Malicious URL detection with feature extraction based on machine learning

Baojiang Cui* and Shanshan He

Nation Engineering Laboratory for Mobile Network Security,
Beijing University of Post and Telecommunications,
Beijing, China
Email: cuibj@bupt.edu.cn
Email: shanshan_2512@sina.com
*Corresponding author

Xi Yao

QIHU 360 Software Co. Limited,
Building #2, No. 6 Jiuxianqiao Road,
Chaoyang District, Beijing, China
Email: yaoxi@360.cn

Peilin Shi

Nation Engineering Laboratory for Mobile Network Security,
Beijing University of Post and Telecommunications,
Beijing, China
Email: noahstone@163.com

Abstract: Many web applications suffer from various web attacks due to the lack of awareness concerning security. Therefore, it is necessary to improve the reliability of web applications by accurately detecting malicious URLs. In previous studies, keyword matching has always been used to detect malicious URLs, but this method is not adaptive. In this paper, statistical analyses based on gradient learning and feature extraction using a sigmoidal threshold level are combined to propose a new detection approach based on machine learning techniques. Moreover, the naïve Bayes, decision tree and SVM classifiers are used to validate the accuracy and efficiency of this method. Finally, the experimental results demonstrate that this method has a good detection performance, with an accuracy rate above 98.7%. In practical use, this system has been deployed online and is being used in large-scale detection, analysing approximately 2 TB of data every day.

Keywords: malicious URLs; feature selection; machine learning; multiple algorithms.

Reference to this paper should be made as follows: Cui, B., He, S., Yao, X. and Shi, P. (2018) 'Malicious URL detection with feature extraction based on machine learning', *Int. J. High Performance Computing and Networking*, Vol. 12, No. 2, pp.166–178.

Biographical notes: Baojiang Cui received his BS in the Hebei University of Technology, China, in 1994, MS in the Harbin Institute of Technology, China, in 1998 and PhD in Control Theory and Control Engineering in the Naikai University, China in 2014. He is a Professor in the School of Computer Science at Beijing University of Posts and Telecommunications, China. His main research areas are detection of software, cloud computing and the internet of things.

Shanshan He is a member of Nation Engineering Laboratory for Mobile Network Security. She is also a graduate student in Computer Science at the Beijing University of Post and Telecommunications. Her research focuses on machine learning and network security.

Xi Yao is a Researcher at the Cloud Security Department of 360 Company which is a well-known internet company in China that specialises in network security. He is engaged in the study of safe application of large-scale network attack and defence etc. Their department has launched cloud security platform software for large enterprises and the company provides network security. He and his colleague have made outstanding contributions for China's network security.

Peilin Shi received his BS in Information and Computing Science from the Shenyang Ligong University, China in 2014, and currently is studying for his Master's degree in Computer Technology at the Beijing University of Posts and Telecommunications in China. His main research areas involve cloud computing and big data.

This paper is a revised and expanded version of a paper entitled 'Malicious URL detection with feature extraction based on machine learning' presented at GPC 2016, Xi'an, China, 6–8 May 2016.

1 Introduction

Currently, the web is a widely used platform that supports an increasing number of daily activities. However, the security of web applications is, unfortunately, becoming a serious problem as web-based services become increasingly prevalent (Johnson and Shanmugam, 2011). Such sites suffer from many types of web attacks including SQL injection, XSS attack, information leaks, and so on (Han et al., 2016). The rising number of security events and huge economic losses warn us that malicious web traffic detection has become a great challenge for both individuals and enterprises who need to protect their information security.

Many web applications can be accessed directly via web browsers over the network. All these access attempts require the user to manually input a URL in web browsers or take some action such as clicking a link (Hou et al., 2010). Users request web resources via URLs; meanwhile, URLs also provide an easy point of entry for attackers, who embed executable code into the website content or inject malicious code into the URL, creating a web attack (Lifeng et al., 2012; Yee et al., 2005). For example, hackers can manipulate data in web databases through SQL injection, inject malicious client-side script into web pages through XSS attacks, or obtain the contents of a file through a file inclusion attack (Ma et al., 2009b). Therefore, while URLs are a convenient way to address a web resource, they also impose some risks.

In previous studies, to protect the security of web applications, malicious keyword databases and matching regulations have been employed to identify malicious URLs (Bhuyan et al., 2014; Cui et al., 2015; Prakash et al., 2010). However, this approach is inflexible and cannot correctly detect malicious URLs that have complicated attack behaviours or fuzzy characteristics (Dogru and Subasi, 2012). In addition, attackers can easily avoid these normal types of detection and continue attacking (Sommer and Paxson, 2010; Blum et al., 2010; Zhang et al., 2008). Recent research improving on previous methods has focused on using machine learning techniques to identify complicated behaviours with new patterns of attack.

Therefore, this paper focuses on exploring the behaviours of malicious URLs and extracting their features from five different dimensions that can distinguish malicious URLs from benign URLs based on machine learning techniques. Moreover, two additional progressive algorithms are applied during the processing: statistical analysis on gradient learning and feature extraction at a sigmoidal threshold level. The gradient learning method attaches the most accurate weight to each risk level, while the automatic feature extraction algorithm based on a sigmoidal threshold level is applied to validate the effectiveness of each feature. Furthermore, a comparison of the results using evaluation indicators to test multiple machine learning algorithms (naïve Bayes, decision tree and SVM) using self-refit and ten-fold cross-validation was able to identify the decision tree classifier as the best algorithm for this purpose.

Compared with previous work, the experimental results show that this approach is both more accurate and more adaptive in detecting malevolent URLs than previous approaches. To perform the evaluation of this detection model, 3,309,915 distinct URLs were selected as a training set; subsequently, in the real world, the model was able to detect approximately 50,000 malicious URLs everyday. The testing results show that the proposed model achieves an overall prediction accuracy between 98% and 99.9%. Moreover, this system has been deployed online for well-known internet companies in China that specialise in network security, where it handles URL detection on a large-scale, analysing approximately 2 TB of data everyday.

The remainder of this paper is organised as follows: Section 2 summarises the related work about the latest detection methods, Section 3 describes the proposed approach for detecting malicious web traffic by analysing URLs, including our detection framework, statistical analysis algorithm and feature selection algorithm, and introduces the classification models. Section 4 analyses our features in detail. Section 5 presents the system implementation and the experimental results, and Section 6 concludes and presents directions for future work.

2 Related work

This section reviews related works on malicious traffic detection systems and methods. These approaches include matching from blacklisting databases and feature extraction methods based on machine learning.

2.1 Matching through blacklisting

Zhang focused on the blacklisting strategy and introduced a blacklisting system based on a relevance ranking scheme borrowed from the link-analysis community (Ageev et al., 2015). The ranking scheme measures how closely an attack source is related to a contributor, using that attacker's history and the contributor's recent log production patterns. Finally, this work produced individualised blacklists and tested the scheme on a corpus of over 700 million log entries.

Prakash proposed an approach that used an approximate matching algorithm and five heuristics for enumerating simple combinations of known phishing sites to discover new phishing URLs (Kim et al., 2004). To match URL components, they mainly matched IP addresses, hostnames, directory structures, and brand names. They also used Google's browser blacklist to help complete their blacklist.

However, in general, matching using blacklisting approaches suffers from poor generalisation ability, making it unsuitable for detecting new malicious URLs or unknown malicious patterns. Our work also applies blacklisting in this system, by dividing the URLs into different risk levels and then applying a weight and calculating the risk value for each URL. However, here, the assigned risk value is regarded as just one feature used as input to the machine learning process.

2.2 Feature extraction

In previous studies, a variety of machine learning approaches have been proposed for malicious URL detection.

In 2009, Shabtai created a taxonomy for classifying detection methods of malicious code by machine learning methods based on static features (Shabtai et al., 2009). The study addressed various facets of the detection challenge such as file representation, feature selection methods, classification algorithms, and weighting ensembles. The work concluded that it is possible to design a framework for detecting malicious code in executable files that would achieve high accuracy. However, the study was solely a survey of detection methods; it did not propose a method or conduct experiments to prove that conclusion.

Garera focused on studying the structure of URLs without any knowledge of the corresponding page data (Garera et al., 2007). They used logistic regression with several features to distinguish phishing URLs from benign ones. The features include the presence of 'red flag' keywords in the URL, which were based on Google's Page Rank and Google's web page quality guidelines. As a result, their classification technique achieved an accuracy of 97.3% in phishing detection.

To solve class-imbalanced learning problems, Zhao presented a novel framework for cost-sensitive online active learning (CSOAL), which queries only a small fraction of the training data for labelling and directly optimises two cost-sensitive measures to address the class-imbalance issue: the weighted sum of sensitivity and specificity and the weighted cost (Zhao and Hoi, 2013).

The work by Ma is the most closely related to our study with regard to feature extraction (Ma et al., 2009a). The paper used statistical methods to discover the telltale lexical and host-based properties of malicious website URLs. It extracted nine features for machine learning and compared the proposed classification model against logistic regression, naïve Bayes and the SVM model.

Although similar in motivation and methodology to the preceding papers, this paper proposes a new method to evaluate features automatically and extract more effective features to detect malicious URLs. In addition to the lexical features, this paper focuses on specific attack types and formats them into features for machine learning. These features are validated as having good performance in improving the detection rate of malevolent URLs.

3 Approach

This section provides a detailed discussion of our approach to detect malicious web traffic by analysing URL requests. A URL request can be treated as a binary classification problem in which positive instances are malicious or vulnerable URL requests and negative instances are benign (Zhiwang et al., 2010). This learning-based approach to the problem can succeed if the distribution of feature values for malicious instances are different from those of benign instances and the training dataset shares the same parsing process with the testing dataset (Kinder et al., 2005). In the process of building the detection model, three algorithms were tested on the training dataset and we selected the best one as the practical detection model (Zhang et al., 2008).

3.1 Detection with URL requests

After analysing URLs containing multiple types of attacks, we concluded that malevolent URLs with integrated attack behaviours all contain some special context structures. Table 1 lists some examples of different attack types. As a result, we regard all the attack types as malicious but do not classify them to a specific category.

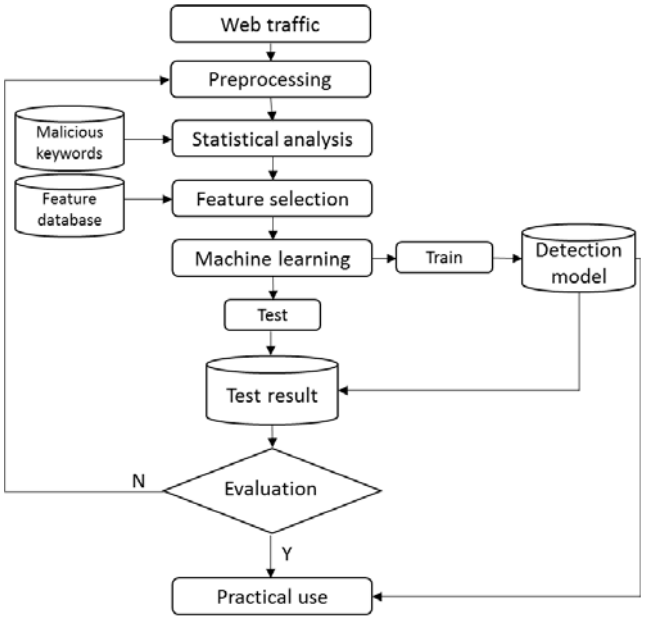
3.2 Detection framework

The proposed detection framework based on machine learning is shown in Figure 1. This detection framework can be divided into three main modules: the processing procedure, the machine learning procedure and the evaluation procedure. The processing procedure makes sure the data are in a useable format and transfers the values to vector features. The machine learning module is where the core algorithms reside; depending on the algorithm, it feeds the algorithm valid clean data from the processing module and extracts knowledge or information. The evaluation module is devoted to estimating whether the preceding processing can result in good performance and be put to use.

Table 1 Some types of URL attacks

No.	*Attack types*	*Example*
1	Code Injection	/search.php?searchsubmit=yes;print(md5(SEUPtIpN));
2	PHP Injection	/member.php?mod=data:text/plain,<?php phpinfo();?>
3	J2EE Attack	/WEB-INF/web.xml
4	LDAP Injection	/data/cache/ajax.js?gzd^(#$!@#$)(()))******
5	Filename Attack	/*~1.*/x.aspx
6	XSS ATTACK	/console/portal/"><script>alert(12345067809)</script>
7	Ddos attack	/data/cache/fuck.php?ip=24.153.96.151&port=3389&time=4000
8	Anwsion Attack	/api.php?op=ajax_domain&url=/etc/passwd
9	Information Leak	/.htaccess
10	Backup File Attack	/config.bak
11	Get SQL Inject	/member.php?mod=-1 OR 1=1&action=login
12	SSI Injection	/member.php?mod=<!--#exec cmd="expr 268409241-2"-->&action=login
13	Command Injection	/data/cache/forum.php?mod=ping-n 25 127.0.0.1 &&fid=15&filter=typeid&typeid=182
14	Remote Code Injection	/search.php?mod='{${print(int)0xFFF9999-22}}'&srchtxt=ÊÖ»ú&formhash=015aabe0&searchsubmit=true&source=hotsearch
15	ddos attack	/data/cache/fuck.php?ip=24.153.96.151&port=3389&time=4000

Figure 1 Detection framework



3.2.1 Processing procedures

The three processing procedures are listed below.

- *Preprocessing*. All the web traffic is filtered and parsed in a unified format, after which the URLs are transformed into a unified format for mining and detection.
- *Statistical analysis*. This procedure loads the malicious keywords and calculates the risk value of each URL. A higher risk value indicates a higher possibility of vulnerability. This is just one – but an important – feature of the feature database; this proposed module performs a statistical analysis based on a gradient learning algorithm to allocate an appropriate weight for each level and compute the risk value.
- *Feature extraction*. Feature extraction is at the core of the processing; it identifies the essential characteristics of malicious web traffic to distinguish malicious URLs from normal traffic. To do this, the features are fed into an algorithm that builds the detection model. For this purpose, an automatic feature extraction algorithm based on a sigmoidal threshold level is proposed to validate the effectiveness of each feature. Finally, 21 features won out as the features used in the detection model.

3.2.2 Machine learning

The machine learning module contains two independent processes. The training process uses the machine learning algorithm to build the detection model by labelling web traffic (URLs). The testing process loads the detection model to test unknown traffic and outputs the test result.

Different algorithms are adopted at analysing different data types. To obtain the best performance, multiple classification models are tested to obtain the most appropriate algorithm for this project. Self-refit and cross-validation are basic functions, and the precision rate, recall rate and ROC area are common indicators for comparing such algorithms. In general, we apply all these methods integrally in the machine learning module.

3.2.3 Evaluation procedure

Algorithms should be evaluated to see how well they perform before deploying the detection model for practical use. Machine learning procedures already have key statistical indicators; therefore, we can use those as evaluation measures to select the best algorithm at the same

time. If the precision and recall ratio are good, we can use the training model in the real program; otherwise, we go back to the previous step and test again. Often, data collection or preparation can be the cause of the problem. In that case we need to revisit the preprocessing step. Statistical analysis and feature extraction may also cause problems; therefore, it is sometimes necessary to rectify the malicious keywords or extract more features and then retest.

In conclusion, creating an accurate evaluation procedure is a difficult process; each step can influence the performance. At the same time, each step may also offer room for improvement. The best performance can be identified only through repeated evaluation and testing. Our system currently has good performance and is in practical use; it has been running online for several months.

3.3 Algorithm description

The core idea of detection is to feed valid clean data to the machine learning classification model to improve the detection precision. Many machine learning classification models have good performance and could be applied; therefore, determining how best to process the web data is the most significant procedure. The algorithm consists of two parts: statistical analysis and feature selection.

3.3.1 Statistical analysis based on gradient learning

One type of malicious URL is injected with an executable file or with source code. These can easily be recognised by matching values with known malicious keywords. Therefore, we built databases of malicious keywords and classified them into three risk levels based on their degree of risk. Table 4 through Table 7 are indicative of the databases of malicious keywords with different risk levels. Furthermore, a gradient learning algorithm is used to calculate the weight for each risk level. To solve for these three risk levels, the weights of the high and low risk levels are fixed initially, and the algorithm is applied only to calculate the middle risk level.

Notations:

Let *Weight* be the risk level of keywords. $Weight_{low}$ and $Weight_{high}$ are fixed initially, while $Weight_{middle}$ is the output of this algorithm.

Let *Fr* be the integrated risk value feature.

Let $Instance = \{f_1, f_2, \ldots, f_m, r_l\}$ be the instance of data that contains the features and the real label.

Let M_{Inst} be the number of the instance such that $DataSet = \{Instance_1, Instance_2, \ldots, Instance_w\}$.

Let *PNum* be the accuracy rate of the *DataSet*.

Algorithm 1 Gradient learning

input: *DataSet*, *Instance*, $Weight_{low}$, $Weight_{high}$, M_{Inst},
output: $Weight_{middle}$

function EVALUATION_MID(*DataSet*, *Instance*, $Weight_{low}$, $Weight_{high}$)
  $PNum_{max} \leftarrow 0$
  for $Weight_{temp} \leftarrow Weight_{low}$ to $Weight_{high}$ **do**
    $F_r \leftarrow$ JUDGERISK(*DataSet*)
    $Instance \leftarrow F_r$
    $PNum_i \leftarrow$ (RIGHTNUM (*DataSet*, *Instance*)) / M_{Inst}
    if $PNum_i > PNum_{\max}$ **then**
      $PNum_{\max} \leftarrow PNum_i$
      $Weight_{middle} \leftarrow Weight_{temp}$
    end if
  end for
  return $Weight_{middle}$
end function

input: *DataSet*, *Instance*
output: Num_{right}

function RIGHTNUM(*DataSet*, *Instance*)
  model ← LoadDetectionModel(*modelpath*)
  for i ← 1 to M_{Inst} **do**
    $testlabel_i \leftarrow$ model.classify($Instance_i$)
    $reallabel_i \leftarrow Instance_i.get(\text{Length}(Instance_i) - 1)$
    if $testlabel_i == reallabel_i$ **then**
      $Num_{right} \leftarrow Num_{right} + 1$
    end if
  end for
  return Num_{right}
end function

The function JUDGERISK computes the integrated risk value for each URL after obtaining the weight for each risk level.

Notations:

Let data be the testing string. The purpose of the algorithm is to compute the risk value of the input data.

Let $HighSet = \{h_1, h_2, \ldots, h_m\}$ be the set of malicious keywords with high-level weight, where h_s is the keyword and *m* is the number of keywords.

Let $MiddleSet = \{m_1, m_2, \ldots, m_n\}$ be the set of malicious keywords with middle-level weight, where ms is the keyword and where n is the number of keywords.

Let $LowSet = \{l_1, l_2, \ldots, l_p\}$ be the set of malicious keywords with low-level weight, where l_s is the keyword and p is the number of keywords.

Let $Weight_{High}$, $Weight_{middle}$, $Weight_{low}$ be the weights for each level.

Algorithm 2 shows the pseudocode for the algorithm.

Algorithm 2 Get integrate risk value

input: *data*
output: *riskvalue*

function JUDGERISK(*data*)
  for *i* ← 0 **to** $Length_{HighSet}$ **do**
    if data.contain(*HighSet*[*i*]) **then**

```
            riskvalue ← WeightHigh
            return riskvalue
        end if
    end for
    Num_middle ← FINDTOTALNUM(MiddleSet, data)
    Num_low ←FINDTOTALNUM(LowSet, data)
    riskvalue ← Num_middle × Weight_middle + Num_low × Weight_low
    return riskvalue
end function
```

```
Input: KeywordsSet, data
Output: num
function FINDTOTALNUM(KeywordsSet, data)
    Num ← 0
    for p ← 0 to Length_KeywordsSet do
        Num_temp ← 0
        for q ← 0 to Length_data do
            index ← data.indexof(KeywordsSet[p], q)
            if index! = –1 then
                k ← index
                Numt_emp ← Num_temp + 1
            end if
        end for
        Num ← Num + Num_temp
    end for
    return Num
end function
```

Analysis:

Statistical analysis based on gradient learning provides $Weight_{low}$, $Weight_{middle}$ and $Weight_{high}$ to estimate the risk value of malicious keywords in a URL. The $Weight_{low}$ and $Weight_{high}$ are constant values that are determined first. Algorithm 1 generates the $Weight_{middle}$ value for the best result detected. It will obtain an appropriate value for $Weight_{middle}$ to evaluate the level of URLs that exhibit malicious behaviour. The integrated risk value will then be calculated by Algorithm 2 based on the $Weight_{low}$, $Weight_{middle}$ and $Weight_{high}$ scores determined by Algorithm 1. The integrated risk value distinguishes abilities that not only take effect at the critical point between the normal and the malicious but also differentiate the malicious. The integrated risk feature value is the most important feature that uses the idea of blacklist matching.

3.3.2 Feature selection based on hyperbolic tangent

A good selection of features is important to obtain better performance for the subsequent machine learning operations (Dogru and Subasi, 2012). This section describes how to systematically choose a proper set of features based on the hyperbolic tangent function. The criteria are that the set of chosen features must be able to present the essential characteristics of malicious web traffic while the level of generality for the features should not be too low (Akbani et al., 2004). Algorithm 3 describes the principle of how to select the best performance features from the feature database.

Notations:

Let f be a feature in an instance of data.

Let $vector = \{f_1, f_2, \ldots, f_m\}$ be a set of data features where f_s represents the individual features and where m is the number of features in an instance of data.

Let $Instance = \{vector, r_l\}$ be an instance of data where the vector contains its features and r_l is the real label for each instance.

Let $DataSet = \{Instance_1, Instance_2, \ldots, Instance_w\}$ be the vector dataset. Each URL will be parsed to an instance based on the feature database and the real label will be added after the feature vector. Here, w is the number of instances in the dataset.

Let F_n be the candidate feature and N be the number of candidates. After an F_n completes the detection process, the feature database is updated.

Let M_{Inst} be the number of instances in the $DataSet = \{Instance_1, Instance_2, \ldots, Instance_w\}$.

Algorithm 3 Feature selection

```
input: DataSet, CandidateFeatureSet
output: updateInstance
function FE_SELECTION(DataSet, CandidateFeatureSet)
    for n ← 0 to N – 1 do
        PNum_previous ← (RIGHTNUM (DataSet, Instance)) / M_Inst
        updateInstance ← Instance
        add F_n to updateInstance where f_m+1 ← F_n
        updateInstance ← {f_1, f_2, …, f_m, F_n, r_l}
        PNum_update ← RIGHTNUM(DataSet, updateInstance) / M_Inst
        Parameter ← (PNum_update – PNum_previous) / [(1 – PNum_update) * PNum_update]
        Support_m ← (e^Parameter – e^–Parameter) / (e^Parameter + e^–Parameter)
        if Support_m <= 0 then
            delete F_n from updateInstance
        end if
    end for
    return updateInstance
end function
```

Analysis:

The value of $PNum_{update} - PNum_{previous}$ indicates the difference in the accuracy rate after adding a new feature f_i. The higher the value of $PNum_{update} - PNum_{previous}$, the more important the role that f_i plays in the machine learning process. We place more emphasis on the accuracy rate

improvement from adding f_i and, thus, use ($PNum_{update}$ – $PNum_{previous}$) / $PNum_{update}$ to indicate the proportion that $PNum_{update}$ – $PNum_{previous}$ contributes to $PNum_{update}$. Considering 1 / (1 – $PNum_{update}$), 1 – $PNum_{update}$ is the false rate after adding f_i. If f_i has an obvious effect, the $PNum_{update}$ will increase and 1 / (1 – $PNum_{update}$) will increase correspondingly. 1 / (1 – $PNum_{update}$), on the other hand, shows the importance of f_i. Finally, ($PNum_{update}$ – $PNum_{previous}$) / [(1 – $PNum_{update}$) * $PNum_{update}$] functions as a parameter for the hyperbolic tangent function shown below:

$$S(x) = \frac{e^{x} - e^{-x}}{x^{x} + e^{-x}} \tag{3.1}$$

$S(x)$ is a threshold function that is continuous, smooth and strictly monotonic. A hyperbolic tangent function is used to test the significance of features. As shown in the preceding equation, $S(x)$ is the hyperbolic tangent function and x is the unique parameter to that function. When x tends to positive infinity, the value of S tends to 1, or conversely, tends to –1. In addition, when x is equal to 0, the value of S is 0. Therefore, 0 is the threshold for judging the significance.

$$x = \frac{PNum_{update} - PNum_{previous}}{\left(1 - PNum_{update}\right) \times PNum_{update}} \tag{3.2}$$

($PNum_{update}$ – $PNum_{previous}$) / [(1 – $PNum_{update}$) * $PNum_{update}$] will be the value of x in the hyperbolic tangent function $S(x)$. $S(x)$ is the threshold and is used to determine whether the feature f_i is significant for machine learning.

4 Feature selection

The two algorithms of statistical analysis based on gradient learning and feature selection based on a sigmoidal threshold level, respectively, are introduced in this section.

First, Algorithm 1 is implemented to determine the $Level_{middle}$ value based on the accuracy rate. Initially, setting $Level_{low}$ = 1 and $Level_{high}$ = 5, the result of Algorithm 1 calculated that $Level_{middle}$ = 2. When the URL does not contain a value matching $HighSet[i]$, the following calculation can be made:

$$Riskvalue = Num_{middle} \times 2 + Num_{low} \times 1 \tag{4.1}$$

Riskvalue is regarded as the initialised feature generated by Algorithm 2. Second, Algorithm 3 is implemented to select features based on the sigmoid function using the parameter calculated in Algorithm 3. We choose features in which the value of the sigmoid function is greater than 0.5. Finally, Algorithm 3 extracts 21 features from among the 40 features available.

4.1 Structure extraction

This paper focuses on the analysis of URL requests. It divides URL requests into the URI field (the path portion) and the parameter portion at the question mark ('?') and then uses the ampersand character ('&') to divide the parameter field into name-value pairs (Zhang et al., 2008). In every name-value pair, the name is on the left side of the equals sign ('=') and the value is on the right side of the '='. An example URL request is shown below:

- '/modules/article/wapreader.php?aid=19605 &format=json&type=6&imagemode=byte'

In the above URL request,

- '/modules/article/wapreader.php'

is the URI content, and

- 'aid=19605 &format=json&type=6&imagemode=byte'

is the parameter content. In the parameter content, 'aid', 'format', 'type' and 'imagemode' are the names and the values are '19605', 'json', '6', 'byte', respectively.

Therefore, based on these four dimensions, we test and validate the algorithms using the statistical analysis of gradient learning and feature selection based on a sigmoidal threshold level. Finally, 21 features are selected and classified to five feature types that include:

1 the length of features

2 the number of features

3 the type of features

4 the risk level of features

5 other advanced features.

Table 2 shows the 21 features we used in this paper.

Table 2 Extracted URL features

No.	*Feature name*	*type*
1	URI length	Numeric
2	URI number	Numeric
3	URI maxlen	Numeric
4	URI type	Nominal
5	URI risk value	Numeric
6	Parameter length	Numeric
7	Parameter number	Numeric
8	Name type	Nominal
9	Name maxlen	Numeric
10	Name risk value	Numeric
11	Value type	Nominal
12	Value maxlen	Numeric
13	Value risk value	Numeric
14	Digit percentage	Numeric
15	Letter percentage	Numeric
16	URL unknown amount	Numeric
17	If contain keywords	Nominal
18	Nginx test	Nominal
19	Value contain IP	Nominal
20	Html test	Nominal
21	Simple SQL injection test	Nominal

4.2 Feature analysis

This section contains a more detailed explanation of the 21 selected features.

4.2.1 The length features

The length features contain three length measurements that describe different features from various viewpoints, namely, the total length (‘URI length’), the ‘Parameter length’ and the maximum lengths of features (‘URI maxlen’, ‘name maxlen’, and ‘value maxlen’). No one feature can accurately represent the entirety of the length characteristics because they exist separately in the number of URI components and the number of parameters. For example, the following URIs have the same total URI length and number of URI components, but they are obviously different. In this case, the URI max length feature works to distinguish them.

a /question/info/name/

b /questioninfo/na/me/.

4.2.2 The number features

The ‘URI number’ and ‘parameter number’ refer to the number of features in the URI portion and the parameter portion. The URI number is the count of the URI segments divided by a forward slash (‘/’) while the parameter number is the count of the name-value pairs divided by the ampersand (‘&’). For example, for the following URL,

- ‘/modules/article/wapreader.php?aid=19605 &format=json&type=6&imagemode=byte’

the URI number is 3 and the parameter number is 4. The number of features combines with the length features to express the basic characteristics of a URL.

4.2.3 The type features

As we all know, a string can contain digits, letters, or other characters. A normal URL string is usually composed of letters, digits and some standard characters such as underscores (‘_’) and hyphens (‘-’). In contrast, a malicious URL always contains abnormal characters such as single quotes (‘‘‘), less-than signs (‘<‘), curly brackets (‘{‘), dollar signs (‘$’), extra ampersands (‘&’), and so on. Therefore, based on the composition of the URL string, we can extract the following type features and their values as shown in Table 3. The ‘unknown’ type means that the string contains more than one abnormal character and refers to characters other than hyphens (‘-’) and underscores (‘_’). The type features include ‘URI type’, ‘name type’ and ‘value type’. In addition, the feature ‘URL unknown amount’ accounts for the number of abnormal characters.

4.2.4 The risk value features

Table 4 shows malicious keywords that appear in the suffix (the file extension) portion of the URI field. The typical attack behaviour is to reference an illegal file in which the filename extension is one of the keywords shown in Table 4.

Table 3 Context structure of a string

Type	*Value*	*Example*
Empty	0	“”
Pure digitals	1	213456
Pure letters	2	username
Pure normal characters	3	_-
Digitals and letters	4	username213
Digitals and normal characters	5	213-456
Letters and normal characters	6	user_name
All contain	7	user_name-213
Unknown	8	Test’ and (select

Table 4 High risk level malicious keyword file extensions in URIs

.bashrc	.mdb	.asax	.frm	.inc
.bak	.htaccess	.swp	.tmp	.backup
.err	.php~	.asp~	.aspx~	.jsp~
.cvs	.svn	.git	.ini	.bash_history
.sql	.myd			

Table 5 shows the high risk level keywords that may appear not only in the URI field but also in a name field or value field.

Table 5 High risk level malicious keywords in URLs

java.lang	ascii(	sleep(	php://filter	eval(
length(	hex(	prompt(	updatexml(	phpinfo
ofc_upload_image	base64	md5(	data://text/plain	/web-inf/
utility/convert	../	xp_	updatexml(	redirect:
information_schema	sysobject	substr	benchmark	waitfor
shutdown	<scrip	<iframe	@@version	exec(
php://input	wget	utl_	Extractvalue(	alert(
response.write(	/etc/			

Table 6 Low risk level malicious keywords in URLs

and	or	like	from	insert
update	create	else	exist	table
database	master	echo	print	where
ping	exec	system	eval	sleep
href	[]	()		

Table 7 Malicious keywords of middle risk level in URL

select	if(	union	regexp	group by
count(	/**/	char(	drop	delete
chr(	onclick=			

Table 6 and Table 7 show the low risk and middle risk level keywords. These keywords are less severe than the high risk level keywords but they usually coexist in malicious URLs. If a URL contains one high risk level keyword, it has a greater than 90% probability of being malicious, but if a URL contains one middle risk level keyword, it has less than a 40% probability of being malevolent. The low risk level keywords do not pose a threat when they appear alone because many normal URL also contain these keywords (Johnson and Shanmugam, 2011). On the other hand, a URL that contains greater amounts of middle and low level keywords increase the probability that it is malicious.

The risk level features contain 'URI risk value', 'name risk value' and 'value risk value'. When calculating the values of these features, the first task is to detect any high risk key words. If the field contains more than one high risk malicious keyword, the value of the feature is 5. Otherwise, the process continues to detect the middle and low risk words as follows. Let $N(m)$ denote the number of middle risk keywords, and let $N(l)$ denote the number of low risk keywords in the detection field. The value of this feature is then calculated by the formula $N(m)*2 + N(l)*1$. The detailed computing process is described by Algorithm 2.

The feature 'If contain keywords' signifies whether the URL contains high risk level keywords: when it does, the value is 1; otherwise, the value is 0.

4.2.5 The other advanced features

The features 'digit percentage' and 'letter percentage' calculate the percentages of pure digits and pure letters in the entire URL. These two features help eliminate the influence of one or more abnormal characters. The larger the digit or letter percentage is, the higher the probability is that the URL is normal. The digit percentage is calculated as follows:

$$\frac{DNum(URI + NAME + VALUE)}{Num(URI) + Num(parameter) * 2} * 10 \quad (4.2)$$

where *DNum* is the number of pure digits. The letter percentage is defined by equation (4.3), where *LNum* is the number of pure letters.

$$\frac{LNum(URI + NAME + VALUE)}{Num(URI) + Num(parameter) * 2} * 10 \quad (4.3)$$

The feature 'Value contains IP' utilises regular expressions to detect whether an IP address appears in a value field. The regular expression to detect an IP address is '\D((25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-9]{1}|[1-9])\.(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-9]{1}|[1-9]|0)\.(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-9]{1}|[1-9]|0)\.(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-9]{1}|[0-9]))'.

The feature 'Nginx test' detects whether a URL is an Nginx attack. The regular expression to find such attacks is '.*\\..*\\/*.*\\.php'.

The feature 'Html test' detects whether the URL contains html tags. The regular expression '.*<(\\S*?)[^>]*>.*?</\1>.*|.*</.*?>.*' and '.*<[a-zA-Z]+(\\s+[a-zA-Z]+\\s*=\\s*(\"([^\"]*)\".*|.*\'([^\']*)\'))*\\s*/>.*' match various HTML tags such as '<a>username' and '<a username />'.

The feature 'simple SQL injection test' detects whether the URL contains a SQL injection attack. The regular expression used to detect this type of attack is '.*(\\b(select|and|or|exec|insert)\\b).*'. Note that this feature alone cannot accurately detect SQL injection attacks, but SQL injection keywords are also listed in the middle and low risk malicious keywords; therefore, SQL injection attacks can be detected using this feature in combination with the risk level feature.

5 Implementation and experiment

This section describes the dataset used for the evaluation and introduces the experiment that tests multiple algorithms. It also includes the proofing process and the results.

5.1 Dataset

Our data are all from well-known internet companies in China that specialise in network security. These companies protect the security of numerous websites and capture approximately 2TB of web logs everyday.

In the training dataset, we collected benign URL features on May 19, 2015 and malicious URL features between March 1, 2015 and May 31, 2015. The total number of benign URLs was 2,953,700 and the total number of malicious URLs was 356,215. During the parsing process, the pre-treatment was used to filter out some static files in URLs and reduce the complexity at the same time; these URLs are not pertinent for our purposes. We also filtered duplicate input data. At the end of this process, the URL requests used for testing the machine learning algorithms totalled 138,925 benign URLs and 24,520 malicious URLs. The benign URLs are labelled 'no' and the malicious URLs are labelled 'yes'.

To test our system's ability to detect variants of malicious URL, we tested raw web data between October 15, 2015 and October 29, 2015 using our detection model in various phases. The model exhibited good performance during this period.

5.2 Data parsing

Parsing data is a basic and important process in developing a machine learning application. Valid clean data must be input to the algorithm and some prior algorithm-specific formatting was required to make sure the algorithm would work as expected (Ageev et al., 2015). The following steps were taken to prepare the input data.

5.2.1 Preprocessing

To clean the available input data, the preprocessing step was designed as follows:

1 convert all capital letters to lower case

2 unify the coding format by converting all the input to UTF-8

3 filter static files in the suffix URI field for benign URL requests such as '.jpg', '.png', '.jco', '.mp3', '.js', '.gif' and so on

4 replace double slashes (//') with single slashed ('/').

5.2.2 Feature analysis

In this section, the 21 features we extracted were parsed. Each URL string is parsed to a vector that contains 21 tuples and its label; each tuple represents one relevant feature result. For example, a benign URL such as

- '/ajax.php?n=service&h=show&chatfromid=20787195 &chatorid=20036292'

would be parsed to '9, 1, 8, 8, 2, 0, 54, 4, 12, 2, 10, 1, 4, 8, 0, 2, 7, 0, 0, 0, 0, 0, 0, no', and a malicious URL such as

- '/celive/live/deals?end_time=1&searchName=' and 1=1 and ''='&start_time=1'

would be parsed to '18, 3, 6, 5, 2, 0, 53, 3, 17, 6, 10, 0, 8, 18, 5, 2, 4, 6, 1, 0, 0, 0, 1, yes', where 'no' is the label for benign URLs and 'yes' is the label for malicious URLs.

5.2.3 Data duplication filtering

Removing duplicate input data reduces the amount of data that must be processed and improves execution speed primarily because we guarantee each URL has the same probability during algorithm training. Therefore, we can regard identical parsed results as one result for the subsequent steps.

5.2.4 Data quantification

There are many 'numeric' type values among the 21 features that are related to length and count. This type of value can be widely distributed from 0 to 20,000 or even larger. Therefore, we use log2 to reduce the value and make it easier to handle.

After these four steps, we ended up with 138,925 benign URLs and 24,520 malicious URLs, which produced a vector of 163,445*22 (163,445 is 138,925 benign URLs plus 24,520 malicious URLs times 22 which is the 21 features plus the label). This vector was fed into the machine learning algorithm.

5.3 Experimental results

5.3.1 Training approach

Multiple classification algorithms were tested during the training phase including naive Bayes, decision tree and the support vector machine. Additionally, self-refit testing and ten-fold cross-validation were tested on the dataset and the accuracy of the three classifiers described in Section 3 was compared, each of which has different trade-offs between model complexity and training execution time. Self-refit testing uses the training dataset as the testing dataset; this function can verify the correctness of extracted features. The ten-fold cross-validation works as follows: the dataset is first randomly separated into ten equal groups or 'folds', nine of which are used for training. The remaining fold is used for testing. The experiment is performed ten times with ten different combinations of training and testing data, and the average of the ten different testing results is reported.

5.3.2 Calculation formula

To demonstrate the effectiveness and feasibility of our detection model, experiments were conducted to test the precision, recall rate and ROC area in detecting malevolent URL requests. To calculate the above three formulas, we collect the following statistics while processing the data.

a the total number of benign URLs that were correctly detected (TN)

b the total number of malicious URLs that were correctly detected (TP)

c the total number of benign URLs which tested to malicious (FN)

d the total number of malicious URLs which tested to benign (FP).

Table 8 Result distribution

	Yes	*No*	← *classified as*
Yes	True positive (TP)	False negative (FN)	Actual positive (TP + FN = 24,520)
No	False positive (FP)	True negative (TN)	Actual negative (FP + TN = 138,925)
	Predicted positive (TP + FP)	Predicted negative (FN + TN)	TP + FP + FN + TN = 163,445

5.3.3 The result of testing

We compared the results of three different classifiers: naïve Bayes (Aung and Naing, 2015), decision tree (Muniyandi et al., 2012) and SVM (Akbani et al., 2004). We used all the features for each classifier – in other words, the initial analysis process and the resulting input data were identical for all three classifiers. As described in the previous section, we performed self-refit and ten-fold cross-validation for each tested classifier.

Table 9 The testing result statistics

	Self-refit		*10-fold cross-validation*	
Naïve Bayes	TP = 23,901	FP = 619	TP = 23,898	FP = 621
	FN = 1,479	TN = 137,446	FN = 1,463	TN = 137,460
Decision tree	TP = 24,338	FP = 185	TP = 24,271	FP = 252
	FN = 145	TN = 138,921	FN = 258	TN = 138,808
SVM	TP = 23,738	FP = 785	TP = 23,739	FP = 784
	FN = 359	TN = 138,707	FN = 359	TN = 138,707

Table 10 The three measures of the testing results with self-refit

		Precision	*Recall*	*ROC area*
Naïve Bayes	YES	0.942	0.975	0.99
	NO	0.996	0.989	0.991
	Weighted avg.	0.987	0.987	0.991
Decision tree	YES	0.994	0.992	0.998
	NO	0.999	0.999	0.998
	Weighted avg.	0.998	0.998	0.998
SVM	YES	0.985	0.968	0.983
	NO	0.994	0.997	0.983
	Weighted avg.	0.993	0.993	0.983

Table 11 The three measures of the testing results with 10-fold cross-validation

		Precision	*Recall*	*ROC area*
Naïve Bayes	YES	0.942	0.975	0.99
	NO	0.996	0.989	0.991
	Weighted avg.	0.988	0.987	0.99
Decision tree	YES	0.989	0.99	0.996
	NO	0.998	0.998	0.996
	Weighted avg.	0.997	0.997	0.996
SVM	YES	0.985	0.968	0.983
	NO	0.994	0.997	0.996
	Weighted avg.	0.993	0.993	0.993

Figure 2 The results of finding a 'yes' from Table 10 and Table 11 (see online version for colours)

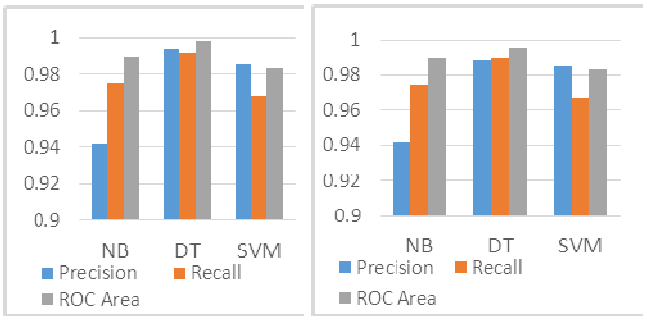


Figure 3 The results of finding a 'no' from Table 10 and Table 11 (see online version for colours)

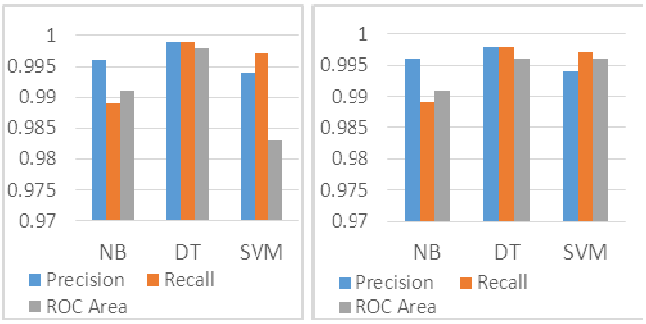


Figure 4 The results of finding a 'weighted avg.' from Tables 10 and 11 (see online version for colours)

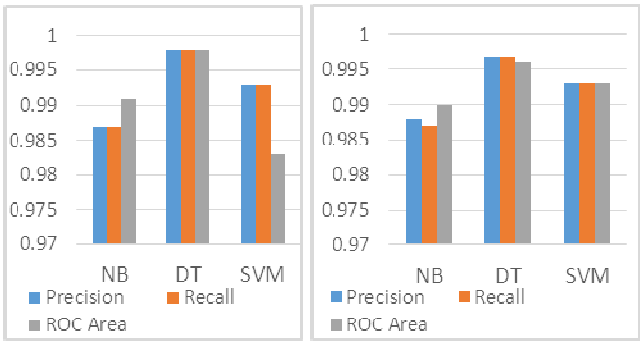


Table 12 The results of evaluating online malicious URLs

Date	*Number of malicious*	*Wrong detection*	*Recall rate*
10\15	50,051	525	0.9895107
10\16	47,707	504	0.9894355
10\17	43,130	487	0.9887086
10\18	55,481	519	0.9906454
10\19	59,904	559	0.9906684
10\20	49,350	563	0.9885917
10\21	56,828	661	0.9883684
10\22	56,983	594	0.9895758
10\23	47,464	491	0.9896553
10\24	42,005	527	0.9874539
10\25	47,726	457	0.9904245
10\26	49,706	503	0.9898805
10\27	56,687	573	0.9898919
10\28	56,390	526	0.9906721
10\29	48,225	490	0.9898393

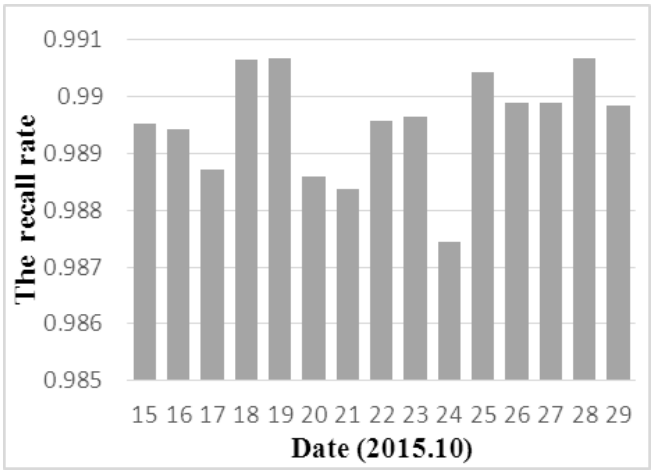
The test results are shown in Table 9. The total number of malicious URLs was 24,520 and the total number of benign URL was 138,925. Additionally, Table 10 and Table 11 show the precision, recall rate and ROC areas for self-refit and the ten-fold cross-validation. The decision tree classifier achieved the best performance for both self-refit and ten-fold cross-validation. As shown in these results, the precision scores are all above 0.98, which proves the validity of the feature extraction and data analysis from another aspect.

The following histograms describe the test results with the naïve Bayes, decision tree and SVM classifiers. The left histograms are the result of self-refit and the right histograms are the result of ten-fold cross-validation. The Y-axis expresses the results of the evaluation measures.

5.3.4 The result of practical use

Based on the testing results, the decision tree classifier was shown as preferable for use in the final machine learning algorithm and included in the detection model. In practical application, this system has been deployed online at well-known internet companies in China that specialise in network security. It processes an enormous number of URLs, approximately 2TB everyday. We tested it with a list of online malicious URLs encountered between October 15, 2015 and October 29, 2015, and it exhibited good performance. The accuracy is above 98.7% with only modest false positives.

Figure 5 The recall rate of the data shown in Table 12



6 Conclusions

This paper described a new approach for automatically classifying URLs as either malicious or benign based on machine learning techniques. By analysing the characteristics of malicious URLs, we used two statistical analysis algorithms that were based on gradient learning and automatic feature extraction based on a sigmoidal threshold level to solve the problem of identifying effective features. This process identified 21 state-of-the-art features from more than 40 possible features using these algorithms for validation. The experiment shows that this approach is better than blacklist methods, which cannot predict the status of previously unseen malicious patterns. Furthermore, we compared three classification algorithms and found that the decision tree algorithm has the best performance.

Compared with traditional detection methods, this system can accurately detect malicious URLs that have fuzzy characteristics. The detection model does not rely on any one characteristic or on a blacklist but is a comprehensive approach that utilises all the features for the machine learning. In addition, this approach has been developed into a practical project and deployed online to protect large volumes of web traffic.

In future work, the detection can be made more specific by also analysing the behaviours of malicious URLs. The precision may be improved to 99.99% by perfecting the malicious key words and extracting more features. The next step is to try to identify other types of web attacks that appear not only in URLs but also in user agent strings, cookies and so on.

Acknowledgements

This work was supported by NSFC-General technical joint basic research fund (No. U1536122) and the Fundamental Research Funds for the Central Universities (2014ZD03-03).

References

Ageev, S., Kopchak, Y., Kotenko, I. and Saenko, I. (2015) 'Abnormal traffic detection in networks of the internet of things based on fuzzy logical inference', in *2015 XVIII International Conference on Soft Computing and Measurements (SCM)*, May, pp.5–8.

Akbani, R., Kwek, S. and Japkowicz, N. (2004) 'Applying support vector machines to imbalanced datasets', in *European Conference on Machine Learning*, September, pp.39–50.

Aung, S.S. and Naing, T.T. (2015) 'Naïve Bayes classifier based traffic prediction system on cloud infrastructure', in *2015 6th International Conference on Intelligent Systems, Modelling and Simulation*, February, pp.193–198.

Bhuyan, M.H., Bhattacharyya, D.K. and Kalita, J.K. (2014) 'Network anomaly detection: methods, systems and tools', *IEEE Communications Surveys and Tutorials*, Vol. 16, No. 1, pp.303–336.

Blum, A., Wardman, B., Solorio, T. and Warner, G. (2010) 'Lexical feature based phishing URL detection using online learning', in *Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security*, October, pp.54–60.

Cui, B., He, S. and Jin, H. (2015) 'Multi-layer anomaly detection for internet traffic based on data mining', in *2015 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, July, pp.277–282.

Dogru, N. and Subasi, A. (2012) 'Traffic accident detection by using machine learning methods', in *Third International Symposium on Sustainable Development (ISSD'12)*, May, p.467.

Garera, S., Provos, N., Chew, M. and Rubin, A.D. (2007) 'A framework for detection and measurement of phishing attacks', in *Proceedings of the 2007 ACM Workshop on Recurring Malcode*, November, pp.1–8.

Han, H., Li, R. and Gu, X. (2016) 'Identifying malicious Android apps using permissions and system events', *International Journal of Embedded Systems*, Vol. 8, No. 1, pp.46–58.

Hou, Y.T., Chang, Y., Chen, T., Laih, C.S. and Chen, C.M. (2010). 'Malicious web content detection by machine learning', *Expert Systems with Applications*, Vol. 37, No. 1, pp.55–60.

Johnson, S. and Shanmugam, V. (2011) 'Effective feature set construction for SVM-based hot method prediction and optimisation', *International Journal of Computational Science and Engineering*, Vol. 6, No. 3, pp.192–205.

Kim, M.S., Kong, H.J., Hong, S.C., Chung, S.H. and Hong, J.W. (2004) 'A flow-based method for abnormal network traffic detection', in *Network Operations and Management Symposium, 2004, NOMS 2004, IEEE/IFIP*, April, Vol. 1, pp.599–612.

Kinder, J., Katzenbeisser, S., Schallhart, C. and Veith, H. (2005) 'Detecting malicious code by model checking', in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, July, pp.174–187.

Lifeng, Z., Ying, C. and Lu, Z. (2012) 'Detection and classification of calcifications in digital mammograms by multi-scale and multi-position', *International Journal of Computational Science and Engineering*, Vol. 7, No. 3, pp.224–231.

Ma, J., Saul, L.K., Savage, S. and Voelker, G.M. (2009a) '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*, June, pp.1245–1254.

Ma, J., Saul, L.K., Savage, S. and Voelker, G.M. (2009b) 'Identifying suspicious URLs: an application of large-scale online learning', in *Proceedings of the 26th Annual International Conference on Machine Learning*, June, pp.681–688.

Muniyandi, A.P., Rajeswari, R. and Rajaram, R. (2012) 'Network anomaly detection by cascading k-means clustering and C4.5 decision tree algorithm', *Procedia Engineering*, Vol. 30, pp.174–182.

Prakash, P., Kumar, M., Kompella, R.R. and Gupta, M. (2010) 'Phishnet: predictive blacklisting to detect phishing attacks', in *INFOCOM, 2010 Proceedings IEEE*, March, pp.1–5.

Shabtai, A., Moskovitch, R., Elovici, Y. and Glezer, C. (2009) 'Detection of malicious code by applying machine learning classifiers on static features: a state-of-the-art survey', *Information Security Technical Report*, Vol. 14, No. 1, pp.16–29.

Sommer, R. and Paxson, V. (2010) 'Outside the closed world: on using machine learning for network intrusion detection', in *2010 IEEE Symposium on Security and Privacy*, May, pp.305–316.

Yee, G., Korba, L., Lin, N.H. and Shih, T.K. (2005) 'Context-aware privacy and security agents for distance education', *International Journal of High Performance Computing and Networking*, Vol. 3, Nos. 5–6, pp.395–404.

Zhang, J., Porras, P.A. and Ullrich, J. (2008) 'Highly predictive blacklisting', in *USENIX Security Symposium*, July, pp.107–122.

Zhao, P. and Hoi, S.C. (2013) 'Cost-sensitive online active learning with application to malicious URL detection', in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August, pp.919–927.

Zhiwang, C., Jungang, X. and Jian, S. (2010) 'A multi-layer bloom filter for duplicated URL detection', in *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, August, Vol. 1, pp.V1–586.