Received: 23 January 2020 | Revised: 26 March 2020 | Accepted: 18 June 2020

DOI: 10.1002/ett.4085

SPECIAL ISSUE ARTICLE

# Ensemble machine learning approaches for webshell detection in Internet of things environments

**Binbin Yong**[1] | **Wei Wei**[2] | **Kuan-Ching Li**[3] | **Jun Shen**[4] | **Qingguo Zhou**[1] | **Marcin Wozniak**[6] | **Dawid Połap**[6] | **Robertas Damaševičius**[7]

[1]School of Information Science and Engineering, Lanzhou University, Lanzhou, China

[2]School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, China

[3]Department of Computer Science and Information Engineering, Providence University, Taichung, Taiwan

[4]School of Computing and Information Technology, University of Wollongong, Wollongong, New South Wales, Australia

[6]Institute of Mathematics, Silesian University of Technology, Gliwice, Poland

[7]Multimedia Engineering Department, Kaunas University of Technology, Kaunas, Lithuania

**Correspondence**
Qingguo Zhou, School of Information Science and Engineering, Lanzhou University, Lanzhou, Gansu, China.
Email:zhouqg@lzu.edu.cn

**Abstract**
The Internet of things (IoT), made up of a massive number of sensor devices interconnected, can be used for data exchange, intelligent identification, and management of interconnected "things." IoT devices are proliferating and playing a crucial role in improving the living quality and living standard of the people. However, the real IoT is more vulnerable to attack by countless cyberattacks from the Internet, which may cause privacy data leakage, data tampering and also cause significant harm to society and individuals. Network security is essential in the IoT system, and Web injection is one of the most severe security problems, especially the webshell. To develop a safe IoT system, in this article, we apply essential machine learning models to detect webshell to build secure solutions for IoT network. Future, ensemble methods including random forest (RF), extremely randomized trees (ET), and Voting are used to improve the performances of these machine learning models. We also discuss webshell detection in lightweight and heavyweight computing scenarios for different IoT environments. Extensive experiments have been conducted on these models to verify the validity of webshell intrusion. Simulation results show that RF and ET are suitable for lightweight IoT scenarios, and Voting method is effective for heavyweight IoT scenarios.

## 1 | INTRODUCTION

Over the past few years, the Internet has made significant progress than it was two decades ago. It is now extensively used in modern life and has spawned the emerging Internet of things (IoT) technology.[1,2] Nowadays, IoT technologies are widely used to monitor and control the mechanical, electrical, and electronic systems used in various types of buildings in home and building automation systems, for instance. According to the analysis in Reference 3, 25 billion IoT devices will appear by the year 2020. Generally, there will be Web servers in the IoT network to provide services for data processing and retrieval for the network.[4,5] However, as the IoT deals with user's personal data and sensitive industrial information, it is crucial to implement robust solutions to protect them from security threats.[6-8] Hackers often utilize the bugs of Web code to break into the servers.[9] In addition, the servers unknowingly render services for intruders to achieve their aims, which are usually termed as webshell. With the increase of IoT scale, webshell is increasingly threatening IoT networks. Moreover, massive data and small computing resources make IoT server difficult to detect webshells effectively.

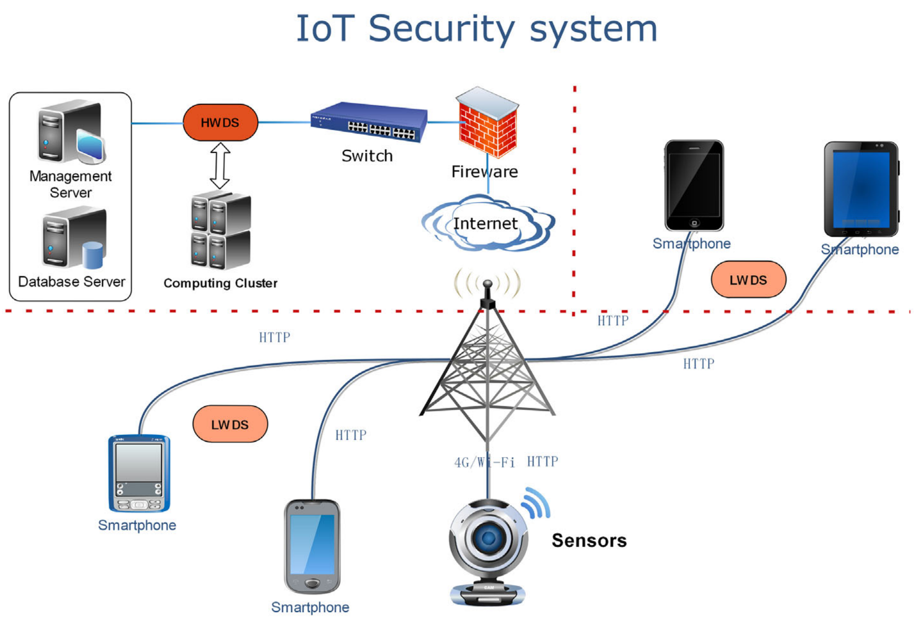


**FIGURE 1** The lightweight (LWDS) and heavyweight (HWDS) webshell detection system for IoT security

On the other side, hypertext preprocessor (PHP) programing language is most commonly employed as Web construction language. Meanwhile, PHP is also a basic server programming language for IoT network and PHP is vulnerable to attack. Hence, the research of PHP based webshell detection is highly significant for IoT security. A basic structure of the IoT network is shown in Figure 1. To ensure network security in this research, we mainly investigate two types of the webshell detection system (WDS) based on PHP, the lightweight WDS (LWDS), and the heavyweight WDS (HWDS). The former is mainly based on traditional machine learning models, which need moderate computing resource and perform poor performance. Thus, LWDS is mainly deployed in routers, smart devices and servers with weaker computing processing power. For the latter with powerful computing capabilities, ensemble machine learning approaches are feasible and can be deployed.

## 1.1 | Contributions

In this article, we make five main research contributions, which are as follows:

(1) A dataset including 1551 malicious PHP webshells and 2593 normal PHP scripts are collected for IoT server security experiments.
(2) We study term frequency inverse document frequency (TFIDF), opcode and combined Opcode-TFIDF feature extraction methods for data preprocessing.
(3) Feature clustering analysis based on principal component analysis (PCA) is performed to analyze the dataset.
(4) We study the traditional machine learning models and their ensemble models for LWDS IoT scenarios. Extensive experiments are conducted to compare the performances of different models. The best model for this scenario is given.
(5) Feature importances for webshell detection are evaluated, and top-10 relevant opcodes to identify webshells are ranked.

The above contributions will support empirical IoT deployments to detect and avoid webshell attacks, which are threatening IoT security.

## 1.2 | Organization

The rest of the article is organized as follows. Section 2 presents related work in IoT security and malicious Web detection. The feature extraction methods and detection models for IoT webshells detection are shown in Section 3, where the traditional machine learning models for Web security detection are discussed. Meanwhile, the architecture and the implementation details of ensemble models are presented. Section 4 depicts the experimental results and analysis, and finally, conclusions and future work are addressed in Section 5.

## 2 | RELATED WORK

In order to enhance the security of IoT network, several related work on security has already been conducted. Stergiou et al[10] presented a survey of IoT and cloud computing with a focus on the security issues of both technologies, and they showed that cloud computing technology could improve the security of the IoT. Huang et al[11] attempted to design a security framework for body IoT, home IoT and hotel IoT scenarios. Mathur et al[12] proposed a IoT solution to guarantee data and network security of wireless devices. Suárez-Albela et al[13] studied the security evaluation of IoT gateways in resource-constrained. Recently, Qiang et al presented a survey on Web security,[14] in which some machine learning-based defensive techniques are detailed introduced. With the development of machine learning, it is being applied to malicious Web activity detection in IoT environments, as IoT generates a vast amount of heterogeneous data. Abubakar et al[15] proposed a cyber security framework to protect the IoT based integrated internet-based smart grid from being attacked. Jiankang et al[16] utilized a decision tree (DT) algorithm to detect webshell. Based on the optimal threshold values, Tu et al studied a novel method to detect malicious Web codes.[17] Azmoodeh et al[18] proposed an approach for IoT malware detection via the device's operational code (OpCode) sequence and achieved excellent results. Recently, Brun et al[19] presented a deep learning methodology to detect network attacks online against IoT gateways. Nowadays, DNN[20] is widely applied in many fields,[21,22] and such DNN applications assist people to get rid of tedious recognition works. Hence, DNN is also a promising approach for malicious activities detection in IoT network.[23] However, IoT security has only been extensively studied recently, and there is a lack of a holistic comparative study based on the popular machine learning and DNN approaches consume vast amount of resources, which is difficult in IoT environments. Therefore, we carry out this research to support reference models in the field of IoT security.

## 3 | METHODS

In this section, we first introduce the feature extraction methods. Then, we discuss the machine learning models for IoT webshells detection. Dataset and training method are also presented.

### 3.1 | Feature extraction

Word of bag model (WOG) is a commonly used method in text data preprocessing,[24] which can be used to extract features for text representation. One PHP script file is a text character set, which is suitable for WOG modeling. TFIDF is another frequently used feature extraction method, used to further string data processing. Moreover, the combination of TFIDF and 2-Gram[25] WOG is a standard preprocessing method to improve the model accuracy that is also adapted in this article. PHP script is executed on Zend,[26] which is designed as a type of virtual machine for PHP code in the proposed IoT system. When running a PHP application, the code is transformed into opcode, which can be run on Zend. Therefore, opcode expresses the same instructions as the original PHP script. To extract the PHP opcode, Vulcan logic dumper tool is used. In our design, the combined Opcode-TFIDF preprocessing method is used for PHP webshell detection. Particularly, the opcode method is able to extract the detailed instruction operations of PHP scripts, and TFIDF method is effective to find the internal characteristics of these webshells. The combined Opcode-TFIDF method can utilize these two advantages to preprocess the data. Therefore, in the experiments that follow next, we will adopt Opcode-TFIDF preprocessing method as the default method.

### 3.2 | Machine learning models

Realistically, the basic webshell detection in IoT could be implemented by traditional machine learning methods to classify the normal and malicious PHP scripts by the extracted text features. Traditional machine learning models includes

K-Means, k-nearest neighbor (KNN),[27] multilayer perceptron (MLP), support vector machine (SVM),[28] naive Bayes (NB), DT,[29] and so on. As known, K-Means and KNN are both classical clustering algorithms, and there are large number of Web-based analysis conducted using these algorithms, such as References 30,31. However, only a few references related to webshell detection using these two clustering methods are found. MLP is a type of simple neural network, which is usually trained by back propagation algorithm (BP). Wu and Tsai[32] applied MLP to classify spam e-mails. Chang et al tried to detect the intrusion with MLP,[33] and they showed that both the performance and the overall execution efficiency are effected by features and samples. Stevanovic et al[34] used MLP into malware detection. SVM is often used for security detection, such as Reference 35. NB is especially effective for binary classification problems, such as webshell detection. Gao et al[36] designed a NB model to avoid information leakage, and Sayamber and Dixit utilized NB to detect malicious URL automatically.[37] DT, which utilizes the idea that divides the dataset into smaller datasets based on the descriptive features and tree-like graph until a small enough set that contains data points falling under one label is reached, has been applied in security field.[16] These approaches consume a small amount of resources, which is suitable for the LWDS scenario. However, these models may lack accuracy for IoT webshell detection. Therefore, we try to ensemble these models to achieve robust and accurate ensembles for HWDS scenario.

### 3.3 | Ensemble models

Two families of ensemble methods are usually used to ensemble classifiers, which are averaging methods and boosting methods, respectively. For averaging methods, some base classifiers are trained independently, and the ensemble model is designed to average the predictions of these base classifiers. The conventional averaging methods include Bagging, random forest (RF), and others. For boosting methods, base classifiers are built sequentially and then it is tried to reduce the bias of the combined model. Boosting methods aim to combine some weak models to produce a powerful ensemble. RF[38] is a standard ensemble method that includes many DTs. Alam and Vuong et al have applied RF to detect Android malware,[39] and Chihab et al developed methods to detect Internet intrusion based on NB and RF.[40] Extremely randomized trees (ET)[41] is also an ensemble learning method similar to RF, on the other hand.

In this article, the machine learning models perform relatively satisfactorily for webshell detection, as shown next. These models can be seen as reliable classifiers, and hence, averaging methods are mainly used to ensemble these models. As shown in Figure 2, we first train six types of machine learning models, which are K-Means, MLP, NB, DT, SVM, and KNN. Then, these models are combined by voting. Two types of ensembling methods are used in this article. The first method is shown as Figure 2A, for each type model, we train 20 classification estimators and ensemble them by voting. For the second method, as shown in Figure 2B, six types of models are trained as six estimators, and these estimators are also ensembled by voting. That is, we will calculate the classification probabilities of all models, and take their average probabilities as the final probabilities of classification.

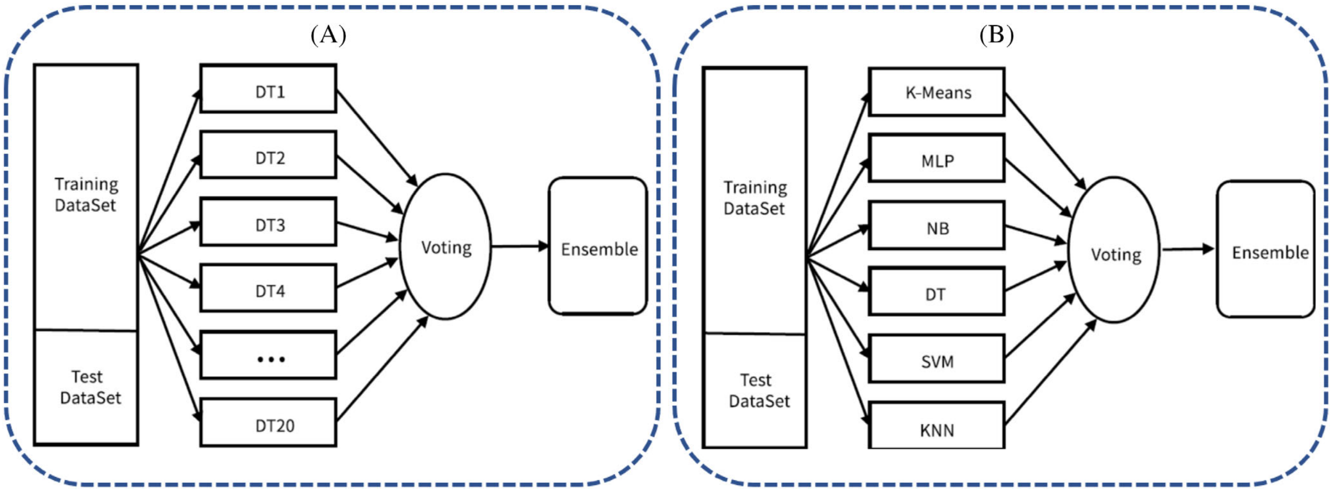
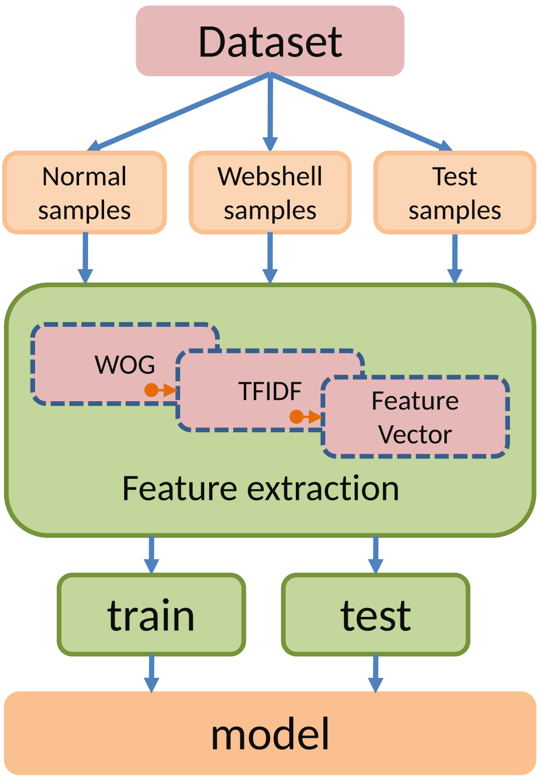


**FIGURE 2** The ensemble model for webshell detection

**FIGURE 3** The flowchart of model training



## 3.4 | Dataset and training method

In this research, malicious IoT webshells are detected by PHP script files. We have gathered many positive and negative PHP scripts from many websites as samples since PHP is the most widely used Web programming language. We collected the normal PHP samples from public Web sites providing regular services, and the webshells are mainly gathered from Web security sites. Finally, 1551 malicious PHP scripts and 2593 normal PHP scripts are collected for the test in this article. Hence, we obtained 4144 PHP scripts for experiments.

The flowchart used to train and test machine learning models is shown in Figure 3. First, the dataset is split into training samples and test samples, which consist of normal PHP scripts and webshell scripts. Then, these samples are preprocessed and represented by WOG model. Next, TFIDF and opcode methods are used to extract the features of the samples, which have different word counts. Then, 100 features are extracted as input feature vector of these detection models, and the categories are used as the output of these models. In our experiments,80% of samples are randomly chosen as training samples, and the remaining 20% of samples are used as test samples.

## 4 | EXPERIMENTS AND ANALYSIS

In this section, we first explain the dataset and the measure metrics, followed next with the presentation of the experimental results for machine learning models and ensemble models.

## 4.1 | Measure metrics

In this section, we depict the experimental results and analysis of machine learning models and ensemble models. We use four metrics, which include Accuracy, Precision, Recall, and F1 score to evaluate the performances of these models as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}, \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{3}$$

$$F1 = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}. \tag{4}$$

These four metrics are frequently used in security analysis and based on four basic metrics which are TP (true positive), FP (false positive), TN (true negative), and FN (false negative). They can give an objective evaluation of these detection models.

### 4.2 | Model parameters

In our experiments, K-Means, MLP, NB, DT, SVM, and KNN are tested to get better model parameters. For K-Means, the number of clusters is set as 2 since it is a binary classification algorithm. Meanwhile, the number of initialization parameter is set as 10, which means the algorithm attempted 10 times initialization and find the best one. We have experimentally verified the validity of these parameters. For MLP, the input dimension is 100, which represents 100 features extracted for each sample. The MLP model is designed with two hidden layers, which have 30 and 10 hidden nodes, respectively, and it is trained by BP algorithm. In fact, we got the best number of hidden nodes by many experiments. In NB and DT, there are few important parameters. We limit the maximum depth of DT as 10 to prevent the model from overfitting. In the SVM model, the kernel type is selected as the radial basis function (RBF), and grid search method is used to find best parameters of RBF function. In KNN algorithm, the number of neighbors is set as 20, which is proved effective because the number of samples is small. These parameters are set as the default parameters in the following experiments.

### 4.3 | Detection results

In this subsection, we will compare the webshell detection results between machine learning models and ensemble models based on the mixed Opcode-TFIDF preprocessing methods.

#### 4.3.1 | Nonensemble methods

We use the combined Opcode-TFIDF preprocessing methods to preprocess the PHP scripts, which are first converted to opcode scripts. Then, based on the TFIDF method, the opcode scripts are converted to training and test samples. Based on the Opcode-TFIDF preprocessing method, six types of machine learning models are trained, and the test results are shown in Table 1.

**TABLE 1** Experimental results: Detection results of machine learning models based on Opcode-TFIDF preprocessing method

| models | Accuracy (%) | Precision (%) | Recall (%) | F1 (%) |
|---|---|---|---|---|
| K-Means | 76.79 | 71.40 | 86.29 | 78.14 |
| MLP | 91.82 | 91.21 | 92.75 | 91.97 |
| NB | 85.36 | 93.58 | 76.25 | 84.03 |
| DT | **94.86** | 94.17 | 95.08 | **94.62** |
| SVM | 81.85 | **99.54** | 62.96 | 77.13 |
| KNN | 74.75 | 65.74 | **97.68** | 78.59 |

*Note*: The **boldfaces** are the best results.
Abbreviations: DT, decision tree; KNN, k-nearest neighbor; MLP, multilayer perceptron; NB, naive Bayes; SVM, support vector machine; TFIDF, term frequency inverse document frequency.

From this table, we can see that KNN achieved the highest Recall of 97.68% yet achieved the lowest Accuracy of 74.75% and lowest Precision of 65.74%. DT achieves the best detection results, and the metrics of Accuracy, Precision, Recall, and F1 are all greater than 94%. It also achieves the highest F1 score of 94.62%. From these observations, we can conclude that DT is suitable for LWDS scenario.

### 4.3.2 | Ensemble methods

In this section, we will test the detection effect of ensemble methods. We first test the bagging method to improve single models. As shown in Figure 2, 20 base estimators are first trained for each model. Then, these similar models are combined by voting, and the webshells detection results are shown in Table 2.

We can see that, all machine learning models are improved by ensembling single type models, compared with Table 1. For the DT model, it reaches a Recall score of 98.30% and F1 of 97.07%. While MLP reaches the highest Recall of 98.59% and highest F1 of 97.60%. For the SVM model, it improves its Recall score from 62.96% to 96.48%. The increase rate of Recall reaches 53.24%. In addition, the ensemble models are more balanced in four metrics. It indicates that the ensemble single type models are useful for these traditional models to improve the detection results.

With the PCA method,[42-46] we select two components that contain the main information of the samples. Based on these two components, classification boundaries for these ensemble methods are plotted in Figure 4. The black dots represent the actual webshell samples and the gray shaded areas represent the webshell classification region of models. On the contrary, the red dots represent the normal PHP samples and the light red shaded areas represent the normal samples classification region of models. It can be seen that K-Means (Figure 4A) and KNN (Figure 4F) both have complex classification regions, representing overfitting. In addition, in Table 2, K-Means and KNN have the lowest F1 scores representing the worst detection results. NB (Figure 4C) gives a straightforward classification region, corresponding to general forecasting results in Table 2. SVM (Figure 4E) performs poorly in the mixed regions of webshell samples and normal samples, and it also achieves a general F1 score in Table 2. MLP (Figure 4B) and DT (Figure 4D) have slightly better classification boundaries, and they obtain the highest F1 scores. RF (Figure 4G) and Voting (Figure 4H) models integrate advantages of single models with better classification boundaries in the fixed regions of webshell samples and normal samples. Actually, there are many scenarios that two types of samples are mixed together. In this case, two components are helpless to separate them. Hence, we need more components and features to separate the samples better.

Furthermore, we combine six types of machine learning models to get an ensemble model, which is named as "Voting." In the experiments, all models have the same voting weights, and the Voting ensemble model averages the classification probabilities of these six models via the voting method. Intuitively, it can integrate all the advantages of these models. Meanwhile, ensemble RF and ET models are also tested for comparison. The detection results are shown in Table 3, and we can see that ensemble models RF, ET, and Voting all achieve good detection results. The voting ensemble model is better than RF and ET models according to the Recall and F1 metrics, which are the highest Recall of 99.57% and highest F1 of 98.32%. It is noted that the metrics are all larger than 97%, which is better than the results of single type ensemble models in Table 2. In other words, the ensemble model of different type models outperforms the ensemble model of same models for webshell detection. Therefore, the ensemble of different type models is more effective for webshell detection in IoT network.

**TABLE 2** Experimental results: Detection results of single type model ensemble

| models | Accuracy (%) | Precision (%) | Recall (%) | F1 (%) |
|---|---|---|---|---|
| K-Means | 78.55 | 73.26 | 87.07 | 79.57 |
| MLP | **97.56** | 96.64 | **98.59** | **97.60** |
| NB | 89.63 | **97.32** | 75.36 | 84.94 |
| DT | 97.15 | 95.86 | 98.30 | 97.07 |
| SVM | 88.74 | 79.11 | 96.48 | 86.94 |
| KNN | 78.76 | 70.32 | 96.39 | 81.32 |

*Note*: The **boldfaces** are the best results.

Abbreviations: DT, decision tree; KNN, k-nearest neighbor; MLP, multilayer perceptron; NB, naive Bayes; SVM, support vector machine.

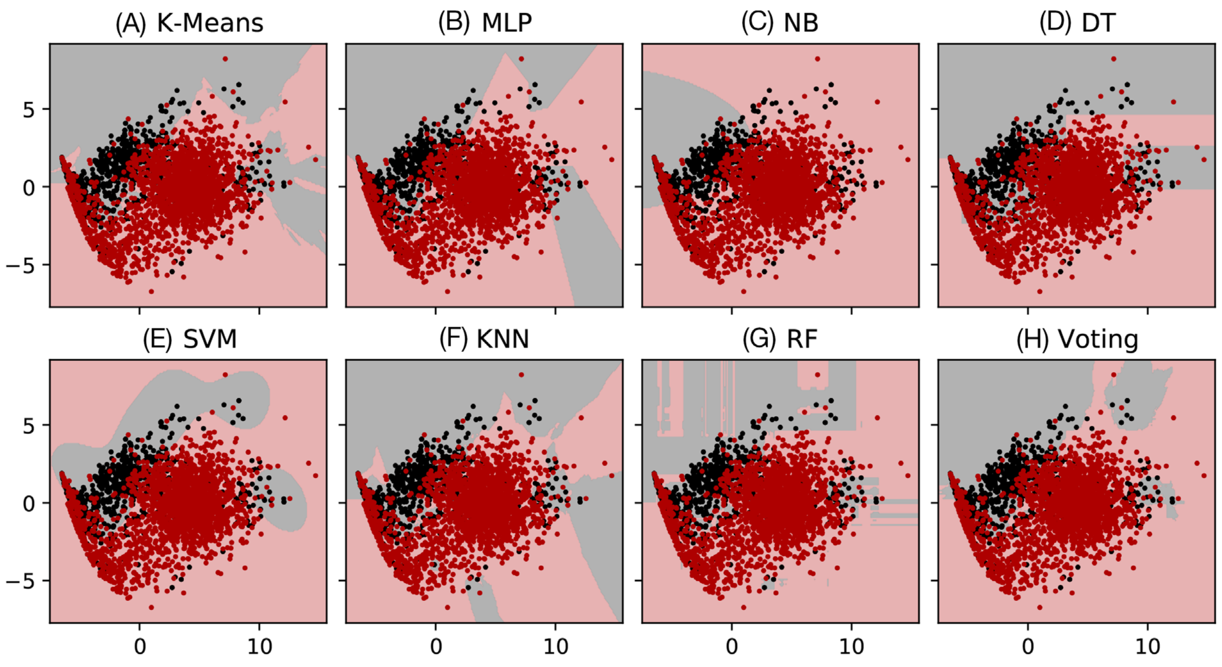


**FIGURE 4** Classification boundaries for single type ensemble methods based on two components by PCA. PCA, principal component analysis

**TABLE 3** Experimental results: Detection results of different type ensemble models

| models | Accuracy (%) | Precision (%) | Recall (%) | F1 (%) | time (ms) |
|---|---|---|---|---|---|
| RF | 97.94 | **97.99** | 97.84 | 97.92 | **15.874** |
| ET | 98.06 | 97.48 | 98.51 | 97.99 | 92.028 |
| Voting | **98.37** | 97.10 | **99.57** | **98.32** | 1306.9 |

*Note*: The **boldfaces** are the best results.
Abbreviations: ET, extremely randomized trees; RF, random forest.

In order to observe the detection effect more intuitively, the receiver operating characteristic (ROC) curves of these three types of ensemble models are drawn in Figure 5. In this figure, we can see that the areas under ROC for RF, ET, and Voting ensemble models reach 0.99, 0.99, and 1.00. This illustrates that these ensemble models are excellent in detecting webshells.

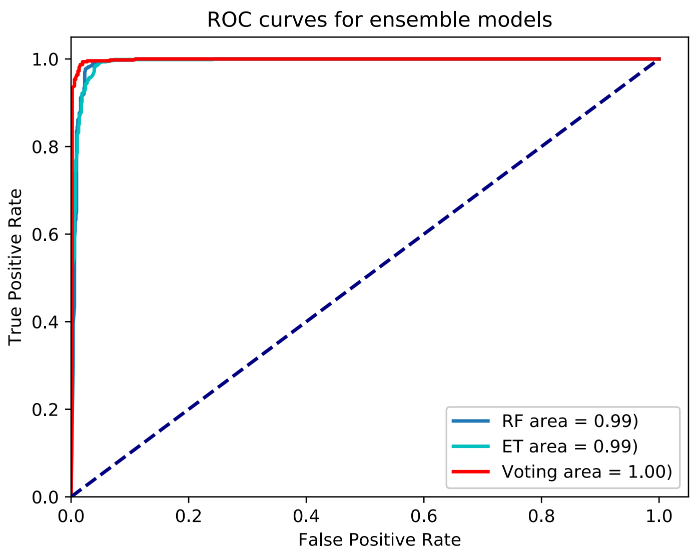


**FIGURE 5** The ROC curves of ensemble models. ROC, receiver operating characteristic

## 4.4 | Features importances analysis

Based on RF and ET methods, we use forests of trees to evaluate the importances of features for webshell classification task. The top-10 importance of features and their opcodes are shown in Table 4 and Figure 6.

In Figure 6A,C give the importances sorted by RF and ET, and Figure 6B,D show the heatmaps of this importance. We can see that only the first 10 or so features have bright colors, representing greater importances. In addition, we can see that features 3, 4, 5, and 6 have the most important values for both RF and ET models, which correspond to ECHO, _FCALL, INIT, and RETURN opcodes according to Table 4. In fact, these opcodes often appear in the scripts that invoke a new function to implement certain functions. This is similar behavior as webshells. However, the sum of importances for these features are 0.434 and 0.402, respectively. In other words, the models cannot detect webshells correctly only by these

**TABLE 4** Experimental results: Important features and opcodes for RF and ET

| RF | | | ET | | |
|---|---|---|---|---|---|
| NO. | Importance | Opcode | NO. | Importance | Opcode |
| 3 | 0.139 | ECHO | 6 | 0.142 | _FCALL |
| 6 | 0.117 | _FCALL | 4 | 0.106 | RETURN |
| 5 | 0.094 | INIT | 3 | 0.104 | ECHO |
| 4 | 0.084 | RETURN | 5 | 0.050 | INIT |
| 7 | 0.070 | SEND | 8 | 0.040 | _VAL |
| 10 | 0.053 | BEGIN | 13 | 0.033 | ASSIGN |
| 8 | 0.043 | _VAL | 12 | 0.031 | END |
| 9 | 0.037 | DO | 11 | 0.028 | _SILENCE |
| 11 | 0.035 | _SILENCE | 10 | 0.026 | BEGIN |
| 13 | 0.035 | ASSIGN | 7 | 0.026 | SEND |

Abbreviations: ET, extremely randomized trees; RF, random forest.

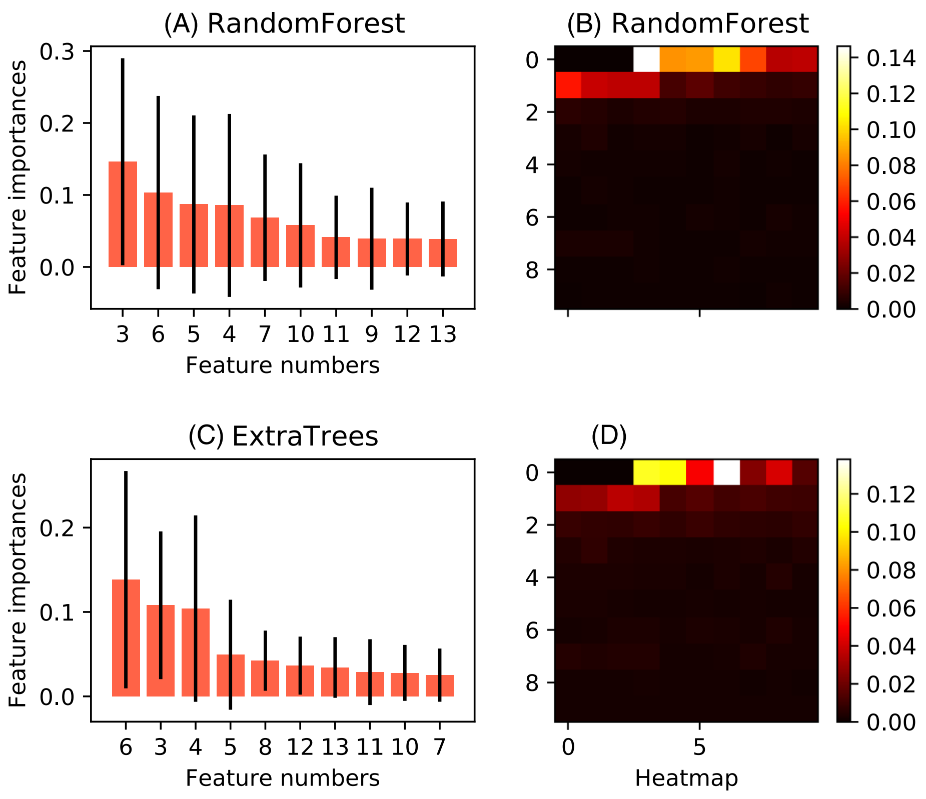


**FIGURE 6** The feature importances and heatmaps for RF and ET. ET, extremely randomized trees; RF, random forest

features. In addition, the order of top-10 importance of features is different for RF and ET, which indicates that different models tend to choose different features for classification. Therefore, we combine different models to get an ensemble to integrate the advantages of multiple models. Our results prove that the ensemble webshell detection model is effective for webshell detection.

## 5 | CONCLUSIONS

With the rapid development of IoT technology, applications based on IoT are widely applied in IT infrastructure.[43,44] Meanwhile, the security of IoT network is becoming more and more critical. In this article, we proposed the LWDS and the HWDS for lightweight and heavyweight IoT network security detection. Based on machine learning models, we have presented specific solutions for these two scenarios. In order to detect webshells more accurately, ensemble methods based on traditional machine learning models are used to improve the performance of detection models. Based on ensemble models, we analyzed the features of the samples and acquired important features of opcodes for distinguishing webshells, and top-10 important features were also extracted to show the key opcodes in webshells. The experiment results show that, the proposed ensemble models could significantly improve the malicious webshell detection results in IoT, compared with the traditional machine learning models. RF and ET ensembles are more suitable for lightweight LWDS scenario for their efficiency. Although it requires more substantial computing resources and longer computing time, the Voting method achieves the maximum Recall score of 99.57% and maximum F1 score of 98.32%. Therefore, it is suitable for IoT servers in HWDS scenario with reliable computing power. We believe that the experimental results are significant for the other IoT security researchers. However, IoT servers could be built in other programming languages, and we only test the machine learning models for webshell detection on PHP scripts. In the future, more types of webshell scripts need to be studied though the underlying methods may be similar.

### ACKNOWLEDGMENTS

This work was partially supported by National Natural Science Foundation of China under Grant No. 61402210, The Fundamental Research Funds for the Central Universities under Grant No. lzujbky-2019-kb51 and lzujbky-2018-k12, Ministry of Education - China Mobile Research Foundation under Grant No. MCM20170206, Major National Project of High Resolution Earth Observation System under Grant No. 30-Y20A34-9010-15/17, State Grid Corporation Science and Technology Project under Grant No. SGGSKY00FJJS1800403 and No.522722160071, Program for New Century Excellent Talents in University under Grant No. NCET-12-0250, and Double first class Funding-International Cooperation and Exchange Program under Grant No. 227000-560001, and Strategic Priority Research Program of the Chinese Academy of Sciences with Grant No. XDA03030100. Google Research Awards and Google Faculty Award. Prof Jun Shen is partially supported by UOW's UGPN RCF 2018-2019, NSF of China 61872079 and UOW's UIC international exchange and sabbatical leave program supporting his visit at Lanzhou University and MIT. This work is also supported by the Key Research and Development Program of Shaanxi Province (No.2018ZDXM-GY-036) and Shaanxi Key Laboratory of Intelligent Processing for Big Energy Data (No.IPBED7).

## CONFLICT OF INTEREST

The authors declare no potential conflict of interests.

### ORCID

*Qingguo Zhou* https://orcid.org/0000-0001-8054-5446
*Dawid Połap* https://orcid.org/0000-0003-1972-5979
*Robertas Damaševičius* https://orcid.org/0000-0001-9990-1084

### REFERENCES

1. Zhou X, Liang W, Huang S, Fu M. Social recommendation with large-scale group decision making for cyber-enabled online service. *IEEE Trans Comput Soc Syst*. 2019;6(5):1073-1082. https://doi.org/10.1109/TCSS.2019.2932288.
2. Liang W, Zhou X, Wang KSS. Multi-modality behavioral influence analysis for personalized recommendations in health social media environment. *IEEE Trans Comput Soc Syst*. 2019;6(5):888-897. https://doi.org/10.1109/TCSS.2019.2918285.

3. Muhammad F, Anjum W, Mazhar KS. A critical analysis on the security concerns of Internet of Things (IoT). *Int J Comput Appl*. 2015;111(7):1-6.
4. Zhou X, Liang W, Wang K, Huang R, Jin Q. Academic influence aware and multidimensional network analysis for research collaboration navigation based on scholarly big data. *IEEE Trans Emerg Topics Comput*. 2018. https://doi.org/10.1109/TETC.2018.2860051.
5. Wu B, Zhou X, Jin Q. Analysis of user network and correlation for community discovery based on topic-aware similarity and behavioral influence. *IEEE Trans Human-Mach Syst*. 2018;48(6):559-571. https://doi.org/10.1109/THMS.2017.2725341.
6. Shui Y. Big privacy: challenges and opportunities of privacy study in the age of big data. *IEEE Access*. 2016;4:2751-2763. https://doi.org/10.1109/ACCESS.2016.2577036.
7. Wei W, Fan X, Wozniak M, et al. Control of network control system for singular plant. *Inf Technol Control*. 2018;20(1):39-48.
8. Wei W, Marcin W, Robertas D, Fan X, Li Y. Algorithm research of known-plaintext attack on double random phase mask based on WSNs. *J Internet Technol*. 2019;47(1):140-150.
9. Shui Y, Guojun W, Wanlei Z. Modeling malicious activities in cyber space. *IEEE Netw*. 2015;29(6):83-87. https://doi.org/10.1109/MNET.2015.7340429.
10. Stergiou C, Psannis KE, Kim BG, Gupta B. Secure integration of IoT and Cloud Computing. *Futur Gener Comput Syst*. 2016;78(3):964-975.
11. Huang X, Craig P, Lin H, Yan Z. SecIoT: a security framework for the Internet of Things. *Secur Commun Netw*. 2016;9(16):3083-3094.
12. Mathur A, Newe T, Elgenaidi W, Rao M, Dooly G, Toal D. A secure End-to-End IoT solution. *Sensor Actuat A Phys*. 2017;263(C):291-299.
13. Suárez-Albela M, Fernández-Caramés TM, Fraga-Lamas P, Castedo L. A practical evaluation of a high-security energy-efficient gateway for IoT fog computing applications. *Sensors*. 2017;17(9):1978-2017.
14. Qiang L, Pan L, Wentao Z, Wei C, Shui Y VCML, A survey on security threats and defensive techniques of machine learning: a data driven view. *IEEE Access* 2018; 6: 12103-12117. https://doi.org/10.1109/ACCESS.2018.2805680
15. Abubakar SS, Dong Y, Jiong J, Longxiang G, Shui Y, ZhaoYang D. Cyber security framework for Internet of Things-based energy internet. *Futur Gener Comput Syst*. 2019;93(1):849-859.
16. Jiankang HU, Zhen XU, Duohe MA, Yang J. Research of webshell detection based on decision tree. *J Netw New Media*. 2012;1(6):15-19.
17. Tu TD, Cheng G, Guo X, Pan W. Webshell detection techniques in web applications. Paper presented at: Proceedings of the 5th International Conference on Computing Communication and Networking Technologies, United States; vol 1, 2014:1-7.
18. Azmoodeh A, Dehghantanha A, Choo KKR. Robust malware detection for internet of (Battlefield) things devices using deep eigenspace learning. *IEEE Trans Sustain Comput*. 2018;99:1-9.
19. Brun O, Yin Y, Gelenbe E, Kadioglu YM, Augusto-Gonzalez J, Ramos M. Deep learning with dense random neural networks for detecting attacks against IoT-connected home environments. Paper presented at: Proceedings of the International ISCIS Security Workshop, London, UK; 2018:79-89.
20. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521:436-444.
21. Zhang Q, Yang LT, Chen Z. Deep computation model for unsupervised feature learning on big data. *IEEE Trans Serv Comput*. 2016;9(1):161-171.
22. Liu C, Cao Y, Luo Y, et al. A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure. *IEEE Trans Serv Comput*. 2018;11(2):249-261.
23. Yong B, Liu X, Liu Y, Yin H, Huang L, Zhou Q. Web behavior detection based on deep neural network. Paper presented at: Proceedings of the 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI 2018), Guangzhou, China; October 8, 2018:1911-1916. https://doi.org/10.1109/SmartWorld.2018.00320
24. Hong TP, Lin CW, Yang KT, Wang SL. Using TF-IDF to hide sensitive itemsets. *Appl Intell*. 2013;38(4):502-510.
25. Tripathy A, Agrawal A, Rath SK. Classification of sentiment reviews using n-gram machine learning approach. *Expert Syst Appl*. 2016;57:117-126.
26. Lyman F. *Beginning Zend Framework*. Berkeley, California, USA: Apress; 2013.
27. Miao X, Gao Y, Chen G, Zheng B, Cui H. Processing incomplete k nearest neighbor search. *IEEE Trans Fuzzy Syst*. 2016;24(99):1349-1363.
28. Chen D, Tian Y, Liu X. Structural nonparallel support vector machine for pattern recognition. *Pattern Recogn*. 2016;60:296-305.
29. Kim K. A hybrid classification algorithm by subspace partitioning through semi-supervised decision tree. *Pattern Recogn*. 2016;60:157-163.
30. Ling H, Wang H, Management SO. Integration of bacterial foraging with K-means for Web user session clustering. *Comput Eng Appl*. 2012;48(36):121-124.
31. Alam M, Sadaf K. Web search result clustering based on heuristic search and k-means. *Comput Sci*. 2016;81(2):96-116.
32. Wu CH, Tsai CH. Robust classification for spam filtering by back-propagation neural networks using behavior-based features. *Appl Intell*. 2009;31(2):107-121.
33. Chang RI, Lai LB, Su WD, Wang JC, Kouh JS. Intrusion detection by backpropagation neural networks with sample-query and attribute-query. *Int J Comput Intell Res*. 2008;3(1):6-10.
34. Stevanovic D, Vlajic N, An A. Detection of malicious and non-malicious website visitors using unsupervised neural network learning. *Appl Soft Comput J*. 2013;13(1):698-708.
35. Choi J, Kim H, Chang C, Kim P. Efficient malicious code detection using N-gram analysis and SVM. Paper presented at: Proceedings of the International Conference on Network-Based Information Systems, Tirana, Albania; 2011:618-621.
36. Gao CZ, Cheng Q, He P, Susilo W, Li J. Privacy-preserving naive Bayes classifiers secure against the substitution-then-comparison attack. *Inf Sci*. 2018;444:72-88.
37. Sayamber AB, Dixit AM. Malicious URL detection and identification. *Int J Comput Appl*. 2014;99(17):17-23.

38. Paul A, Mukherjee DP, Das P, Chintha AR, Gangopadhyay A, Kundu S. Improved random forest for classification. *IEEE Trans Image Process*. 2018;27(8):4012-4024.
39. Alam MS, Vuong ST. Random forest classification for detecting android malware. *IEEE IoT IEEE Cyber Phys Soc Comput*. 2013;663-669.10. 1109/GreenCom-iThings-CPSCom.2013.122.
40. Chihab Y, Ouhman AA, Erritali M, Ouahidi BE. Detection and classification of internet intrusion based on the combination of random forest and Naı¨ve Bayes. *Int J Eng Technol*. 2013;5(3):2116-2126.
41. Pinto A, Pereira S, Rasteiro D, Silva CA. Hierarchical brain Tumour segmentation using extremely randomized trees. *Pattern Recogn*. 2018;82:105-117.
42. Rizvi S, Mohammadpour J, Toth R, Meskin N. A kernel-based pca approach to model reduction of linear parameter-varying systems. *IEEE Trans Control Syst Technol*. 2016;24(5):1883-1891.
43. Qi Y. Information potential fields navigation in wireless Ad-Hoc sensor networks. *Sensors*. 2011;11(5):4794-4807.
44. Song H, Li W, Shen P, Vasilakos A. Gradient-driven parking navigation using a continuous information potential field based on wireless sensor network. *Inf Sci*. 2017;408(C):100-114. https://doi.org/10.1016/j.ins.2017.04.042.
45. Xu Q, Wang L, Hei X, Shen P, Shi W, Shan L. GI/Geom/1 queue based on communication model for mesh networks. *Int J Commun Syst*. 2014;27(11):3013-3029.
46. Chen G, Li C, Wei W, et al. Big data analytics enabled by feature extraction based on partial independence. *Neurocomputing*. 2017;288:3-10. https://doi.org/10.1016/j.neucom.2017.07.072.

**How to cite this article:** Yong B, Wei W, Li K-C, et al. Ensemble machine learning approaches for webshell detection in Internet of things environments. *Trans Emerging Tel Tech*. 2020;e4085. https://doi.org/10.1002/ett.4085