IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 12, 2019, accepted May 28, 2019, date of publication June 4, 2019, date of current version June 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2920655

OFS-NN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network

ERZHOU ZHU, YUYANG CHEN, CHENGCHENG YE, XUEJUN LI, AND FENG LIU
School of Computer Science and Technology, Anhui University, Hefei 230601, China

Corresponding author: Feng Liu (fengliu@ahu.edu.cn)

This work was supported by the Natural Science Foundation of Education Department of Anhui Province, China, under Grant KJ2018A0022.

ABSTRACT Phishing attack is now a big threat to people's daily life and networking environment. Through disguising illegal URLs as legitimate ones, attackers can induce users to visit the phishing URLs to get private information and other benefits. Effective methods of detecting the phishing websites are urgently needed to alleviate the threats posed by the phishing attacks. As the active learning capability from massive data sets, the neural network is widely used to detect the phishing attacks. However, in the stage of training data sets, many useless and small influence features will trap the neural network model into the problem of over-fitting. This problem usually causes the trained model that cannot effectively detect phishing websites. In order to alleviate this problem, this paper proposes OFS-NN, an effective phishing websites detection model based on the optimal feature selection method and neural network. In the proposed OFS-NN, a new index, feature validity value (FVV), is first introduced to evaluate the impact of sensitive features on the phishing websites detection. Then, based on the new FVV index, an algorithm is designed to select the optimal features from the phishing websites. This algorithm is able to alleviate the over-fitting problem of the underlying neural network to a large extent. The selected optimal features are used to train the underlying neural network, and finally, an optimal classifier is constructed to detect the phishing websites. The experimental results show that the OFS-NN model is accurate and stable in detecting many types of phishing websites.

INDEX TERMS Information security, intrusion detection, machine learning, neural network.

I. INTRODUCTION

Phishing is now one of the fastest growing cyber attacks. Phishing attacks mainly utilize the social engineering and technology deception to obtain user privacy information. The most common way of phishing attacks is to send an illegal link to the user and induce the users to click. The users are then tricked by entering private information without their acknowledgement. At present, phishing attacks frequently appear in the personal computers and mobile platforms. Furthermore, phishing attacks are growing rapidly.–The Anti-Phishing Alliance of China (APAC) has reported that, at the last month of 2018, there is a total of 435193 phishing websites detected [1]. Effective methods of detecting the phishing websites are urgently needed to alleviate the threats posed by phishing attacks.

Generally speaking, the possible values of the phishing websites detection can be classified as a two-way problem: the ''phishing websites'' or the ''legitimate websites''. Since phishing attacks usually take advantages of users' careless behaviors or ignorance on using networking tools, it is a hard problem to be permanently resolved [2]. Aiming at mitigating the threat of phishing attacks, many approaches are proposed to train and educate end users to recognize and detect phishing URLs [3], [4]. These approaches take effect to some extent by periodically sending messages to warn end users with potential phishing threats. But they still

The associate editor coordinating the review of this manuscript and approving it for publication was Sungroh Yoon.

dependent on the users' behaviors and knowledge of utilizing the underlying systems [5].

Due to high accuracy and efficiency, the software based automatic method is widely used to detect phishing attacks. At present, the automatic phishing detection methods can be classified into four categories: the blacklist and whitelist methods, the heuristics methods, the visual similarity methods and the machine learning methods [6]. Through recording previously detected phishing or legal URLs, IP addresses and keywords, the method of constructing black and white lists can effectively prevent phishing attacks [7]. Due to small workload on analyzing the content of websites, this method has its advantage of requiring small resources on the underlying systems. However, since databases for storing black and white lists are constructed based on the previously detected URLs, this method has the difficulty in dealing with newly emerged phishing attacks.

The heuristic phishing detection approaches can be taken as the extension of black and white lists [8]. The heuristic approaches are usually based on assigned signatures for identified phishing attacks. Through scanning websites for the assigned signatures, this kind of approaches raises a warning if malicious behaviors are found [9]. Because of the ability of detecting newly emerged URLs, the heuristic approach exhibits better performance than the method of blacklist and whitelist. However, due to complicated nature of phishing attacks and the time consuming heuristic tests, this method tends to have higher false positive on phishing detection than the blacklist and whitelist method.

Through visually comparing the suspicious website with legitimate target, the visual similarity method can also achieve the similar accuracy as the blacklist and whitelist technique [10]. This method is based on catching snapshots of websites' appearances in the web browsers and sorting the acquired snapshots. However, these works may incur very high time and space costs.

Given the active learning capability from massive data sets, the machine learning method is widely used to detect the phishing websites. This method usually uses feature selection algorithm to extract a sensitive features vector that could help distinguish phishing and legitimate websites at first. Then, based on the extracted features, the underlying machine learning classifiers are trained to detect the phishing websites [11]. The classifiers are usually constructed based on the neural network model, the support vector machine model, the Naïve Bayes model, and so on. The machine learning method is accurate in phishing websites detection. Meanwhile, it also has the ability to adapt to newly emerged phishing websites. The key to the success of this method is to acquire highly qualified features from phishing URLs and their relevant websites [12]. However, improper selection of sensitive features will make the underlying classifier not able to precisely detect the phishing websites. Meanwhile, some useless or small impact features will cause machine learning methods falling into the problem of over-fitting.

This paper proposes OFS-NN, an effective phishing attacks detection model based on the optimal feature selection and neural network. Under this model, the new FVV index is firstly defined to evaluate the impact of sensitive features on phishing websites detection. Then, based on the FVV index, an algorithm is designed to select optimal features from phishing URLs and their relevant websites. Finally, the selected features are used to train the underlying neural network to build a classifier to detect phishing attacks. Generally speaking, the contributions of this paper are listed as follows:

(1) Defines a new index—FVV. In order to better evaluate the impact of a selected sensitive feature on detecting phishing attacks, this paper presents the FVV index. The new FVV is defined by combining the positive and negative features of URLs. Through calculating the FVV values, some useless or small impact features can be eliminated to improve the performance of the entire model.

(2) *Designs an optimal feature selection algorithm.* This algorithm calculates the FFV values of all features of the input URLs and their relevant websites at first. Then, a threshold is set to select sensitive features to construct an optimal feature vector. Through this algorithm, many useless and small influence features are pruned. Due to no disturbance from these redundant features, the over-fitting problem of the underlying neural network is alleviated. Meanwhile, this algorithm is also able to reduce the time cost of the process of phishing websites detection.

(3) *Presents the OFS-NN model.* Through the selected sensitive features and a large number of experimental analyses, the optimal structure of the neural network is trained and constructed as the final classifier of the OFS-NN model. This model is able to accurately detect many types of phishing attacks. Benefits from the powerful learning and fitting capabilities of the neural network, OFS-NN exhibits better performance than many existing systems in phishing websites detection.

This manuscript is the continuation work of our previous conference paper "An Effective Neural Network Phishing Detection Model Based on Optimal Feature Selection" [13]. Based on the conference paper, we completely re-write this new manuscript and add many new contents including the new section of the related work, the workflow of the OFS-NN model, experiments on analyzing our new FVV index and threshold ρ, experiments on finding the optimal structure of the underlying neural network, more metrics (Information gain and ROC curve) in evaluating the performance of the OFS-NN model, and so on. The remainder of this paper is organized as follows: Section 2 discusses the related work. Section 3 selects the optimal sensitive features. Section 4 trains the optimal neural network classifier and uses it to detect the phishing websites. Section 5 discusses the

experimental results of the OFS-NN model. Section 6 concludes this paper and outlines the future work.

II. RELATED WORK

In addition to the method of training and educating end users, software based automatic detection method is the most common way to ease the threat of phishing attacks.

As the most direct method, the black and white lists can mitigate phishing attacks with relatively low resource consumptions [14]. The Google Safe Browsing API is a representative work of blacklist that is constantly maintained by Google Inc. [15]. Through communicating with the established APIs, client applications can check whether the target URL is in the blacklists. Based on the whitelist, Kang [16] proposes a method to detect the phishing websites. This method organizes user access to the site by detecting URL similarity. It deals with local and DNS spoofing attacks by comparing DNS enquiry results. Sharifi and Siadati [17] proposes a blacklist generator method for detecting phishing websites. This method determines whether it is a phishing website by matching the domain name of the website and Google's search results. Han [18] sandboxes live phishing toolkits by measuring the impact of blacklisting services on phishing websites at the beginning when these services are installed. In order to mitigate the threats of newly emerged phishing URLs, Lee [19] proposes the PhishTrack framework for automatically updating the blacklist of phishing sites. The black and white lists consume small resources on the underlying systems. However, it cannot properly deal with the newly emerged phishing attacks [20]. In order to mitigate this shortage, as the work that is presented in this paper, the black and white lists are usually working in collaboration with other methods [21].

Due to active learning ability, machine learning methods are extensively studied to detect the phishing websites. Cantina [22] is a phishing attacks detection model built on 27 sensitive features extracted from URLs and their relevant websites. This model makes use of the TF-IDF algorithm to detect phishing attacks. This algorithm can detect many kinds of phishing attacks but at the expense of consuming much time cost of the underlying systems. Meanwhile, some legal websites are reported as phishing ones [23]. CANTINA+ [12] is the continuation work of Cantina. Compared with Cantina, CANTINA+ adds 10 more features. Meanwhile, the phishing detection TF-IDF algorithm is replaced with SMV. Through these improvements, shortages of Cantina are mitigated. However, the new CANTINA+ has a narrow range of applications [24].

As an automatic phishing detection model, PhishStorm [25] is implemented as an interface between social networking tools and email servers or HTTP proxies. In this model, a random forest classifier is trained by extracting 12 URL relevant features. However, due to small coverage of sensitive features, it cannot detect many types of phishing attacks. PhishShield [26] is implemented as heuristic phishing detecting tool running on top of personal computers. This tool is able to identify phishing attacks that are designed by changing contents of websites. Because of lacking active learning capacity from phishing attacks, it cannot properly deal with attacks that continuously change their sensitive features. The PhiDMA [27] model is implemented as a multi-filter. It consists of five levels of filters: the auto-upgrade whitelist layer, the URL feature layer, the lexical signature layer, the string matching layer and the accessibility rating comparison layer. This model detects phishing websites based on accessibility errors (known error, likely error and potential error) comparison. However, in some websites, the model has lacked the ability to encounter all the three errors. Meanwhile, this model only gets 92.72% phishing network detection accuracy. The Off-the-Hook [28] model is built as plugins to certain browsers. Through combining blacklist, machine learning method and 210 features, this model can detect many phishing attacks. However, too many sensitive features seriously degrade the performance of this model.

The machine learning method is able to accurately deal with newly emerged phishing attacks. However, it relies too much on the selected sensitive features. As a matter of fact, many existing machine learning methods use the collected sensitive features to training the underlying classifier. However, they do little work on pruning out useless and small impact features. The two kinds of features will increase the workload of the classifier. To this end, this paper defines the new FVV index to obtain optimal sensitive features. Merits from the selected features, the trained optimal neural network classifier (the core of the OFS-NN model) exhibits better performance than many existing models.

III. OPTIMAL SENSITIVE FEATURES GENERATION

In the Internet environment, URLs are the addresses for specifying websites. The common URL is made up of four components: the protocol, the domain name, the file path and the query parameters. According to the addresses specified by URLs, Internet resources (like documents, images and videos), can be accessed. Through extracting sensitive features from legitimate and phishing URLs and their relevant websites, the underlying neural network classifier can be trained for detecting phishing attacks. As shown in Fig. 1, the extracting algorithm is firstly employed to extract sensitive features from the input sample URLs and their relevant websites. Then, the selecting algorithm is used to generate the optimal features vector. The process of selecting optimal features can reduce the workload of training the underlying neural network classifier. Finally, the refined URL sample set with optimal features will be taken as the input to train the underlying classifier.

A. SENSITIVE FEATURE EXTRACTION

Illegal URLs are generally disguised as legitimate ones by phishing attackers. Through doing this, they can lure users to click to launch phishing attacks. Fortunately, different from legal URLs, phishing URLs have obvious

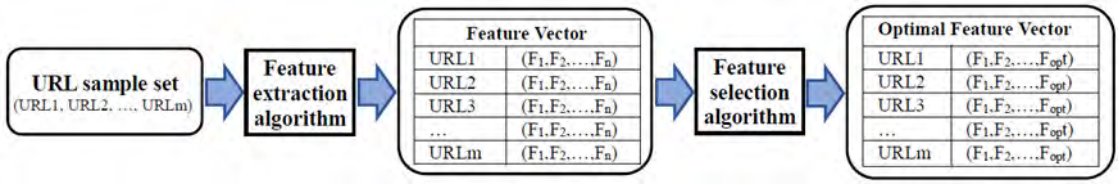


FIGURE 1. Workflow of generating optimal sensitive features from URLs and their relevant websites.

TABLE 1. Sensitive features for phishing attacks detection.

Category	No.	Phishing Item	Abbreviation	Value
Address bar features	F_1	URL contains an IP Address	having_IP_Address	-1, 1
	F_2	Length of the input URL	URL_Length	-1, 0, 1
	F_3	Uses URL shortening services	Shortining-Service	-1, 1
	F_4	URL contains '@' symbol	having_At_Symbol	-1, 1
	F_5	URL uses redirect symbol '//'	double-slash-redirecting	-1, 1
	F_6	URL contains '-' symbol	Prefix_Suffix	-1, 1
	F_7	Number of '.' in the URL	having_Sub_Domain	-1, 0, 1
	F_8	Protocol and SSL certificate status	SSLfinal_State	-1, 0, 1
	F_9	Domain registration length	Domain_register_length	-1, 1
	F_{10}	Favicon	Favicon	-1, 1
	F_{11}	Uses non-standard port	Port	-1, 1
	F_{12}	"HTTPS" token in the domain part	HTTPS_token	-1, 1
Abnormal features	F_{13}	Request URL	Request_URL	-1, 1
	F_{14}	URL of Anchor	URL_of_Anchor	-1, 0, 1
	F_{15}	Links in <Meta>, <Script>, <Link>	Links_in_tags	-1, 0, 1
	F_{16}	Server Form Handler	SFH	-1, 0, 1
	F_{17}	Submits Info. to email	Submitting_to_email	-1, 1
	F_{18}	Abnormal URL	Abnormal_URL	-1, 1
HTML and JavaScript features	F_{19}	Website forwarding	Redirect	-1, 0, 1
	F_{20}	Status bar customization	on_mouseover	-1, 1
	F_{21}	Disables right click	RightClick	-1, 1
	F_{22}	Uses pop-up window	popUpWidnow	-1, 1
	F_{23}	IFrame redirection	Iframe	-1, 1
Domain features	F_{24}	Age of domain	age_of_domain	-1, 1
	F_{25}	DNS record	DNSRecord	-1, 1
	F_{26}	Website traffic	web_traffic	-1, 0, 1
	F_{27}	PageRank	Page_Rank	-1, 1
	F_{28}	Google index	Google_Index	-1, 1
	F_{29}	Number of links pointing to page	Links_pointing_to_page	-1, 0, 1
	F_{30}	Statistical-reports based features	Statistical_report	-1, 1

identifiable features. As listed in Table 1, the sensitive features are divided into four classes: the address bar relevant features, the abnormal features, the HTML and JavaScript relevant features and the domain relevant features. In Table 1, each feature is assigned with a number. There are at most three values (−1, 0 or 1) for each of them.

1) ADDRESS BAR FEATURES

In order to detect phishing attacks accurately, it is necessary to check if the input URLs contain short addresses, IP addresses, and special characters (such as "@", "//", "-" and "."). Meanwhile, the lengths of the input URLs are all needed to be specified. As a matter of fact, many phishing attacks are launched by utilizing these features. Phishing attackers can replace legitimate addresses with short addresses to deceive users. During the process of a Web browser parsing URLs, contents before the "@" character are usually ignored. Through utilizing this feature, phishing addresses are usually appended after this character. The "." character is used to connect several sub-domains of a URL. Phishing attacks can be launched by utilizing these sub-domains.

In addition, it is also necessary to detect the port status, address bar icon source and SSL certificate status via URL. Under normal circumstances, this information on the phishing website will behave abnormally. Due to short life cycles, phishing URLs have shorter domain registration time and certificate time than legitimate ones. Furthermore, many phishing websites do not have the SSL certificates.

As listed in Table 1, $F_1 \sim F_{12}$ are the corresponding features to be tracked. F_1 is used to specify if the IP address is contained in the domain of the input URL. If the corresponding URL contains an IP address, F_1 is set to 1; otherwise, F_1 is set to -1. F_2 is used to specify the length of the input URL. If the corresponding URL contains more than 75 characters, F_2 is set to 1; if the corresponding URL contains less than 54 characters, F_2 is set to -1; otherwise, F_2 is set to 0. F_3 detects whether a short address is used in the URL. If the target URL uses a short address, F_3 is set to 1; otherwise, F_3 is set to -1. F_4 detects whether the target URL contains the '@' symbol. If the URL contains '@' symbol, F_4 is set to 1; otherwise, F_4 is set to -1. F_5 is used to detect whether the '//' symbol is included in the URL. If the symbol '//' appears after the seventh character in a URL, F_5 is set to 1; otherwise, F_5 is set to -1. F_6 specifies whether the '-' symbol is included in the URL. If the target URL contains the '-' symbol, F_6 is set to 1; otherwise, F_6 is set to -1. F_7 is used to specify the number of '.' symbols in an input URL. If this number is 1, F_7 is set to -1; if the number is 2, F_7 is set to 0; if the number is no less than 3, F_7 is set to 1. F_8 detects the HTTP protocol and SSL certificate statuses of the target website. If the age of the used "https" by the trusted issuer is no less than 1 year, F_8 is set to -1; if the used "https" by the untrusted issuer, F_8 is set to 0; otherwise, F_8 is set to 1. F_9 is used to specify the expiry time of domain name of the target website. If the domain expiry time is no more than 1 year, F_9 is set to 1; otherwise, F_9 is set to 0. F_{10} is used to denote the sources of "favicon/icon" of the target URL.

If the ''favicon/icon'' is loaded from other domains, F_{10} is set to 1; otherwise, F_{10} is set to -1. F_{11} is used to specify the situation of standard port numbers (such as 21, 22, 23, 80, 443, 445, 1433, 1521, 3306, etc.) are used in a website. If the port status is abnormal, F_{11} is set to 1; otherwise, F_{11} is set to -1. F_{12} detects whether the target URL has ''https'' strings in the domain name portion. If the domain contains ''https'', F_{12} is set to 1; otherwise, F_{12} is set to -1.

2) ABNORMAL FEATURES

The abnormal features can be extracted by analyzing the page contents of the input URLs. The relevant features can be acquired by checking the number of requested links on the page, the number of hyperlinked tags, the SFH status information, the ''WHOIS'' information and whether the page contains the mailbox information. For most of phishing websites, these features behave abnormally. It is also necessary to record the following abnormal behaviors: the segments between the <a> tag and the website are different; the null value of the SFH; the <Meta>, <Script> and <Link> tags point to different webpages; the input URL contains email address.

As listed in Table 1, $F_{13} \sim F_{18}$ are the corresponding features to be tracked. F_{13} is used to specify the percentage of resources from the same domain that is requested by a single URL. If the percentage is less than 22%, F_{13} is set to -1; if the percentage is greater than 61%, F_{13} is set to 1; otherwise, F_{13} is set to 0. F_{14} is used to detect pointing direction of the ''anchor'' in the webpage. An anchor is defined as the <a> tag in the HTML. If the ratio of the anchor in the webpage is less than 31%, F_{14} is set to -1; if this ratio is greater than 67%, F_{14} is set to 1; otherwise, F_{14} is set to 0. F_{15} detects links in tags in the HTML page of the website. If the ratio of links in tags (<meta>, <script> and <link>) is smaller than 17%, F_{15} is set to -1; if this ratio is greater than 81%, F_{15} is set to 1; otherwise, F_{15} is set to 0. F_{16} is used to detect the SFH of webpages. If the SFH contains a null character or ''about:blank'', it is set to 1; if the domain name of the SFH is different from the webpage, it is set to 0; otherwise, it is set to -1. F_{17} detects whether the webpage contains an email address. If the webpage contains the email address, it is set to 1; otherwise, it is set to -1. F_{18} detects the ''WHOIS'' information of the website. If the URL exists in the WHOIS database, it is set to 1; otherwise, it is set to -1.

3) HTML AND JAVASCRIPT FEATURES

In Table 1, $F_{19} \sim F_{23}$ are the relevant features needed to be extracted from source codes of the HTML and the JavaScript. The corresponding features are: the redirection and label elements numbers of a webpage; the ''onMouseOver'' event changes the status bar of the input URL; the right-clicking operation is disabled.

Specifically, F_{19} detects the number of pages redirected by the website. If the number of forwarding websites is no more than 1, F_{19} is set to -1; if this number is greater than 4, F_{19} is set to 1; otherwise, F_{19} is set to 0. F_{20} detects whether there is an ''onMouseOver'' event on the webpage. If the webpage has the onMouseOver event and the changes to the status bar, it is set to 1; otherwise, it is set to -1. F_{21} detects whether the webpage prohibits the right click of the mouse operation. If the mouse right click operation is prohibited, it is set to -1; otherwise, it is set to 1. F_{22} detects whether the webpage uses a pop-up window (using pop-up window). If it is used, it is set to 1; otherwise, it is set to -1. F_{23} detects the setting of the ''Iframe'' in the webpage. If the Iframe is used, it is set to 1; otherwise, it is set to -1.

4) DOMAIN FEATURES

Generally speaking, phishing websites have short life cycles. The corresponding DSN record of a phishing website cannot be found. Statistical indexes, like website traffic, PageRank and Google Index, measure the importance of websites. The phishing websites usually have small number of visitors. A URL may be a phishing one if the traffic is small. The PageRank index is a measure for evaluating the importance of webpages. Phishing URLs generally have zero or much smaller PageRank values than normal ones. Due to short life cycles, phishing URLs are rarely indexed by Google. Since the PhishTank usually publishes newly emerged phishing websites, it is necessary to determine if the target URL is in the database of the PhishTank.

As listed in Table 1, $F_{24} \sim F_{30}$ are the corresponding features to be tracked. F_{24} detects the age of a website. If the age of the domain name is less than 6 months, F_{24} is set to 1; otherwise, F_{24} is set to -1. F_{25} detects the DNS record of the website. If the domain has the DNS record, F_{25} is set to -1; otherwise, F_{25} is set to 1. F_{26} detects the website traffic. If the webpage rank is less than 100,000, it is set to -1; if the webpage rank is greater than 100,000, it is set to 0; otherwise, it is set to 1. F_{27} detects the PageRank of a website. If the PageRank value is smaller than 0.2, it is set to 1; otherwise, it is set to -1. F_{28} detects the Google Index of the website. If the target URL is included in the Google Index, it is set to -1; otherwise, it is set to 1. F_{29} detects the number of links pointing to the target webpage and specifies whether the link label on the URL pointing to the same domain. If the number of links pointing to the webpage is equal to 0, it is set to 1; if this number is less than 2, it is set to 0; otherwise, it is set to -1. F_{30} specifies whether the website is in the PhishTank or StopBadware statistical reports. If the URL is in the PhishTank or StopBadware reports, it is set to 1; otherwise, it is set to -1.

It is necessary to extract sensitive features from URLs and their corresponding websites and set them as the input of the proposed OFS-NN. Fig. 2 gives the algorithm for extracting the above features from the input URLs and their relevant websites. In Fig. 2, $F_v[1 \ldots 30]$ is the vector for storing 30 sensitive features of an input URL.

As shown in Fig. 2, the list of URLs is taken as the input of the algorithm (Line 1); then, for a given URL, the corresponding feature information is acquired (Line 3 - Line 5);

Input: UrlList;
Output: The set of F_v;
Each feature is assigned with a value in {1, 0, -1}.

```
(1)  Load(UrlList);
(2)  for each URL in UrlList
(3)    UrlInfo←URL.find();
(4)    PageInfo←URL.request();
(5)    DomainInfo←URL.requestDomain();
(6)    Fv←[0]×30;
(7)    Fv [1…12] ←extract Address bar features from UrlInfo;
(8)    Fv [13…18] ←extract Abnormal features from PageInfo;
(9)    Fv [19…23] ←extract HTML/JavaScript features from PageInfo;
(10)   Fv [24…30] ←extract Domain features from DomainInfo;
(11) end for.
```

FIGURE 2. Algorithm of extracting phishing sensitive features.

lastly, F_v is used to save the extracted sensitive features (Line 7 - Line 10).

B. OPTIMAL SENSITIVE FEATURE SELECTION

For the purpose of training an optimal model suitable for detecting phishing websites, it is needed to construct a concise sensitive feature vector based on the extracted sensitive features. In order to select optimal sensitive features effectively, this paper proposes the new FVV index. Through the FVV index, an algorithm for selecting optimal features can be designed for training the underlying neural network classifier.

1) DEFINITION OF THE FVV INDEX

Suppose there is a categorical data set $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}y^{(2)}), \ldots, (x^{(n)}, y^{(n)})\}$ containing n sample points, where x records the features vector and y records the results of classification. For a given feature A with the positive value, if it is also predicted as positive by a classifier, A is called the positive feature. On the other hand, the negative A is also predicted as negative by a classifier, A is called the negative feature. As a matter of fact, only positive features and negative features can provide the help on the decision of a classifier. Based on this observation, the FVV index is defined in equation (1) as follows:

$$FVV = P\left(A_{x-positive \text{ and } y=positive}\right) + P\left(B_{x=negative \text{ and } y=negative}\right) \quad (1)$$

In the phishing data sets, $P\left(A_{x=\text{positive and } y=positive}\right)$ indicates the probability that the feature value is represented as a phishing website and the detection result is also the phishing one; $P\left(B_{x=negative \text{ and } y=negative}\right)$ indicates the probability that the feature value is a legitimate website and the detection result is also the legitimate one.

As an example, Fig. 3 calculates the FVV values of 30 sensitive features (as listed in Table 1) for Data set 1 (in the first line of Table 3). Through calculating the FVV values, some useless and small impact features can be eliminated to reduce the workload of the whole OFS-NN model.

2) OPTIMAL SENSITIVE FEATURE SELECTION ALGORITHM

In the process of selecting optimal features, a threshold ρ for the value of FVV index is calculated to eliminate useless and small impact features. Based on the FVV index, the threshold ρ is defined in equation (2) as follows:

$$\rho = \sum_{i=1}^{m} FVV_i/(2 \times m) \quad (2)$$

In the formula, m represents the number of sensitive features; ρ is set as 1/2 of the average FVV values of all features. Formally, for an arbitrarily input data set, FVV values of all features of different URLs are calculated at first. Then, the threshold ρ can be calculated by comprehensively evaluating all the FVV values. In equation (2), many useful features may be pruned out when the threshold ρ is set to a relatively big value. On the contrary, many useless features may be included in the process of training of the underlying classifier. Moderate size of threshold is critical to the performance of the classifier [27].

In this equation, the value of ρ is dynamically changeable with different input data sets. As a matter of fact, different input data sets have different FVV values. Since the threshold ρ is derived from the FVV values, the value of ρ varies with the change of input data sets.

Following the example in Fig. 3, we calculate the accuracy (this metric will be defined in Section 5 of this paper) of our model when threshold ρ is set to different values. In this paper, as 30 sensitive features are collected, m is set to 30. As shown in Fig. 4, the values of accuracy are different when

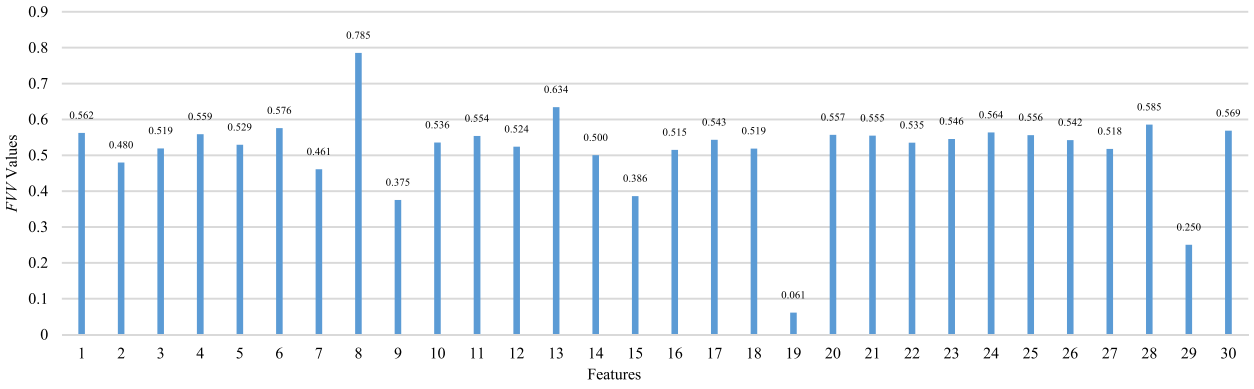


FIGURE 3. FVV values of 30 sensitive features of samples in Dataset 1.

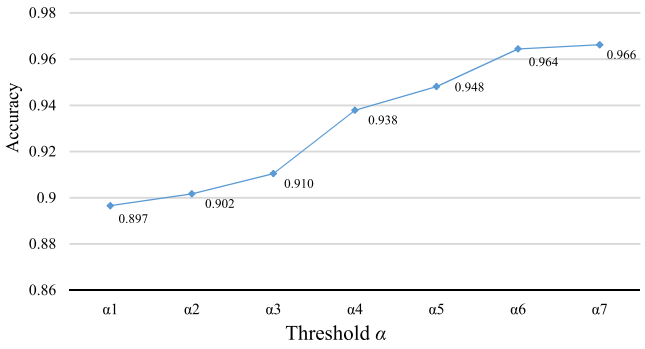


FIGURE 4. Model prediction accuracy in different thresholds.

Input: Feature vector F=(f_1, f_2, …, f_m); Threshold ρ;
Data set D = {($x^{(1)}$, $y^{(1)}$), ($x^{(2)}$, $y^{(2)}$), …, ($x^{(n)}$, $y^{(n)}$)}.
Output: Optimal feature vector F_{ofs}=(f_1, f_2, …, f_k).

(1) Initializes F_{ofs} and ρ;
(2) **for** i=1 to n **do**
(3) **for** j =1 to m **do**
(4) Calculates $P(A_{x=positive \text{ and } y=positive})$ and $P(B_{x=negative \text{ and } y=negative})$ **for** all features
(5) Calculates FVV values for each feature by equation (1) and stores them in an orderly table;
(6) **end for;**
(7) **end for;**
(8) **for** i=1 to m **do**
(9) **if** the FVV value of a feature > ρ (according to equation (2))
(10) F_{ofs}←(f_1, f_2, …, f_k);
(11) **end for.**

FIGURE 5. Optimal feature selection algorithm based on *FVV* index.

ρ is set to different values. Meanwhile, the accuracy of ρ increases slightly when the value of ρ exceeds ρ_6(0.257). So, the optimal value of ρ is ρ_6 (0.257) for the Dataset 1 (in Table 3).

In this example, if the threshold of FVV is set to 0.257, two small impact features (F_{19} and F_{29}) can be eliminated by the phishing websites sensitive features extraction (as shown in Fig. 2) and optimal features selection (as shown in Fig. 5) algorithms. As shown in Fig. 3, the FVV values of the two features are less than the threshold. So, the two features from the target data set can be deleted and use only 28 useful features to train neural network classifier. Specifically, for a given input URL, the 28-dimensional optimal features vector ($x_1, x_2, \ldots, x_{28}$) is extracted.

Through equation (2), the suitable threshold can be set for selecting optimal features for different data sets. Since FVV values of useless and small impact features are generally smaller than the average value of all features, they can be deleted for obtaining an optimal features vector. By the FVV index and the threshold ρ, the optimal feature selection algorithm is designed as shown in Fig. 5.

IV. NEURAL NETWORK CLASSIFIER

Generally speaking, the structure of a neural network model is composed of 3 layers: the input, the hidden and the output layers. Algorithms utilized by neural network generally incorporate two phases: the forward propagation and the backward propagation. The forward propagation starts to work when it receives a positive propagation signal. The backward propagation is invoked when the model detects the occurrence of evident deviation between the calculation result of the output layer and the actual value. The backward propagation utilizes the gradient descent algorithm to adjust parameters of different layers of neural units to minimize the deviation.

A. NEURAL NETWORK CLASSIFIER TRAINING

Generally speaking, more layers will bring better performance of the neural network. However, too many layers will trap the neural network into the problem of over-fitting. The over-fitting problem will seriously degrade the detection precision of the underlying neural network classifier. For this reason, the obtained structure of neural network must satisfy two requirements, with no over-fitting problem and better detection precision.

For the purpose of obtaining an optimal structure of the neural network, eight models with different layers and hidden units are compared. The eight models include: (1) 1-layer with no hidden unit; (2) 2-layer with 5-1 hidden units in each layer; (3) 3-layer with 20-5-1 hidden units; (4) 4-layer with 50-20-5-1 hidden units; (5) 5-layer with 100-50-20-5-1 hidden units; (6) 6-layer with 150-100-50-20-5-1 hidden units; (7) 7-layer with 200-150-100-50-20-5-1 hidden units; and (8) 8-layer with 300-200-150-100-50-20-5-1 hidden units. It is important to mention that before the stage of training the neural network classifier, the optimal features vector is generated for a given training data set. Fig. 6 shows the accuracy of the eight different models in different proportions of URLs. From this figure, it can be seen that the accuracy drops when the 8-layer neural network is used. The accuracy of the 7-layer neural network is the highest, which can get 98.49% classification accuracy.

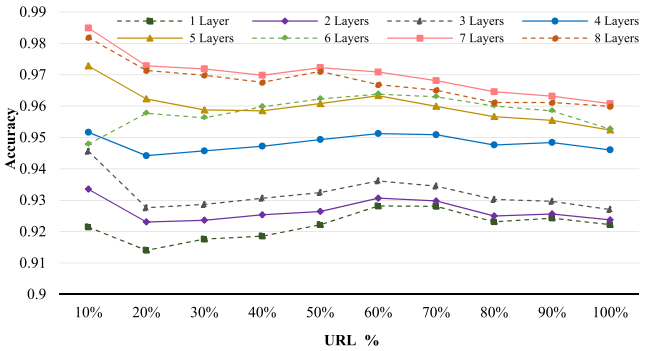


FIGURE 6. Accuracy comparisons of neural network models with different layers.

As a matter of fact, in the selection of structure of neural network, the number of neurons in the input layers and the number of neurons in the output layers are set as the number of selected features and the number of classification categories respectively. Meanwhile, with the deepening of the layers of the neural network, the number of the current hidden

units has to be ensured that they are bigger than the ones of the former layer.

Based on the above analysis, the 7-layer fully connected neural network is selected as the underlying classifier of our OFS-NN model. Specifically, it has 526 hidden units, 41305 ($26 \times 200 + 200 \times 150 + 150 \times 100 + 100 \times 50 + 50 \times 20 + 20 \times 5 + 5 \times 1$) weights and 7 biases. Table 2 gives the detailed configuration of the classifier. Fig. 7 shows the overall structure of the underlying neural network Classifier of our OFS-NN model.

TABLE 2. Detailed configuration of the neural network classifier.

ρ	Features	Layers	Hidden units
0.257	28	7	526

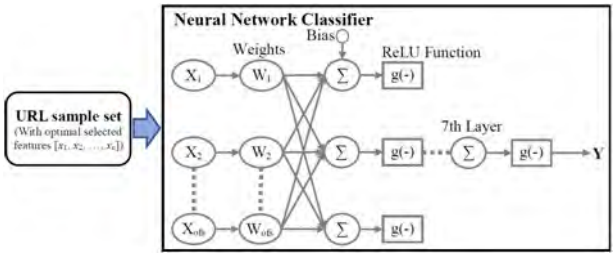


FIGURE 7. Overall structure of the classifier of the OFS-NN model.

Before the *Classifier* starts to work, the optimal feature selection algorithm is used to transform the original feature vector $(x_1, x_2, \ldots, x_n)$ into the optimal feature vector $(x_1, x_2, \ldots, x_{ofs})$ for each input URL. Consequently, the weight parameter sequence will also change to $(w_1, w_2, \ldots, w_{ofs})$. URLs with the generated optimal feature vector are taken as the input of the *Classifier*. Meanwhile, the bias b is input to the activation function $g(\cdot)$.The $g(\cdot)$ function is the *ReLU* activation function. The value calculated by the current layer of neuron activation function is then put to the next layer of neurons to participate in the operation. It iteratively calculates these functions until the result of the seventh layer is generated. The result of the seventh layer is then put to the $g(\cdot)$ activation function to generate the final result Y of the Classifier. The value of Y is also the detection result of the entire OFS-NN model. Specifically, Fig. 8 gives the algorithm on training the underlying Classifier.

In this algorithm, Step 1 initializes the weight sequence $(w_1, w_2, \ldots, w_{ofs})$ and the value of the offset b. Meanwhile, the appropriate activation functions are also selected. The *ReLU* function $(g_1(z))$ and *Sigmoid* function $(g_2(z))$ are selected as the activation functions for the hidden layer and the output layer respectively.

The *Sigmoid*, *Tanh* and *ReLU* are the commonly used activation functions. The *Sigmoid* function is commonly used in the last layer of the neural network. Although the *Tanh* function has the merits of fewer iterations and faster convergence speed than the other activation functions, but the usual occurrence of slowly weight updating will plunge this function into the problem of the gradient disappearance. As a

Input: Training data set, Test data set and Feature vector.
Output: Optimal neural network Classifier for detecting phishing attacks.

(1) Initializes the network parameters and selects the *ReLU* function: $g_1(z) = max(0, z)$, $g_2(z) = 1/(1 + e^{-Z})$.
(2) Performs the forward propagation: $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$, $A^{[l]} = g^{[l]}(Z^{[l]})$.
(3) Calculates the error of each layer (Adding the L2 regularization term can effectively alleviate the over-fitting problem): $J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} log\left(h_\theta\left(x^{(i)}\right)\right)_k + \left(1 - y_k^{(i)}\right) log\left(1 - h_\theta\left(x^{(i)}\right)\right)_k\right] + \frac{\lambda}{2m}||W||^2$.
(4) Performs the backward propagation: $dZ^{[l]} = dA^{[l]} * g^{[l]'}\left(Z^{[l]}\right)$, $dW^{[l]} = dZ^{[l]} \cdot A^{[l-1]}$, $db^{[l]} = dZ^{[l]}$, $da^{[l-1]} = W^{[l]^T} \cdot dZ^{[l]}$.
(5) Runs the gradient descent algorithm to update parameters of w and b until the convergence of the loss function: $W = (1 - \alpha \cdot (\lambda/m))W - \alpha \cdot dW^{[l]}, b = b - db^{[l]}$.
(6) Adjust the parameters (threshold, learning rate, number of layers, hidden layer units) to get the optimal neural network model.

FIGURE 8. Algorithm of the optimal neural network classifier training.

segment linear function, *ReLU* can alleviate the problem of gradient disappearance. Due to wide adaptability, small amount of calculation and fast convergence speed for loss function, the *ReLU* is the default and commonly used activation function in many neural network models.

There are also many improvements for the function of *ReLU*, such as the *Leaky ReLU*, the *PReLU* and the *ELU*. But these improved functions are not commonly used as the *ReLU* function. As a matter of fact, the *Leaky ReLU* is only used to resolve the saturation death in negative region problem of the*ReLU*. So, the *Leaky ReLU* is not often used as the activation function. The *PReLU* is only proposed to improve the fixed gradient problem of *Leaky ReLU*. So, it is also not used as the activation function frequently. Due to the negative saturation region, the*ELU* function can properly deal with noise points. However, the exponential computation seriously limits the application scope of this function.

Step 2 performs the forward propagation. Through taking the input values $A^{[l-1]}$, the weight W and the bias b, values of $Z^{[l]}$ and $A^{[l]}$ are calculated for each layer. In $A^{[l-1]}$, l indicates the number of layers of the neural network. At the first layer, A^1 is set as the input features X.

Step 3 calculates the *loss* function. The result of this function indicates the error between the predicted value and the target value. The target of training the neural network model is to reduce the value of this function. In order to alleviate the over-fitting problem in the training process, the regularization item $L2$ is added to the*loss* function. In this step, m is the sample size; λ is the regularization parameter; $h_\theta\left(x^{(i)}\right)$ is the prediction value and $y_k^{(i)}$ is the actual value.

Step 4 performs the backward propagation. In this step, it is necessary to calculate the gradient values (dA^l, dW^l, and db^l) for parameters A^l, W^l and b^l at first. Then, these gradient values are used to perform the backward propagation until the first layer is reached.

Step 5 runs the gradient descent algorithm to minimize the value of the *loss* function. In order to minimize this value, Step 2 - Step 4 are repeatedly executed until the *loss* function converges to a small value and the training result does not have an over-fitting. In this step, α is the learning rate.

Through repeatedly executing Step 2 - Step 5, the corresponding parameters (threshold, learning rate, layers and hidden units) can be continuously adjusted to obtain the optimal neural network Classifier.

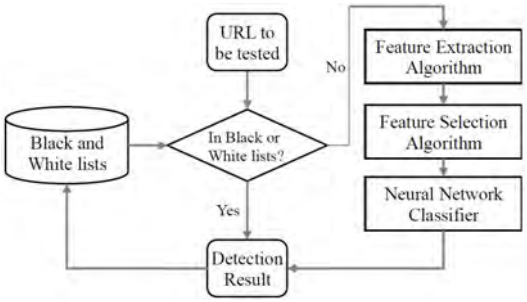


FIGURE 9. Workflow of phishing URL detection of the OFS-NN model.

B. PHISHING URLs DETECTION

Fig. 9 shows the workflow of detecting phishing attacks of the OFS-NN model. As is shown in this figure, a black and white list module is firstly introduced to reduce the time cost of the whole OFS-NN model. The whitelist is mainly constructed on the data selected from the Alexa websites. The blacklist is mainly constructed on the data selected from the PhishTank websites. At the beginning of this phase, it is necessary to check whether the input URL is in our library.

If the input URL is not in the black and white lists, the trained neural network classifier is used to process this URL. Firstly, algorithms of features extracting and selecting are used to generate optimal feature vector for this URL. Then, the refined URL with optimal sensitive features is taken as the input of the classifier. In the underlying classifier, parameters of weights and offsets are loaded to the neural network. For a given layer, the ReLU function multiplies the weight matrix w with the result of the former layer. And the sum of the multiplication result and the offset b form the output of the current layer. The output (Y) of the last layer is the final result of the entire classifier.

As the algorithm shows in Fig. 10, if $Y < 0.5$, the algorithm returns "-1" and marks this URL as a phishing one. Otherwise, the algorithm returns "1" and marks it as a legitimate one. Furthermore, the detection results are stored in the black or white lists for the future usage.

V. EXPERIMENTAL RESULTS

Experiments of this section are performed on a computer with Intel i5 CPU (3.4 GHz), DDR3 RAM (16GB) and Windows 10 operation system (64bit). Meanwhile, experimental programs are written by the Python programming language.

Input: F_v: Feature vector; ω: Weights of NN; b: Bias of NN; L: Layers number of NN.
Output: Y: Phishing detection result.

(1) **if** URL **in** Lists
(2) URL **in** BlackList return -1; //Legitimate
(3) URL **in** WhiteList return 1; //Phishing
(4) **else** NN model ← *LoadWeightsandBias(ω,b)*;
(5) $prediction[0] \leftarrow F_v$;
(6) **for** i **in** L **do**
(7) $Y[i-1] \leftarrow reLU(NN\ model.\omega \times Y[i-2] + b[i-2])$;
(8) $Y[i] \leftarrow Sigmoid(Y[i-1])$;
(9) **end for**
(10) Y ← $prediction[L]$;
(11) **if** Y < 0.5 **return** 1 //Phishing
(12) **else** **return** -1 //Legitimate

FIGURE 10. Phishing URL detection algorithm.

TABLE 3. Descriptions of the two tested datasets.

Data set	Attributes	Instances	Phishing Rate	Legitimate Rate
Data set 1	30	11055	44.31%	55.69%
Data set 2	28	14582	10.12%	89.88%

A. DESCRIPTION OF TESTED DATASETS

As listed in Table 3, two data sets are used to evaluate the performance of the proposed OFS-NN model. Data set 1 is selected form UCI phishing data set [29]. Data set 2 is built on PhishTank records and the Alexa records [30]. Data set 1 is composed of 11055 samples with 55.69% legal URLs and 44.31% phishing URLs. Meanwhile, 70% of total 11055 samples are used to train the Classifier and 30% samples are tested for evaluating the performance of the OFS-NN model. As a matter of fact, small number of training samples will trap the ultimate classifier into problems of under-fitting and weak generalization ability. On the contrary, the classifier will fall into problems of over-fitting and poor classification result when all samples of the data set are used for training. To this end, this paper adopts the conventional method to partition the training set and testing set. For example, the PNN model [6] uses 80% of total samples to train the classifier and 20% samples to test its performance; the Cantina model [22] uses 70% samples to train the classifier and 30% samples to test the performance.

Data set 2 (composed of 14582 samples with 10.12% phishing URLs) is used to evaluate the overall performance between our OFS-NN model and some existing phishing detection models.

Since the performance of the OFS-NN model is tested from many aspects on Data set 1, the detailed descriptions of this data set is given. Table 4 lists the distribution of data objects of Data set 1. For a given feature F_i, column "*Frequency*" gives the number of data objects belonging to each value of this feature. As there are 11055 instances in Data set 1, the total number of data objects belonging to all values of a feature is 11055.

As far as each value of a sensitive feature is concerned, Fig. 11 visualizes the distribution of legitimate and phishing websites. In each sub-graph of this figure, the blue histogram

TABLE 4. Distributions of samples in Data set 1.

Category	No.	Value	Frequency		
			-1	0	1
Address bar features	F_1	{-1, 1}	3793	--	7262
	F_2	{-1, 0, 1}	8960	135	1960
	F_3	{-1, 1}	1444	--	9611
	F_4	{-1, 1}	1655	--	9400
	F_5	{-1, 1}	1429	--	9626
	F_6	{-1, 1}	9590	--	1465
	F_7	{-1, 0, 1}	3363	3622	4070
	F_8	{-1, 0, 1}	3557	1167	6331
	F_9	{-1, 1}	7389	--	3666
	F_{10}	{-1, 1}	2053	--	9002
	F_{11}	{-1, 1}	1502	--	9553
	F_{12}	{-1, 1}	1796	--	9259
Abnormal features	F_{13}	{-1, 1}	4495	--	6560
	F_{14}	{-1, 0, 1}	3282	5337	2436
	F_{15}	{-1, 0, 1}	3956	4449	2650
	F_{16}	{-1, 0, 1}	8440	761	1854
	F_{17}	{-1, 1}	2014	--	9041
	F_{18}	{-1, 1}	1629	--	9426
HTML and JavaScript features	F_{19}	{-1, 0, 1}	0	9776	1279
	F_{20}	{-1, 1}	1315	--	9740
	F_{21}	{-1, 1}	476	--	10579
	F_{22}	{-1, 1}	2137	--	8918
	F_{23}	{-1, 1}	1012	--	10043
Domain features	F_{24}	{-1, 1}	5189	--	5866
	F_{25}	{-1, 1}	3443	--	7612
	F_{26}	{-1, 0, 1}	2655	2659	5741
	F_{27}	{-1, 1}	8201	--	2854
	F_{28}	{-1, 1}	1539	--	9516
	F_{29}	{-1, 0, 1}	548	6156	4351
	F_{30}	{-1, 1}	1550	--	9505

TABLE 5. The confusion matrix.

Actual value / Predict value	Phishing pages	Legitimate pages
Predicted as phishing	TP	FP
Predicted as legal	FN	TN

indicates the number of phishing websites, and the orange histogram indicates the number of normal websites. For example, for feature F_1 listed in Table 4, there are 3793 objects when F_1 has the value of -1 and 7262 objects when F_1 has the value of 1. Consequently, in Fig. 11, there are 1926 phishing websites and 1867 legal websites when F_1 has the value of -1; there are 2972 phishing websites and 4290 legal websites when F_1 has the value of 1. From this figure, it can be seen that a single feature cannot completely distinguish legal websites and phishing websites for this dataset.

B. METRICS USED IN EVALUATION

In this part, the confusion matrix is used to evaluate the performance of the OFS-NN model. As listed in Table 5, the matrix is composed of four indexes: *TP* (true positive), *TN* (true negative), *FP* (false positive) and*FN*(false negative). In this paper, the phishing websites are taken as positive samples; while the legitimate websites are taken as negative samples. Through this designation, *TP* is the number of phishing websites that are correctly predicted as phishing ones; *TN* is the number of legal websites that are correctly predicted as legal ones; *FP* is the number of legal websites that are wrongly predicted as phishing ones; *FN* is the number of phishing websites that are wrongly predicted as legal ones.

The *accuracy* rate is defined as the ratio of the number of correctly predicted URLs ($TP + TN$) to the number of all instances.

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \quad (3)$$

The *precision* rate is defined as the ratio of the number of phishing websites that are correctly predicted to the number of all the predicted phishing websites ($TP+FP$).

$$Precision = TP/(TP + FP) \quad (4)$$

The *recall* rate is defined as the ratio of the number of phishing websites that are correctly predicted to the number of all phishing URLs ($TP + FN$).

$$Recall = TP/(TP + FN) \quad (5)$$

The F_1_score is defined as the harmonic average of the recall rate and the precision rate.

$$F_{1\_}score = 2 \times (PrecisionRecall)/(Precision + Recall) \quad (6)$$

Among the four metrics defined by equation (3) - equation (6), the most important one is the *accuracy* as it reflects the overall performance of a classifier. The F_1_score is a comprehensive judgment metric. The value of the F_1_score is in the interval of [0, 1]. In this interval, the bigger value of F_1_score, the better performance of the underlying classifier.

C. PERFORMANCE EVALUATION

The performance evaluation in this subsection is composed of three parts, the performance of the new defined FVV index, the performance of the optimal sensitive feature selection method and the performance of the entire OFS-NN.

1) PERFORMANCE OF FVV

"*Information entropy*" is a common metric for measuring the purity of a dataset. Assuming that the proportion of the k^{th} sample in a data set D is $p(x_i)$, the information entropy of D is defined below.

$$E(D) = -\sum_{1 \le i \le n} p(x_i) \cdot \log_2 p(x_i) \quad (7)$$

Information gain (*Gain*) is a features selection metric for evaluating the performance of a classification algorithm. In general, the greater value of the *Gain*, the better the selectivity performance of the classification algorithm.

If an attribute a (a is supposed to have V possible values) is used to divide data set D, then V branch nodes are generated. Where, the V branch nodes contain all the values in data set D on attribute a. As in equation (8), the *Gain* indicator is calculated by the difference between the entropy of the sample set to be classified and the conditional entropy of a selected feature.

$$\text{Gain}(D, a) = E(D) - \sum_{1 \le v \le V} \left(|D^v|/|D|\right) \cdot E\left(D^v\right) \quad (8)$$

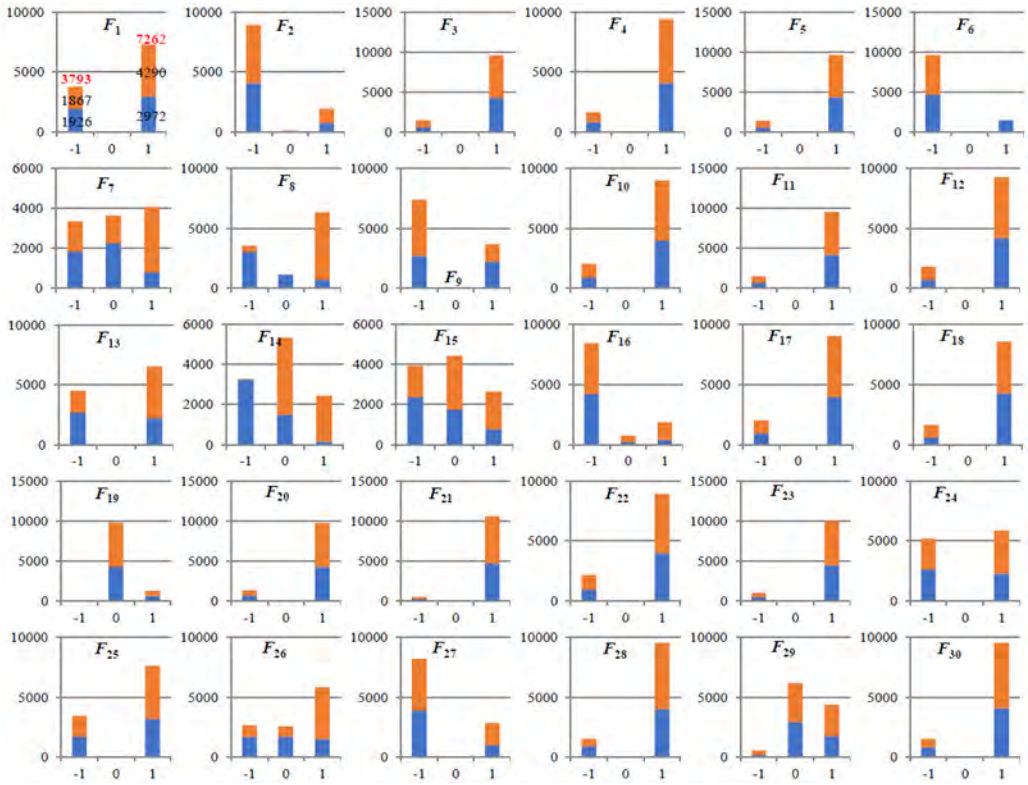


FIGURE 11. Distribution of legitimate and phishing samples of all features of Dataset 1.

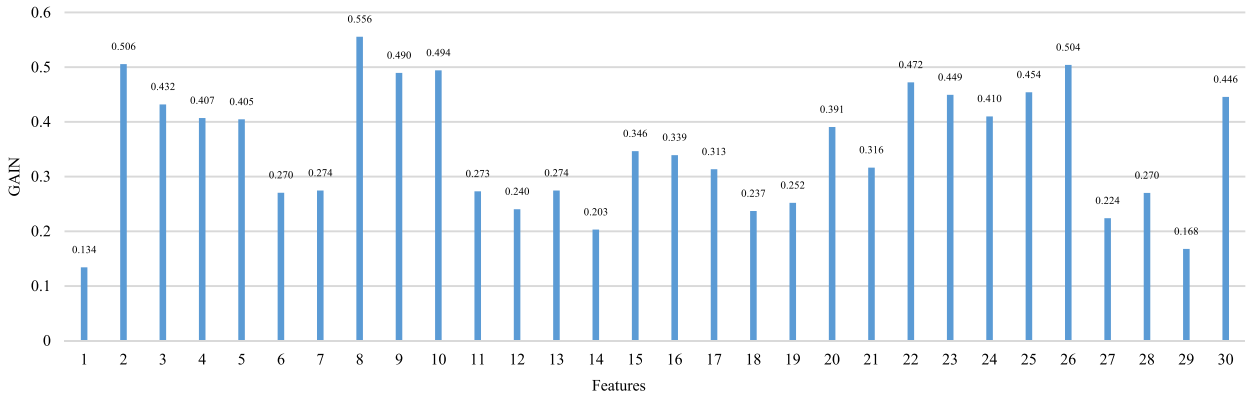


FIGURE 12. "*Information gain*" (*Gain*) values of 30 sensitive features of samples in Dataset 1.

In the formula, V represents the value range of the attribute a; D^v represents that all samples of sample dataset D in the v^{th} branch node take values on attribute a; $|D^v|/|D|$ is the weight of the branch node. Fig. 12 shows the values of the *Gain* indicator corresponding to the 30 sensitive features in Data set 1.

This paper compares the performance of the FVV index and the "*Information gain* (*Gain*)" on the optimal features selection. In the case of the same average threshold, the performance of our model after features selection is shown in the Table 6. In this table, thresholds are obtained by calculating the average of the corresponding indicators for all features. Through doing this, the feature selection ability of the indicator can be quickly evaluated. It can be seen from the table that FVV has better performance than the *Gain* index in evaluating the importance of features for phishing websites detection.

TABLE 6. Performance of the Gain an FVV indexes in phishing websites selection.

Items	Average threshold	Accuracy	Precision	Recall	F_1_score
Gain	0.352	0.900	0.899	0.924	0.911
FVV	0.513	0.945	0.964	0.936	0.950

TABLE 7. Performance of phishing detection under 4 categories of features and optimal selected features.

Features Category	Accuracy	Precision	Recall	F_1_score
Address bar	0.903	0.904	0.923	0.913
Abnormal	0.877	0.838	0.965	0.897
HTML/JavaScript	0.562	0.559	0.987	0.714
Domain	0.749	0.740	0.841	0.787
Optimal	0.945	0.964	0.936	0.950

TABLE 8. Average time cost evaluation of the optimal features selection method.

Items	OFS-NN	NN
Features selection time	1.105s	1.255s
Training time	0.170s	0.282s
Detection time	0.0000105s	0.0000293s
Total	1.2750105s	1.5370293s

2) PERFORMANCE OF OPTIMAL FEATURE SELECTION

This part evaluates the impacts of different categories of features on the performance of the phishing websites detection. The experimental results in Table 7 have shown that the highest category (Address bar relevant features) achieves 90.3% accuracy, while the worst category (HTML and JavaScript relevant features) only has 56.2% accuracy.

The last line of this table also gives the performance of the phishing websites detection by utilizing optimal feature (selection method (selecting optimal features from all the four categories of features). Through using the optimal feature selection method, the accuracy is improved to 94.5%. This experiment shows that the phishing websites detection performs poorly by features extraction of a single category. Therefore, for the better performance, it is necessary to select optimal sensitive features from all categories.

Table 8 lists the results of the average time cost comparisons between "*NN*" (our model without incorporating the optimal feature selection method) and "*OFS-NN*" (our neural network model that incorporates the optimal feature selection method). In this experiment, time costs of selecting features, time costs of training the neural network classifier and time costs of generating detection results are compared. From this experiment, it can be seen that OFS-NN reduces approximately 0.17s average time cost in total (the sum of time costs of "Features extraction time", "Training time" and "Detection time") when it is compared with the ones of the "*NN*". This means, our optimal feature selection method reduces about 17.1% time cost on average for processing URLs.

3) PERFORMANCE OF THE ENTIRE OFS-NN

This subsection evaluates the performance of the OFS-NN model under different scales of the tested data set at first. In this experiment, the proposed model is established as a function for different scale of Data set 1. It can be seen from Table 9, as the number of samples increases, the performance of the OFS-NN model is improved. In the case of small number of samples, the errors occur frequently. As the number of samples increasing, the occurrences of errors become smaller.

TABLE 9. Performance of OFS-NN under different scales of Data set 1.

Dataset	Accuracy	Precision	Recall	F_1_score
10%	0.9245	0.9245	0.9111	0.9292
20%	0.9382	0.9346	0.9528	0.9436
30%	0.9407	0.9669	0.9221	0.9440
40%	0.9487	0.9571	0.9492	0.9532
50%	0.9560	0.9587	0.9618	0.9603
60%	0.9563	0.9625	0.9581	0.9603
70%	0.9582	0.9511	0.9742	0.9625
80%	0.9619	0.9559	0.9761	0.9659
90%	0.9621	0.9657	0.9663	0.9660
100%	0.9644	0.9478	0.9902	0.9685

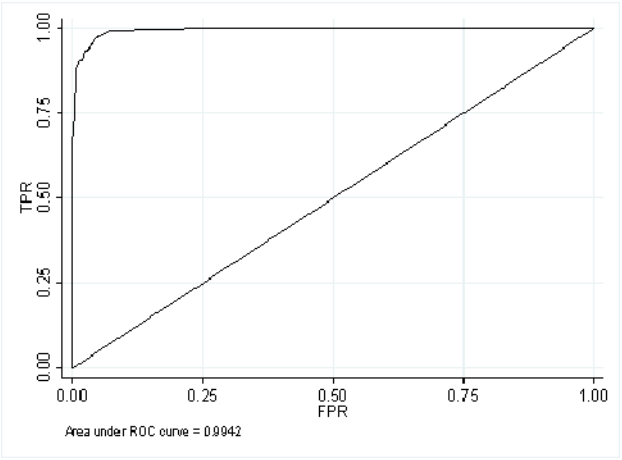


FIGURE 13. ROC curve and AUG value of the OFS-NN model.

When the changing of errors occurrences closed, the trained neural network structure is the optimal one. This structure has no problem of over-fitting or under-fitting.

The receiver operating characteristic (ROC) is a curve in which the false positive rate (*FPR*) is plotted on the abscissa and the true positive rate (*TPR*) is plotted on the ordinate. The ROC curve formed by the (0, 0) and (1, 1) connections represents a random classifier. If the curve drawn by the classification model is above the random classifier curve, it means that the classifier performs better. The area under ROC curve (AUC) value is a standard used to measure the quality of a classification model. The AUC value is the area covered by the ROC curve. Obviously, the larger the AUC value, the better performance of the classification effect of the tested model. If the AUC value is 1, the target classifier is the perfect one. However, in most cases of prediction, there is no perfect classifier. As a matter of fact, if the AUC value is 0.5, the prediction result of the tested classification model follows the probability of machine guessing and this model has no predictive value. When the AUC value is in the interval of (0.5, 1), the performance tested classification model is better than random guess and this model has predictive value.

Fig. 13 shows the ROC curve of the OFS-NN model. From this figure, it can be seen that, the AUG value of the OFS-NN is 0.9942. This means the OFS-NN model has good performance for classification.

Lastly, the overall performance of the OFS-NN model is compared with some existing phishing websites

TABLE 10. Performance comparisons between the OFS-NN and some existing models.

Models	Test data sets		Phishing Ratio	Accuracy	Precision	Recall	F_1_Score
	Legitimate	Phishing					
Cantina	2100	19	0.89%	0.969	0.212	0.89	0.342
Cantina+	1868	940	26.40%	0.955	0.964	0.964	0.964
Jain	405	1120	73.44%	0.894	0.993	0.861	0.922
Xiang	7906	3543	30.94%	0.955	0.957	0.900	0.928
Varshney	2500	500	16.66%	0.936	0.723	0.995	0.837
Kausar	71	89	55.62%	0.875	0.888	0.887	0.887
PhishStorm	48009	48009	50.00%	0.949	0.984	0.913	0.947
PhishShield	250	1600	86.48%	0.966	0.999	0.971	0.985
Off-the-Hook	200000	2000	0.99%	0.999	0.975	0.951	0.963
OFS-NN	13107	1475	10.12%	0.993	0.969	0.959	0.964

detection models. In Table 10, the data source of Cantina [22] is selected from ''Alexa/PhishTank''; the data source of CANTINA+ [12] is selected from ''Alexa/3Sharp''; the data source of Jain and Gupta [31] is selected from ''Alexa''; the data source of Xiang and Hong [32] is selected from ''PhishTank/Alexa/3Sharp/ Yahoo''; the data source of Varshney *et al.* [33] is selected from ''Alexa''; the data source of Kausar [34] is selected from ''PhishTank''; the data source of PhishStorm [25] is selected from ''DMOZ''; the data source of PhishShield [26] is selected from ''PhishTank''; the data source of Off-the-Hook [28] is selected from ''PhishTank/Intel Security''; and the data source of our OFS-NN model is selected from ''Alexa/ PhishTank''.

From Table 10, it can be seen that, except the Off-the-Hook model, the OFS-NN model has the higher accuracy than the other models. The Off-the-Hook model has the highest accuracy, but it has smaller *Recall* rate than our OFS-NN model. Meanwhile, the time cost of the OFS-NN model is much smaller than the ones of the Off-the-Hook model. As a matter of fact, the Off-the-Hook model collects 210 features (without optimal feature selection) from URLs and the relevant websites. However, too many features seriously degrade the performance of this model.

VI. CONCLUSION AND FUTURE WORK

In this paper, an effective phishing attacks detection model OFS-NN based on the optimal feature selection and neural network is proposed. In the OFS-NN model, the new FVV index was firstly defined to evaluate the impact of sensitive features on phishing detection. Then, based on the new FVV index, the optimal feature selection algorithm is designed to construct the optimal feature vector for training the underlying neural network classifier. This algorithm could properly deal with problems of big number of phishing sensitive features and the continuous changes of features. Consequently, it can mitigate the over-fitting problem of the neural network classifier. Finally, by repeated experimental analyses, the optimal neural network classifier is trained to detect phishing attacks. As the continuous changing of sensitive features of phishing attacks, in the future, it is necessary to collect more features to perform optimal feature selection.

ACKNOWLEDGMENT

The authors are grateful for Professor Yun Yang from Swinburne University of Technology in Australia for English proofreading.

REFERENCES

[1] APAC. (Dec. 2018). *Fishing Website Processing Bulletin*. Accessed: 2019. [Online]. Available: http://www.apac.cn/gzdt/

[2] M. Khonji, Y. Iraqi, and A. Jones, ''Phishing detection: A literature survey,'' *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2091–2121, 4th Quart., 2013.

[3] A. Acquisti, I. Adjerid, R. Balebako, L. Brandimarte, L. F. Cranor, S. Komanduri, P. G. Leon, N. Sadeh, F. Schaub, M. Sleeper, Y. Wang, and S. Wilson, ''Nudges for privacy and security: Understanding and assisting users' choices online,'' *ACM Computing Surv.*, vol. 50, no. 3, 2017, Art. no. 44.

[4] M. M. Moreno-Fernández, F. Blanco, P. Garaizar, and H. Matute, ''Fishing for phishers. Improving Internet users' sensitivity to visual deception cues to prevent electronic fraud,'' *Comput. Hum. Behav.*, vol. 69, pp. 421–436, Apr. 2017.

[5] M. Junger, L. Montoya, and F.-J. Overink, ''Priming and warnings are not effective to prevent social engineering attacks,'' *Comput. Hum. Behav.*, vol. 66, pp. 75–87, Jan. 2017.

[6] E.-S. M. El-Alfy, ''Detection of phishing websites based on probabilistic neural networks and K-medoids clustering,'' *Comput. J.*, vol. 60, no. 12, pp. 1745–1759, 2017.

[7] C. Huang, S. Hao, L. Invernizzi, Y. Fang, C. Kruegel, and G. Vigna, ''Gossip: Automatically identifying malicious domains from mailing list discussions,'' in *Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS)*, Abu Dhabi, United Arab Emirates, Apr. 2017, pp. 494–505.

[8] F. Vanhoenshoven, G. Nápoles, R. Falcon, K. Vanhoof, and M. Köppen, ''Detecting malicious URLs using machine learning techniques,'' in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–8.

[9] J. Saxe, R. Harang, C. Wild, and H. Sanders, ''A deep learning approach to fast, format-agnostic detection of malicious Web content,'' in *Proc. IEEE Symp. Secur. Privacy Workshops (SPW)*, San Francisco, CA , USA, Aug. 2018, pp. 8–14.

[10] L. Wu, X. Du, and J. Wu, ''Effective defense schemes for phishing attacks on mobile computing platforms,'' *IEEE Trans. Veh. Technol.*, vol. 65, no. 8, pp. 6678–6691, Aug. 2016.

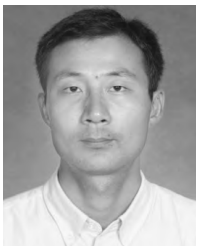
[11] R. Gowtham and I. Krishnamurthi, ''A comprehensive and efficacious architecture for detecting phishing webpages,'' *Comput. Secur.*, vol. 40, pp. 23–37, 2014.

[12] G. Xiang, J. Hong, C. P. Rosé, and L. Cranor, ''CANTINA+: A feature-rich machine learning framework for detecting phishing Web sites,'' *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 2, 2011, Art. no. 21.

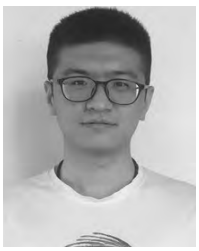
[13] E. Zhu, C. Ye, D. Liu, F. Liu, F. Wang, and X. Li, ''An effective neural network phishing detection model based on optimal feature selection,'' in *Proc. 16th IEEE Int. Symp. Parallel Distrib. Process. Appl. (ISPA)*, Melbourne, NSW, Australia, Dec. 2018, pp. 781–787.

[14] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, ''An empirical analysis of phishing blacklists,'' in *Proc. 6th Conf. Email Anti-Spam (CEAS)*, Mountain View, CA, USA, Jul. 2009, pp. 1–20.

[15] GitHub. *Implementation for the Usage of Google Safe Browsing APIs (v4)*. Accessed: 2019. [Online]. Available: https://github.com/google/safebrowsing
[16] J. Kang and D. Lee, "Advanced white list approach for preventing access to phishing sites," in *Proc. Int. Conf. Converg. Inf. Technol. (ICCIT)*, Gyeongju, South Korea, Nov. 2007, pp. 491–496.
[17] M. Sharifi and S. Siadati, "A phishing sites blacklist generator," in *Proc. 6th ACS/IEEE Int. Conf. Comput. Syst. Appl. (AICCSA)*, Doha, Qatar, Mar./Apr. 2008, pp. 840–843.
[18] X. Han, N. Kheir, and D. Balzarotti, "PhishEye: Live monitoring of sandboxed phishing kits," in *Proc. 23rd ACM Conf. Comput. Commun. Secur. (CCS)*, Vienna, Austria, Oct. 2016, pp. 1402–1413.
[19] L.-H. Lee, K.-C. Lee, H.-H. Chen, and Y.-H. Tseng, "POSTER: Proactive blacklist update for anti-phishing," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Scottsdale, AZ, USA, Nov. 2014, pp. 1448–1450.
[20] A. Aleroud and L. Zhou, "Phishing environments, techniques, and countermeasures: A survey," *Comput. Secur.*, vol. 68, pp. 160–196, Jul. 2017.
[21] L. Shi, D. Lin, C. V. Fang, and Y. Zhai, "A hybrid learning from multi-behavior for malicious domain detection on enterprise network," in *Proc. IEEE 15th Int. Conf. Data Mining Workshops (ICDMW)*, Atlantic City, NJ, USA, Nov. 2015, pp. 987–996.
[22] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: A content-based approach to detecting phishing Web sites," in *Proc. 16th Int. Conf. World Wide Web (WWW)*, Banff, AB, Canada, May 2007, pp. 639–648
[23] X. Li, G. Geng, Z. Yan, Y. Chen, and X. Lee, "Phishing detection based on newly registered domains," in *Proc. IEEE Int. Conf. Big Data (BigData)*, Washington DC, USA, Dec. 2016, pp. 3685–3692.
[24] L. Anh, T. Nguyen, B. L. To, H. K. Nguyen, and M. H. Nguyen, "An efficient approach for phishing detection using single-layer neural network," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, Hanoi, Vietnam, Oct. 2014, pp. 435–440.
[25] S. Marchal, J. FranÃğois, R. State, and T. Engel, "PhishStorm: Detecting phishing with streaming analytics," *IEEE Trans. Netw. Service Manage.*, vol. 11, no. 4, pp. 458–471, Dec. 2014.
[26] R. S. Rao and S. T. Ali, "PhishShield: A desktop application to detect phishing Webpages through heuristic approach," *Procedia Comput. Sci.*, vol. 54, pp. 147–156, Aug. 2015.
[27] G. Sonowal and K. S. Kuppusamy, "PhiDMA—A phishing detection model with multi-filter approach," *J. King Saud Univ.–Comput. Inf. Sci.*, to be published. doi: 10.1016/j.jksuci.2017.07.005.
[28] S. Marchal, G. Armano, T. Gröndahl, K. Saari, N. Singh, and N. Asokan, "Off-the-hook: An efficient and usable client-side phishing prevention application," *IEEE Trans. Comput.*, vol. 66, no. 10, pp. 1717–1733, Oct. 2017.
[29] UCI Machine Learning Repository. *Center for Machine Learning and Intelligent Systems*. Accessed: 2019. [Online]. Available: http://archive.ics.uci.edu/ml/index.php
[30] Phishtank. *Out of the Net, Into the Tank*. Accessed: 2019. [Online]. Available: https://www.phishtank.com/
[31] A. K. Jain and B. B. Gupta, "A novel approach to protect against phishing attacks at client side using auto-updated white-list," *EURASIP J. Inf. Secur.*, vol. 2016, no. 1, Dec. 2016, Art. no. 9.
[32] G. Xiang and J. I. Hong, "A hybrid phish detection approach by identity discovery and keywords retrieval," in *Proc. 18th Int. Conf. World Wide Web (WWW)*, Madrid, Spain, Apr. 2009, pp. 571–580.
[33] G. Varshney, M. Misra, and P. K. Atrey, "A phish detector using lightweight search features," *Comput. Secur.*, vol. 62, pp. 213–228, Sep. 2016.
[34] F. Kausar, B. Al-Otaibi, A. Al-Qadi, and N. Al-Dossari, "Hybrid client side phishing websites detection approach," *Int. J. Adv. Comput. Sci. Appl.*, vol. 5, no. 7, pp. 132–140, 2014.



ERZHOU ZHU received the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2012. He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University, Hefei, China. His current research interests include information security, program analysis, and data mining.



YUYANG CHEN is currently pursuing the master's degree in computer science with Anhui University. His current research interests include information security and machine learning.

CHENGCHENG YE is currently pursuing the master's degree in computer science with Anhui University. His current research interest includes program security.

XUEJUN LI received the Ph.D. degree from Anhui University, Hefei, China, in 2005, where he is currently a Professor with the School of Computer Science and Technology. His research interests include cloud computing and information security.

FENG LIU received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2003. He is currently a Professor with the School of Computer Science and Technology, Anhui University, Hefei, China. His research interests include parallel computing and data mining.