhc1	0.537	0.079	0.501	0.024
	labhc2	0.560	0.152	0.915	0.021
	labhc3	0.509	0.085	0.504	0.003

networks may achieve higher accuracy of the classification. We integrate the four EDCNNs by the following rule. Each classifier classifies a URL sequence by using the subsequences of it. If n or more classifiers classify it as malicious, it is classified as malicious. Otherwise, it is classified as benign. Table 6 shows the results.

Table 6: Classification performance for the URL sequences including multiple web sites when integrating the outputs of the four EDCNNs using subsequences.

	The former case		The later case	
n	TPR	FPR	TPR	FPR
1	0.997	0.492	0.977	0.227
2	0.988	0.278	0.946	0.123
3	0.973	0.196	0.900	0.072
4	0.895	0.095	0.816	0.039

If $n = 1$, TPR and FPR become high. This is because a URL sequence is classified as malicious if any one of the four models classifies it as malicious. On the other hand, if $n = 4$, TPR and FPR become low, because malicious URL sequences are not detected unless all models classify them as malicious. If n is 2 or 3, the method integrating outputs of the EDCNNs achieves high TPR and low FPR. This is because four models learn the different characteristics of the benign and malicious URL sequences. If a model cannot detect a malicious URL sequence, the other models may detect the URL sequence. As a result, the method integrating the models can detect the sequence. Similarly, even if a model mistakenly identifies a benign URL sequence as malicious, the method integrating the models does not identify the URL sequence as malicious unless one of the other models also identifies it as malicious. As a result, integrating the outputs of the models increases the TPR and decreases the FPR.

4.4 Discussion

Our evaluation results demonstrate that the EDCNN using the subsequences achieves the TPR higher than 95% even when URL sequence includes multiple web sites and one of them is related to the drive-by download attacks. In addition, by integrating the outputs of the EDCNNs which learned with the different data sets, we can increase the TPR and decrease the FPR.

The EDCNN using the subsequences may cause some false positives, even if we integrate the outputs of multiple EDCNNs. This is because the EDCNN using the subsequences identifies a URL sequence as malicious when one of the subsequences are identified as malicious. As a result, long URL sequences tend to be identified as malicious.

Table 7 shows the relation between the length of URL sequences and the FPR of the corresponding URL sequences. In this table, FPRs are the averages for the results of the four EDCNNs.

Table 7: FPR according to the length of the URL sequence.

Range of length	-5	6-10	11-20	21-40	41-80	81-160	161-320	321-
Number of samples	4995	2743	1797	1730	1405	869	456	290
False positive rate	0.141	0.196	0.270	0.342	0.401	0.541	0.640	0.520

Table 7 indicates the FPR increases as the length of the sequence becomes long. By investigating URL sequences in our data set, we find that the number of URLs actually related to attack is small, compared with the length of the malicious URL sequences. That is, long URL sequence may be useless for classification. Therefore, FPR can be decreased by modifying the parameter of the URL sequence extraction so that the length of the extracted URL sequences becomes short.

5 Conclusion and Future Work

In the thesis, we proposed a method to detect drive-by download attacks from URL sequences extracted from proxy log information. We focused on the similarity in the data structure of URL sequences and sentences, and applied convolutional neural networks to classification of URL sequences. We also introduce an Event De-noising CNN, which is an extended CNN so as to perform convolution from two adjacent URLs in the sequence.

The evaluation demonstrated that the methods using URL sequences achieves higher TPR than the method using the features of individual URLs.

URL sequences extracted from the proxy log may include URLs of multiple web sites accessed at the same time. To handle such URL sequences, we propose a method that uses the subsequences. The evaluation results show that EDCNN using the subsequences achieves the high TPR even when URL sequences includes URLs of multiple web sites. However, the EDCNN using subsequences cause high FPR. To improve accuracy of the classification, we integrated multiple EDCNNs that learns different data sets. The evaluation results show that integrating multiple EDCNNs achieve high TPR and low FPR.

However, EDCNNs using subsequences cause some false positives, even if we integrate multiple EDCNNs. In this thesis, we investigated the false positives, and clarified that the long benign URL sequence is likely to be mistakenly identified as malicious. Therefore, we need to extract URL sequences so that the lengths of the URL sequences do not become too large.

In our evaluation, we use the URL sequences monitored for 6 months as the training data. However, there may be tradeoffs. By focusing on recent URL sequences, we can learn the features of the recent attacks. However, the number of malicious URL sequences that are recently monitored is limited. On the other hand, by using old URL sequences, we can use more data to train the models. However, the current characteristics of the attacks may be different from the old URL sequences. In this case, the old URL sequences may be useless. Considering above points, we plan to investigate how to collect suitable training data.

Acknowledgements

First, I would like to express my sincere gratitude to my supervisor, Professor Masayuki Murata of Osaka University, for his continuous support and expensive advice throughout my studies, and providing me the rich study environment in his laboratory and the opportunity for the collaborative research with NTT Secure Platform Laboratories.

I especially would like to express my deepest appreciation to Assistant Professor Yuichi Ohsita of Osaka University, for his elaborated guidance and invaluable firsthand advice. All works of this thesis would not be possible without his support.

My deepest thanks are also to Mr. Toshiki Shibahara of NTT Secure Platform Laboratories who gave me precise suggestions and helpful support whenever conducting the evaluation experiments. And I would also like to offer my special thanks to Mr. Yuta Takata, Mr. Daiki Chiba, Dr. Mitsuaki Akiyama and Dr. Takeshi Yagi of NTT Secure Platform Laboratories, for providing me abundant data of actual drive-by download attacks and for beneficial discussions. This study would not be achieved without their support.

Furthermore, I would like to appreciate to Dr. Makoto Imase of NICT, Assistant Professor Naomi Kuze of Osaka University, for helpful discussions and accurate comments on this study. I would also like to thank Assistant Professor Daichi Kominami, Associate Professor Shin'ichi Arakawa of Osaka University for their valuable comments and suggestions.

Finally, I truly thank all the members of Advanced Network Architecture Research Laboratory of Osaka University, for their support, cooperation on traffic capturing and encouragement.

References

[1] J. Zhang *et al.*, “Arrow: Generating signatures to detect drive-by downloads,” in *Proceedings of the 20th international conference, on World wide web*, pp. 187–196, 2011.

[2] M. Antonakakis *et al.*, “Building a dynamic reputation system for DNS,” in *Proceedings of the 19th USENIX Security Symposium*, pp. 273–290, 2010.

[3] M. Akiyama *et al.*, “Marionette: Client honeypot for investigating and understanding web-based malware infection on implicated websites,” in *Proceedings of Joint Workshop on Information Security*, 2009.

[4] B. Hu *et al.*, “Convolutional neural network architectures for matching natural language sentences,” in *Proceedings of Advances in Neural Information Processing Systems*, pp. 2042–2050, 2014.

[5] K. Yamanishi *et al.*, “Drive-by download attacks detection based on URL sequence using convolutional neural networks,” *in Proceedings of the 2016 Computer Security Symposium*, vol. 2016, no. 2, pp. 811–818, 2016 (in Japanese).

[6] C. Kolbitsch *et al.*, “Rozzle: De-cloaking internet malware,” in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 443–457, 2012.

[7] C. Curtsinger *et al.*, “Zozzle: Fast and precise in-browser javascript malware detection,” in *Proceedings of the 20th USENIX Security Symposium*, pp. 33–48, 2011.

[8] T. Nelms *et al.*, “Execscent: Mining for new c&c domains in live networks with adaptive control protocol templates,” in *Proceedings of the 22nd USENIX Security Symposium*, pp. 589–604, 2013.

[9] H. Project, “Capture-hpc.”

[10] J. Nazario, “Phoneyc: A virtual client honeypot.,” *LEET*, vol. 9, pp. 911–919, 2009.

[11] M. Cova *et al.*, “Detection and analysis of drive-by-download attacks and malicious javascript code,” in *Proceedings of the 19th International conference, on World Wide Web*, pp. 281–290, 2010.

[12] J. Ma *et al.*, "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*, pp. 1245–1254, 2009.

[13] H. Mekky *et al.*, "Detecting malicious http redirections using trees of user browsing activity," in *Proceedings of IEEE INFOCOM*, pp. 1159–1167, 2014.

[14] T. Taylor *et al.*, "Cache, trigger, impersonate: Enabling context-sensitive honeyclient analysis on-the-wire," in *Proceedings of the 23rd Annu. Network and Distributed System Security Symposium*, 2016.

[15] T. Nelms *et al.*, "Webwitness: investigating, categorizing, and mitigating malware download paths," in *Proceedings of the 24th USENIX Security Symposium*, pp. 1025–1040, 2015.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira *et al.*, eds.), pp. 1097–1105, 2012.

[17] A. Karpathy *et al.*, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference, on Computer Vision and Pattern Recognition*, 2014.

[18] K. Borgolte *et al.*, "Meerkat: Detecting website defacements through image-based object recognition," in *Proceedings of the 24th USENIX Security Symposium*, pp. 595–610, 2015.

[19] E. C. R. Shin *et al.*, "Recognizing functions in binaries with neural networks," in *Proceedings of the 24th USENIX Security Symposium*, pp. 611–626, 2015.

[20] M. Antonakakis *et al.*, "Detecting malware domains at the upper dns hierarchy.," in *Proceedings of the 20th USENIX Security Symposium*, pp. 1–16, 2011.

[21] D. Canali *et al.*, "Prophiler: a fast filter for the large-scale detection of malicious web pages," in *Proceedings of the 20th international conference, on World wide web*, pp. 197–206, 2011.

[22] K. He *et al.*, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proceedings of the 13th European conference, on Computer Vision*, pp. 346–361, 2014.

[23] G. E. Hinton *et al.*, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[24] M. D. Zeiler, “Adadelta: an adaptive learning rate method,” *arXiv preprint arXiv:1212.5701*, 2012.

[25] MalwareBytes, “hphosts.”

[26] MDL, “Malware domain list.”

[27] Alexa.com, “Alexa top global sites.”