[1]Qi Wu
[2,*]Shuo Wen
[3]Fangxiong Li
[4]Boliang Liu
[5]Wei Zhong

Web Attack Detection Based on Honeypots and Logistic Regression Algorithm

Abstract: - Web security has emerged as one of the most prominent concerns in the realm of cybersecurity. Traditional rule-based methods for detecting web attacks often rely on manual rule definition and pattern matching, leaving them inadequate in accurately identifying new and intricate attack patterns. In the face of these challenges, machine learning techniques have demonstrated potential and advantages. This paper presents a web attack detection method based on honeypots and the logistic regression algorithm. It involves the cleansing, filtering, and analysis of web logs captured by honeypots, followed by the vectorization of the textual data contained in these logs. The logistic regression algorithm is employed to train and test the classification of the text vectors, generating a logistic regression model. This model is then used to predict newly generated web logs, enabling effective dynamic web attack detection. Experimental evaluations using collected datasets are conducted, comparing the proposed method with the support vector machine approach. The results demonstrate that this method achieves rapid and accurate detection and recognition of web attack behaviors while ensuring performance efficacy.

Keywords: Honeypots, TF-IDF, Logistic Regression, Feature Matrix.

I. INTRODUCTION

With the rapid advancement of the Internet, the construction of network security has entered a new era. Web applications have found widespread application in various aspects of life, including education, research, and the economy. However, it is unfortunate that many developers lack security awareness when crafting web applications. They may prioritize application functionality at the expense of security considerations, resulting in the presence of numerous vulnerabilities within web applications. Consequently, web applications often become prime targets for attackers seeking to invade and disrupt, thus leading to a significant increase in various types of web security concerns, becoming one of the most prominent issues within the realm of cybersecurity.

The harms and disruptive nature of web attacks are indeed significant, severely impacting users' normal utilization of the Internet. Web attacks are intricate and constantly evolving, gradually becoming more covert, persistent, and industrialized. Simultaneously, the application of detection technologies to combat web application attacks presents significant challenges for researchers in the field of security. Various types of web attacks continue to emerge, which pose threats to user privacy and data integrity. In order to address these risks, web attack detection systems have become increasingly vital. Traditional rule-based detection methods often require manual rule definition and pattern matching, leaving them inadequate in accurately identifying new and intricate attack patterns. Machine learning techniques have shown immense potential and advantages in the realm of web attack detection.

This study primarily focuses on deploying honeypots for capturing web logs, and based on this, proposes a web attack detection method that relies on honeypot log analysis and logistic regression algorithm. The method involves the collection, storage, filtering, and cleansing of honeypot log data. Furthermore, effective logistic regression algorithms are employed to analyze the honeypot log data. By analyzing the features of web requests in the logs, these requests are classified as normal or malicious attacks, thus achieving web attack behavior detection. This approach enables automatic learning and identification of malicious behavior within web requests, enhancing the accuracy and efficiency of detection. The web attack detection system can dynamically monitor and analyze incoming web requests to identify and prevent malicious attack behaviors. It provides a foundation for subsequent attack tracing, defense, protection, and network threat assessment.

[1] Guangdong Police College, Guangzhou, Guangdong, China
[2] Guangdong Police College, Guangzhou, Guangdong, China
[3] Guangdong Police College, Guangzhou, Guangdong, China
[4] Guangdong Police College, Guangzhou, Guangdong, China
[5] Guangdong Police College, Guangzhou, Guangdong, China
*Corresponding author: Shuo Wen

II. RELATED WORK

In previous research, scholars have conducted extensive studies on web attack detection. Two main approaches can be identified: rule-based and machine learning-based detection [1]. Mahdavi et al. [2] employed static analysis of application code to generate legitimate SQL statement templates. They then matched all SQL statements against these predefined templates to identify whether they were legitimate SQL statements generated by the application itself. ABIKOYE et al. [3] utilized KMP string matching algorithm to detect and prevent attacks. JANA et al. [4] proposed a code-based SQL injection analysis method that assigns complex numbers to each input element to analyze user inputs before submitting a query to the underlying database. They designed a code checker based on assertion techniques to identify suspicious inputs. SQLBlock [5] presented a hybrid static-dynamic analysis approach that restricts each PHP function's access to the database, providing better protection. ALIERO et al. [6] proposed an automatic black-box testing scanner to detect SQL injection vulnerabilities. They conducted experiments using various types of vulnerable websites, reducing the cost of manual detection. HLAING et al. [7] introduced a query tokenization method for detecting SQL injection based on a predefined dictionary. This approach matches the input strings with the contents of the corresponding dictionary to prevent SQL injection attacks. Hankerson et al. [8] employed pattern matching based on predefined patterns. This pattern matching method effectively addresses SQL injection issues with high efficiency, although it may require some modifications to the application to a certain extent. Wang Weiping et al. [9] utilized regular expressions to describe attack patterns, enabling injection attack checks on HTTP requests before they are processed by the system module. Compared to keyword-based filtering, the approach based on regular expressions yields better filtering effectiveness. Shi et al. [10] proposed a SQL injection detection method that matches SQL statements with a knowledge base using pattern matching algorithms. For SQL statements that do not match, they do not immediately classify them as illegal. Instead, they employ a dynamic feature filtering algorithm based on risk values to perform in-depth feature inspection and identify true illegal SQL statements. They designed and implemented a prototype system that exhibits good performance advantages. The majority of the mentioned web attack detection models are rule-based, relying on pattern matching between attack sample features and feature databases.

Rule-based detection approaches are convenient to implement; however, their effectiveness in detecting unknown web attacks is inferior to that of machine learning-based detection methods [11]. Commonly used machine learning algorithms for web attack detection include HMM, K-means, SVM, among others. Alwageed et al. [12] employed an SVM-boosting algorithm framework to analyze the adaptability of various alternating supervised learning (SVM-FS) hybrid methods for detecting FDI attacks using data obtained from smart grids. Wu Shaohua et al. [13] extracted features related to SQL injection and XSS attacks, selected and summarized six features, and trained and classified them using the SVM algorithm. The feasibility of their detection approach is validated on the Weka platform. Rashid et al. [14] explored the impact of adversarial cyber attacks on machine learning models. They proposed a method of using adversarial retraining to enhance the detection accuracy of attacks. Zhu Jingwen et al. [15] proposed a SQL injection detection method based on Hidden Markov Models (HMM). This method enables prediction of SQL injections based on custom logs without the need to capture sensitive information submitted by users. By employing probability analysis and deviation analysis, the method provides a comprehensive evaluation of the likelihood of SQL injection by users, thereby enhancing the attack detection capability for target web applications. Delplace et al. [16] analyzed traffic data containing common botnet activities and evaluated the detection effectiveness of five different machine learning algorithms. Random Forest outperforms the other four models in eight out of thirteen scenarios. Yang et al. [17] introduced an intrusion detection framework called LCCDE, which consists of three integrated learning models. The framework utilizes prediction confidence to determine the model used for each prediction category.

The aforementioned machine learning-based methods for web attack detection, while capable of identifying unknown web attacks, still fall under the category of static detection methods, requiring the assistance of web application source code for detection. To achieve dynamic web attack detection, honeypots can be utilized to capture and analyze web logs. Zhuge Jiawen et al. [18] provided a comprehensive discussion on the application of honeypot technology in security threat monitoring research. Thanks to the efforts of open-source teams like The Honeynet Project, honeypot technology has made significant progress in areas such as attack feature extraction, forensic analysis, and network tracing. Zhai Guangqun et al. [19] developed a defense system that combines intrusion detection with honeypot technology. By collecting intrusion information through honeypots and analyzing the intrusion data using unsupervised clustering algorithms, they extracted new attack features. Experimental evidence has shown that this approach improves the efficiency of detecting unknown intrusion

attacks. Zhu Tao et al. [20] employed honeypot technology to construct a network attack log analysis system and designed a storage format for attack log files. They used regular expression matching to identify five types of keywords related to zombie network attack data and conducted analysis using a two-stage clustering algorithm. The experimental results indicate that most of the selected attributes, except for the four attributes related to packet size, exhibit strong clustering discrimination ability and can serve as important features for further intelligent analysis. Unicorn [21] identifies the traceability diagram of the system's normal execution from logs, thereby identifying abnormal behavior of APT attacks. ProvDetector [22] learns the path of the system's normal execution to identify abnormal APT attacks. Mendon ç a et al. [23] proposed an intrusion detection system based on convolutional neural networks. Jemal et al. [24] converted HTTP requests into code and classified them using CNN to detect malicious requests.

III. APPROACH

The architecture of the web attack detection system implemented in this paper is depicted in Figure 1. The system is divided into four major modules. The honeypot module is responsible for deploying a fictitious web system and opening it up to external access, thus recording its access logs. The data preprocessing module is used to normalize the logs. It first filters out irrelevant logs and then extracts relevant fields from the logs. Based on this, the logs are labeled and the attack types are identified. The machine learning module receives the preprocessed data and generates classification models for the logs, which are then validated. The attack detection module is responsible for detecting newly generated logs, filtering out sensitive logs, and recording their key information.

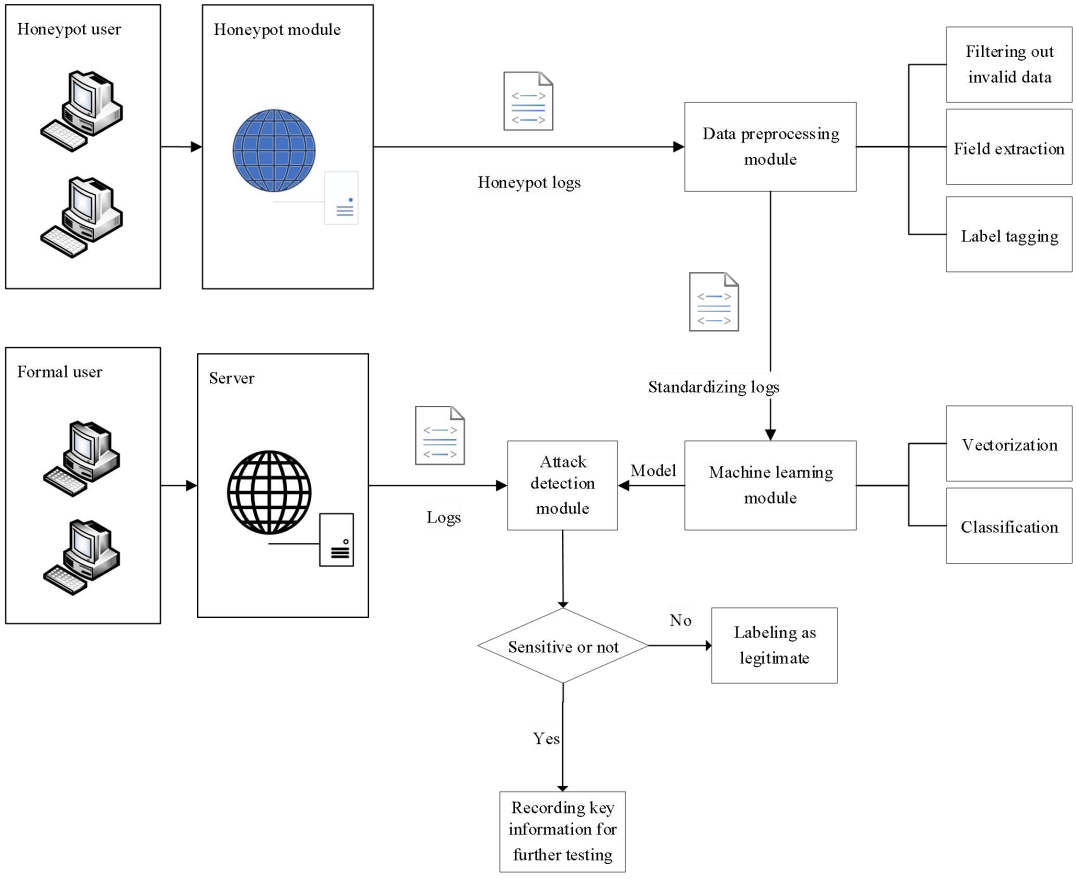

Figure 1: System Modules

A. *Honeypot*

The concept of a honeypot essentially encompasses a deceptive technique employed against malicious attackers. In this system, strategically deployed host machines serve as enticing baits to lure potential attackers on the internet, enabling the capture and analysis of their attack behavior. This facilitates an understanding of the attack methods employed against the system, which in turn allows for the modeling of potential attack scenarios. To ensure the effectiveness and diversity of attack samples, skilled individuals with expertise in attack techniques are also invited to perform targeted attacks. This approach guarantees the universality of the attack detection model.

B. Honeypot Data Preprocessing

The raw samples of log data obtained from the honeypot contain a mixture of various information, including a significant amount of noisy content. This noise can manifest in different forms, such as log format errors or logs generated by web crawlers like Google or Baidu. Therefore, it is not advisable to directly feed the raw log data into the machine learning module for model derivation. Training machine learning models with such raw data, regardless of the algorithm used, can easily lead to substantial errors, deviating from the expected experimental results. To address these challenges, a data preprocessing module is employed to handle the raw log data and ensure its normalization.

This paper lies in the detection of SQL and XSS injection attacks. Therefore, the normalized data is categorized into three classes based on the nature of the log data samples and the complexity of the attack keywords used. These categories include legitimate requests, SQL injection attacks, and XSS attacks. The data is manually labeled, with SQL injection and XSS attacks combined as attack requests.

C. Machine Learning Module

The collected log data samples mentioned above essentially consist of strings of varying lengths, which makes it difficult to directly train them using machine learning techniques. To address this, it is necessary to extract numerical features from these log data samples for training a classification model used for detection. The training process is illustrated in Figure 2.

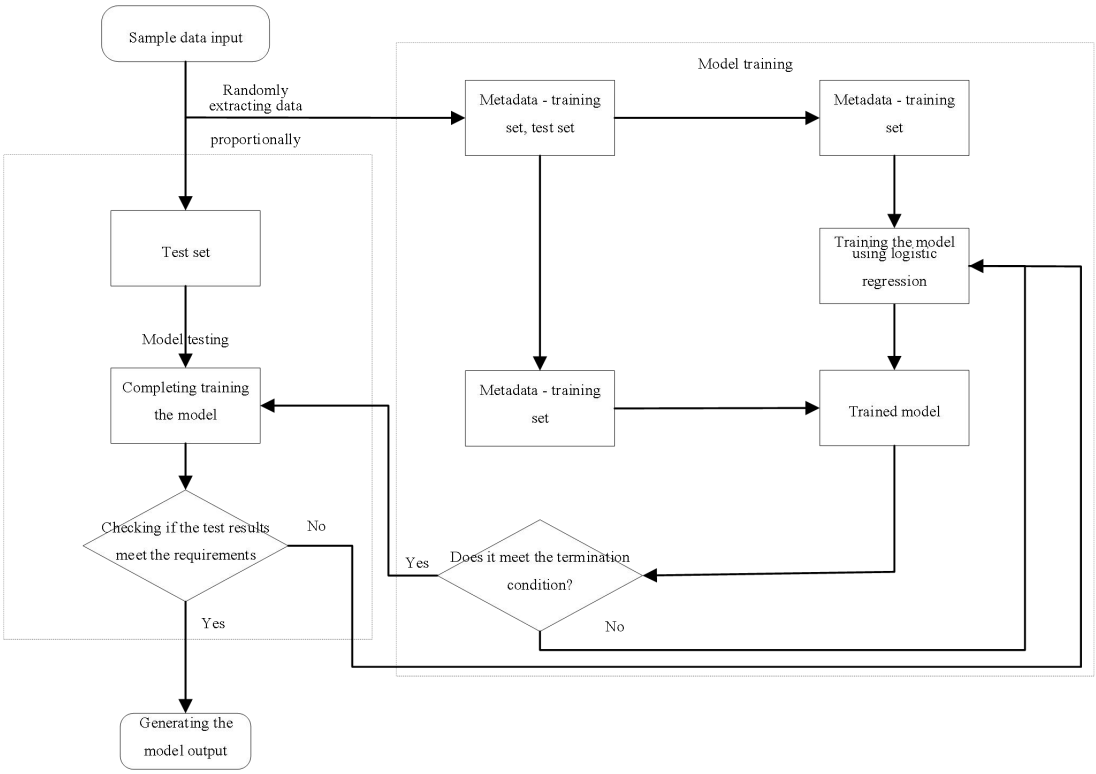

Figure 2: Model Training Flow Chart

In this paper, TF-IDF is first applied to vectorize the text. TF-IDF algorithm is a common text vectorization technique, where TF stands for “Term Frequency”, DF represents the “Document Frequency”, which indicates the number of documents in which a particular term appears. IDF, on the other hand, represents the “Inverse Document Frequency” and is calculated as IDF=log(N/(1+DF)). The value of TF-IDF is obtained by multiplying the values of TF and IDF (TF*IDF).

After vectorization, a feature matrix is generated for the classification model, which is then used to establish a detection model using logistic regression. Logistic regression is a classification and prediction learning model that assigns a “score” to each sample and sets a threshold. Samples that reach or exceed this threshold are classified into one category, while those that do not are classified into another category. For example, let’s assume that 0 represents normal web requests and 1 represents web attacks. The conditional probability P(Y=1|X) represents the probability of an attack occurring based on the feature variables X. Using the logistic regression algorithm, this can be represented as follows:

$$P(Y=1|X)=\frac{1}{1+e^{-g(X)}}=\frac{1}{1+e^{-(\beta_0+\beta_1X_1+\ldots+\beta_nX_n)}} \quad (1)$$

Where, X represents the feature vector of the URL field in the log, which is the TF-IDF feature matrix calculated earlier. β 0 is the intercept term, and β =(β 0, β 1, ..., β n) represents the regression coefficients of the feature vector X. These coefficients can be obtained by training the model on a training set.

D. Attack Detection Module

The attack detection module performs data parsing and feature extraction on the logs generated during the normal operation of the system. It feeds these features into the machine learning module's model for detection, distinguishing between sensitive logs and legitimate logs, and applies corresponding annotations. For sensitive logs, key information is recorded to facilitate further detection. The detection process is illustrated in Figure 3.

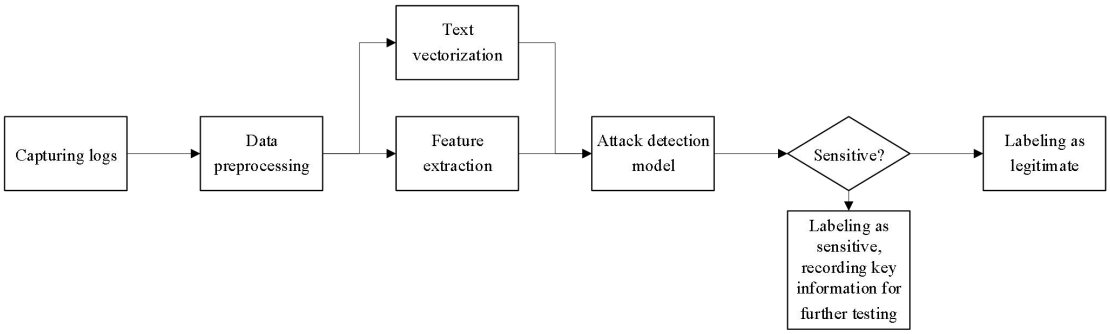

Figure 3: Attack Detection Flow

IV. Implement and Experiment

The architecture of the web attack detection system implemented in this paper is depicted in Figure 1. The system is divided into four major modules. The honeypot module is responsible for deploying a fictitious web system and opening it up to external access, thus recording its access logs. The data preprocessing module is used to normalize the logs. It first filters out irrelevant logs and then extracts relevant fields from the logs. Based on this, the logs are labeled and the attack types are identified. The machine learning module receives the preprocessed data and generates classification models for the logs, which are then validated. The attack detection module is responsible for detecting newly generated logs, filtering out sensitive logs, and recording their key information.

A. Honeypot Module

The present study involves the deployment of a honeypot on an open-source honeypot platform. The honeypot is deployed on a Linux system and follows a B/S architecture, consisting of a management side and a node side. The management side is responsible for generating and managing the nodes, as well as receiving, analyzing, and displaying the data transmitted by the nodes. On the other hand, the node side receives control commands from the management side and handles the construction of the honeypot services.

The deployed honeypot has the ability to collect data by recording information through the management side. It enables the logging of access logs, ensuring the integrity of the log files by preventing attackers from deleting logs on the node side.

The collected log sample, as shown in Table 1, illustrates a modest XSS attack scenario. It documents details such as IP address, timestamp, request method, URL, and access status.

Table 1: Sample Honeypot Log

Logs	178.137.18.55 - - [30/Oct/2013:19:42:53 -0500] "GET /archiver/?tid-39 =;alert(1);// HTTP/1.0" 200 1127 "http://spechome.ru/?m=20120808" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0)
Meanings	IP address: 178.137.18.55 Access time: Oct. 30, 2013 Make a GET request for website resources /archiver/ and submit the parameter tid-39 as alert (1); //The access status is 200, the number of access bytes is 1127, the protocol version is HTTP/1.1, and the refer field is http://spechome.ru/?m=20120808,the user agent is Mozilla 5.0

B. Data Preprocessing Module

The data preprocessing module begins by applying simple cleansing techniques to the log data, primarily using regular expressions, supplemented by manual intervention. It aims to ensure proper formatting by matching

the provided raw log data from the internet and the server log data obtained from our operations on Alibaba Cloud. The regular expression used for matching is based on the Apache2 log format, and is as follows:

'"(?P<remote_addr>[\d\.]{7,}) - - (?:\[(?P<datetime>[^\[\]]+)\]) "(?P<request>[^"]+)" (?P<status>\d+) (?P<size>\d+) "(?:[^"]+)" "(?P<user_agent>[^"]+)""

Once the log data has been matched using the regular expression pattern, it will undergo automatic filtering and be placed into the preprocessing pool. Any logs that were not matched or identified as erroneous will be categorized for manual intervention in the cleansing pool. Logs that significantly deviate from the required format will be promptly discarded, while logs that can still be utilized will be placed into the preprocessing pool for further processing.

Within the preprocessing pool, further cleansing of the data occurs, involving several steps such as removing unique attributes, handling missing values, attribute encoding, data standardization, normalization, feature selection, and principal component analysis. To begin, the preprocessing phase involves identifying and removing duplicate data entries, thereby eliminating unique attributes. For samples that contain missing values, the approach varies depending on the severity of the missing values. If a sample has substantial missing values across all fields, rendering it insufficient for providing valuable labels, it is excluded from further processing. However, if the missing values in a sample do not impact the training sampling significantly, it can be retained. In cases where non-missing critical fields are present in a sample, the preprocessing phase initially reserves the main components of the label attributes. Subsequently, based on the label attributes, corresponding information from unrelated fields can be manually annotated and added.

Tables 2 and 3 present a partial display of logs that meet the requirements and those that do not. From Table 3, it is evident that logs that fail to meet the requirements are typically excluded due to reasons such as inconsistent timestamp format (Log 1), the absence of a timestamp field (Log 2), missing URL information (Log 3), being identified as Google crawler agents (Log 4, 5), or lacking an IP field (Log 5).

Table 2: The Filtered Logs That Meet the Requirements

1	24.148.0.215 - -[21/Jul/2011:14:69:22 -0500] "GET /wp-includes/js/iquery/jquery.js HTTP/1.1"20 72194 "http://www.npcassoc.org/wp-admin/install.php" "Mozilla/5.0 (Windows NT 6.1: WOW64) AppleWebkit/534.30 (KHTML, like Gecko) Chrome/12 日.742.122 Safari/534.30"
2	132.16.114.136 - - - "POST /wp-admin/install.php?step=2HTTP/1.1"200 86"http://www.npcassoc.org/wp-admin/install.php"Mozilla/5. (windows NT 6.1; WOW64)AppleWebKit/534.3 (KHTML,like Gecko) Chrome/12.admin/install.php"Mozilla/5. (windows NT 6.1; WOW64)0.742.122 Safari/534.30"
3	119.228.218.116 - -[29/Jul/2017:14:44:59 -0500] - 200 7547 "http://www.npcassoc.org/wp-login.php" "Mozilla/5.0 (Window5 NT 6.1: WOW64) ApleWebkit/534.39 (KHTML. like Gecko) Chrome/12.0.742.122 Safari/534.30"
4	89.25.39.0 - - [31/Jul/2010:22:26:38 -0500] "GET /ournal/table-of-contents/vol-2-no-3HTTP/1.1"200 5710 "" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
5	- - - [21/Ju1/2019:05:22:48 -51 404 5651 '"Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"

Table 3: The Filtered Logs That Do Not Meet the Requirements

1	124.35.43.7 - [24/0ct/2019:20:18:15 +0800] "GET /coin_operated/ HTTP/1.1" 200 3477 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appleebkit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120 Safari/537.36"
2	123.104.154.137 - - [20/Mar/2020:22:15:19 +0800] "GET /dff_php_frameworkapi-latest/include/dff sku.func.php?dff configIdir include=http://192.168.202.96:8080/frznctvhi0i5?/cacti/myphppagetool/doc/admin/index.php; ptinclude-http://worldfile.com
3	197.187.102.123 - - [20/Mar/220:22:15:19 +0800] "GET /saint-80x37/
4	94.234.229.56 - - [20/Mar/2020:22:15:19 +0800] "GET /apbn/templates/head.php?apb settings[template path]=@rfiurl

After data cleaning, the data can be standardized according to the following format: “|Serial Number (optional)|IP Address (required)|Date (required)|Request Method (required)|Request Resource Location (required)|HTTP Protocol Version (optional)|Bytes Sent (optional)|Requesting Tool (optional)|”.

To ensure the completeness of the attack samples, this paper introduces a portion of publicly available datasets from the internet to supplement the samples. The introduced samples consist of normal samples, XSS attack samples, and SQL attack samples from Internet.

Finally, all samples are labeled accordingly, as mentioned in section 3.2, to facilitate the establishment of the model.

C. Machine Learning Module

1) Text vectorization

In this study, the initial step involves utilizing N-Gram to tokenize the URLs in the log data, thereby vectorizing them. Once the tokenization process is complete, the TF and IDF values are computed. These values are then multiplied together to obtain the TF-IDF value for each individual word.

Taking a simplified version of the log data containing SQL injection and XSS attack keywords as an example, as shown in Table 4:

Table 4: Simplified Version of Log Data with Attack Keywords

line1	Script	null	ActiveXobject	version	alert	version
line2	version	alert	Script	alert	Script	ActiveXobject
line3	load_file	load_file	null	load_file	Concat	null
line4	alert	null	alert	null	null	null

For the mentioned lines and vocabulary, we can calculate the corresponding TF and DF values, as shown in Table 5:

Table 5: Calculation of TF and DF for Simplified Log Data

	Script	alert	ActiveXobject	version	load_file	Concat
TF1	1/5	1/5	1/5	2/5	0	0
TF2	1/3	1/3	1/6	1/6	0	0
TF3	0	0	0	0	3/4	1/4
TF4	0	2/3	0	1/3	0	0
DF	2	3	2	3	1	1

With the TF data and the DF data for the documents mentioned above, we can calculate the inverse document frequency (IDF) and TF-IDF using the formulas. The results are shown in Table 6:

Table 6: Calculation Results of IDF and TF-IDF

IDF=log(N/(1+DF))	log(4/(1+2)) =log(4/3)	log(4/(1+3)) =log(4/4)	log(4/(1+2)) =log(4/3)	log(4/(1+3)) =log(4/4)	log(4/(1+1)) =log(4/2)	log(4/(1+1)) =log(4/2)
TFIDF1	1/5*log(4/3)	1/5*log(4/4)	1/5*log(4/3)	2/5*log(4/4)	0	0
TFIDF2	1/3*log(4/3)	1/3*log(4/4)	1/6*log(4/3)	1/6*log(4/4)	0	0
TFIDF3	0	0	0	0	3/4*log(4/2)	1/4*log(4/2)
TFIDF4	0	2/3*log(4/4)	0	1/3*log(4/4)	0	0

The higher the TF-IDF value of a word in an article, the generally more important the word is within that article. Therefore, by calculating the TF-IDF values for all the words in the article and sorting them in descending order, the top few words are considered to be the keywords of that article.

After completing the description of text features using the TF-IDF algorithm, we obtain a sparse matrix as the feature matrix required for generating a classification model. The sparse matrix is stored in the DOK (Dictionary of Keys) format, where the row-column values serve as keys in the dictionary, and the matrix elements represent the dictionary content. The format is [(m, n) tf-idf], where “m” represents the log number, “n” represents the word number, and “tf-idf” represents the tf-idf value of the word with the corresponding number “n” in the log with the number “m”.

2) Construction of classification model

This paper employs a logistic regression model as a classifier for determining the presence of attack behavior in logs. The experimental setup is presented in Table 7:

Table 7: Experimental Environment

Operating System	Ubuntu 20.04.1 LTS
Development Language	Python 3.7.11
Machine Learning Library	Scikit-Learn 1.3.0
Model	Logistic Regression

The logistic regression model in this study utilizes L2 regularization, optimizing the loss function through iterative updates using the second derivative matrix, known as the Hessian matrix. The classification is performed

using the one-vs-rest (ovr) method. The model is trained by feeding it with input sample data and corresponding label data. Subsequently, a separate test set and its labels are used to evaluate the model's accuracy in predicting outcomes. Based on the test set results, the model achieves a self-test accuracy of 91.90%.

After generating the model, it is further evaluated using a labeled test set to assess its performance. The classification results align with the expected outcomes at a rate of 87.64%, providing compelling evidence for the effectiveness of the generated model.

D. Attack Detection And Performance Evaluation

In this study, in addition to the logistic regression model, the Support Vector Machine (SVM) approach is also employed to generate a classification model for comparison. The sample log data is vectorized, and different amounts of data are fed into the two machine learning models. The experiments are conducted based on accuracy, training time, and testing time as evaluation metrics. The results obtained are presented in Tables 8 and 9. It is evident from the tables that the logistic regression method outperforms the SVM method in terms of both accuracy and time consumption, making it more practical in this context.

Table 8: Calculation Results Using SVM

Data volume	Accuracy	Training time (μs)	Testing time (μs)
1000	0.470000	35904	8976
3000	0.618000	314192	93748
5000	0.318000	945470	309201
8000	0.125000	535209	970370
10000	0.080000	662132	1539995
15000	0.044000	800504	1779567

Table 9: Calculation Results Using Logistic Regression

Data volume	Accuracy	Training time (μs)	Testing time (μs)
1000	0.935000	38895	1031
3000	0.911000	45762	2047
5000	0.870000	206450	3192
8000	0.806000	506543	7037
10000	0.770000	688923	6570
15000	0.714000	981000	12939

V. CONCLUSION

A detection method for XSS and SQL injection attacks, based on honeypots and the logistic regression algorithm, was designed in this study. The log samples captured by the honeypot were preprocessed, and selected features were subjected to vectorization before being incorporated into the training of the logistic regression model. The obtained attack detection model was experimentally validated and found to possess practicality in terms of accuracy and performance. However, it should be noted that the method presented in this paper has certain limitations. These include a single source for the training dataset, the need for expansion in the detectable attack types, and further applications of the detection results. These aspects can be explored as potential research directions in the future.

ACKNOWLEDGMENT

This paper is supported by The "New Generation Information Technology Innovation Project" of the China University Industry, University and Research Innovation Fund(No. 2021ITA07007), Youth scientific research project of Guangdong police college (No. 2022-QN04), Special Funds for the Cultivation of Guangdong College Students' Scientific and Technological Innovation("Climbing Program" Special Funds, No. pdjh2021b0357, No. pdjh2022b0373, No. pdjh2024b276).Provincial undergraduate innovation and entrepreneurship training program(S202311110030).

REFERENCES

[1] Chen Junxin. Research on XSS Attack Detection Technology Basedon Machine Learning. Hangzhou: ZheJiang University of Technology, 2018.

[2] Roghaie Mahdavi, Abolghasem Saiadian. Efficient Scalar Multiplications for Elliptic Curve Cryptosystems Using Mixed Coordinates Strategy and Direct Computations//Cryptology and Network Security-international Conference. DBLP, 2010: 184-198.

[3] Oluwakemi Christiana Abikoye, Abdullahi Abubakar, Haruna Ahmed Dokoro, AKANDE NOAH OLUWATOBI, Aderonke Anthonia Kayode. A Novel Technique to Prevent SQL Injection and Cross-site Scripting Attacks Using Knuth-morris-pratt String Match Algorithm. Eurasip Journal on Information Security, 2020(1): 1-14.

[4] JANA A., MAITY D. Code-based Analysis Approach to Detect and Prevent SQL Injection Attacks//IEEE. 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), July 1-3, 2020, Kharagpur, India. Piscataway, New Jersey: IEEE, 2020: 1-6.

[5] Rasoul Jahanshahi, Adam Doupé, Manuel Egele. You Shall Not Pass: Mitigating SQL Injection Attacks on Legacy Web Applications//SIGSAC. The 15th ACM Asia Conference on Computer and Communications Security, October 5, 2020, Taibei, China. New York: Association for Computing Machinery, 2020: 445-457.

[6] M. S. Aliero, I. Ghani, K. N. Qureshi, Rohani, Mohd Fo' ad. An Algorithm For Detecting SQL Injection Vulnerability Using Black-box Testing. Journal of Ambient Intelligence and Humanized Computing, 2019, 11(1): 249-266.

[7] HLAING Z. CS. S., KHAING M. A Detection and Prevention Technique on SQL Injection Attacks. 2020 IEEE Conference on Computer Applications (ICCA), February 27-28, 2020, Yangon, Myanmar. Piscataway, New Jersey: IEEE, 2020: 1-6.

[8] HANKERSON Darrel, MENEZES Alfred, VANSTONE Scott. Guide to elliptic curve cryptography. Heidelberg: Springer-Verlag Professional Comuting Series, 2004.

[9] WANG Wei-ping, LI Chang, DUAN Gui-hua. Design of SQL Injection Filtering Module Based on Regular Expression. Computer Engineering, 2011, 37(5): 158-160.

[10] Congcong Shi, Tao Zhang, Yong Yu, WeiminLin. New approach for SQL-injection detection, Computer Science, 2012, 39(6): 60-64.

[11] W. Wang, J. Liu, G. Pitsilis, X. Zhang. Abstracting massive data for lightweight intrusion detection in computer networks. Information Sciences, 2016(2): 1-14.

[12] Alwageed, Hathal Salamah. Detection of cyber attacks in smart grids using SVM-boosted machine learning models.Service Oriented Computing and Applications, (2022): 1-14.

[13] WU S.H., CHRNG S. B., HU Y. Web Attack Detection Method Based on Support Vector Machines. Computer Science, 2015, 42(6A): 362-364.

[14] MM. Rashid, J. Kamruzzaman, MM. Hassan, T. Imam, S. Wibowo, S. Gordon, G. Fortino. Adversarial training for deep learning-based cyberattack detection in IoT-based smart city applications. Computers & Security (2022).

[15] Zhu Jingwen, Xu Jing, Chen Liang, Li Jie. SQL injection attack detection based on hidden markov model. Computer Applications and Software, 2023, 40(02): 331-344.

[16] Delplace A., Hermoso S., Anandita K. Cyber attack detection thanks to machine learning algorithms. arXivpreprint, arXvi: 2001. 06309, 2020

[17] Yang L., Shami A., Stevens G., et al. LCCDE: A decision-based ensemble framework for intrusion detection in the internet of vehicles. Proc of the 2022 IEEE Global Communications Conf. Piscataway, NJ:IEEE,2022:3545-3550

[18] ZHUGE Jian-Wei, TANG Yong, HAN Xin-Hui, DUAN Hai-Xin. Honeypot Technology Research and Application. Journal of Software, 2013, 24(4):825-842.

[19] ZHAI Guang-qun, CHEN Xiang-dong, HU Gui-jiang. Research and design on linkage system of Honeypot and intrusion detected technology. Computer Engineering and Design. 2009, 30(21): 4845-4867.

[20] ZHU Tao, XIA Lingling, LI Penghui, Xu Zhongyi. Analysis of Botnet Attack Data Based on Log. Netinfo Security. 2022, 22(10): 82-90.

[21] X. Han, T. Pasquier, A. Bates, J. Mickens, M. Seltzer. UNICORN: Runtime provenance-based detector for advanced persistent threats. In: Proc. of the Network and Distributed Systems Security (NDSS 2020) Symp. 2020. The Internet Society, 2020.

[22] Q. Wang, WU. Hassan, D. Li, K. Jee, H. Chen. You are what you do: Hunting stealthy malware via data provenance analysis. In: Proc. of the Network and Distributed Systems Security (NDSS 2020). The Internet Society, 2020.

[23] RV. Mendonca, AAM. Teodoro, RL. Rosa, M. Saadi, DZ. Rodriguez. Intrusion detection system based on fast hierarchical deep convolutional neural network. IEEE Access, 2021, 9: 61024-61034.

[24] Jemal I., Haddar M. A., Cheikhrouhou O., et al. Malicious http request detection using code-level convolutional neural network. Risks and Security of Internet and Systems: 15th International Conference, CRiSIS 2020, Paris, France, November 4 – 6, 2020, Revised Selected Papers 15. Springer International Publishing, 2021: 317-324.