Science & Technology Development Journal 2023, 26(2):2729-2740

Open Access Full Text Article

Natural Sciences - Research article

Deception and Continuous Training Approach for Web Attack Detection using Cyber Traps and MLOps

Van Hau Pham[1,2], Nghi Hoang Khoa[1,2], Nguyen Huu Quyen[1,2], Phan The Duy[1,2]

Use your smartphone to scan this QR code and download this article

ABSTRACT

With the growth and expansion of the internet, web attacks have become more powerful and pose a significant threat in the cyber world. In response to this, this paper presents a deceptive approach for gathering malicious behavior to understand the strategies used by web attackers. The harmful requests collected through cyber traps or honeypots are analyzed and used to train machine learning (ML) models for web attack detection. Additionally, we implement an ML operations (MLOps) pipeline to automate the continuous training and deployment of these ML models in defensive systems. This pipeline trains the production model with newly collected data by using predefined triggers. Our experiments on two datasets, including Fwaf and our own, demonstrate that a proactive and continuous approach to tracking adversary behavior can effectively detect zero-day attacks, such as CVE-2022-26134 in web application servers.

Key words: Web attack detection, Cyber trap, Cyber attack detection, Cyber deception

INTRODUCTION

Daily life has seen a significant increase in the volume and speed of data since the advancement of information technology. Furthermore, the Internet has transformed traditional methods of daily life. Web applications have become the most widely used applications on the Internet. Web applications are used in different areas and are important in people's daily routines, especially as more individuals transfer their applications and personal information to the cloud. Due to the widespread use of web applications and the significant amount of personal data saved on servers, they become attractive targets for attacks.

A recent report about cybersecurity[1] incidents illustrated that 75% of attacks are detected in the application layer, while web servers are the main targets of hackers. There are two reasons why adversaries are inclined to intrude and break into web servers. First, because large amounts of private data are stored in server databases, attackers can profit significantly by selling that data. Second, the ability to inject malicious code into server source files allows attackers to hack and manipulate users who browse or download these documents.

Obviously, safeguarding web applications from intrusions is essential. Typically, attacks are detected primarily based on recognizable characteristics. This method is useful for detecting known attack types, but it necessitates human involvement in collecting and analyzing attack data samples for identification. Consequently, identity-based website attack detection (WAD) is no longer adequate for detecting attacks with novel exploits.

Additionally, with the rapid and significant advancement in the field of ML, these algorithms have demonstrated their effectiveness in numerous fields, including web attack detection. Some research has achieved desired results when incorporating ML into web protection systems. To automatically detect denial-of-service (DoS) attacks, Francisco *et al.*[2] proposed an ML model to make inferences based on signatures previously extracted from samples of network traffic. The results of this model on four modern benchmark datasets have achieved an online detection rate (DR) of attacks above 96%.

In the same domain of WAD, Liang *et al.*[3] proposed a deep learning-based approach to classify anomalous requests. Recurrent neural networks (RNNs) were used to learn patterns of normal requests using only unsupervised normal requests. Then, a neural network classifier that takes the outputs from the RNNs as input was trained in a supervised manner to differentiate between anomalous and normal requests. Similarly, Yunyi *et al.*[4] suggested a WAD that applied a long short-term memory (LSTM) network to analyze the malicious intentions hidden in user actions. The experimental results on the CSIC 2010 dataset achieved an accuracy of 99.87%.

In addition, the ML approach for detecting many attack types needs administrators or data scientists

[1]*Information Security Laboratory, University of Information Technology, Ho Chi Minh city, Vietnam*

[2]*Vietnam National University, Ho Chi Minh city, Vietnam*

History

- Received: 2023-02-10
- Accepted: 2023-06-21
- Published: 2023-06-30

DOI :

https://doi.org/10.32508/stdj.v26i2.4044

Check for updates

VNU-HCM Press

Cite this article : Pham V H, Khoa N H, Quyen N H, Duy P T. **Deception and Continuous Training Ap-proach for Web Attack Detection using Cyber Traps and MLOps**. *Sci. Tech. Dev. J. 2023;* 26(2):2729-2740.

to understand and select appropriate representation data for the training ML model. These approaches use datasets published on the internet or create a dataset to train the ML model. Therefore, it has a limitation in recognizing rare attacks that are much different from existing attack patterns in datasets. It requires administrators to continuously collect new attack patterns to enrich their datasets. However, there are difficulties in updating data with new attack patterns, which consumes more time and effort. In this context, an effective method for creating a training dataset was introduced by Nikola *et al.*[5] to effectively improve the performance of WAD without many false positives. They combined hostile requests from cyber traps with good requests from a typical website. Based on the features of the deceptive network, a system is vulnerable to the attack that entices an attacker to exploit the vulnerability. It is easy to collect and update datasets with new indicators to overcome the weaknesses of the ML system.

Furthermore, to avoid forgetting the existing knowledge when training the ML model with new data, three incremental learning approaches are also implemented for WAD and obtained good results during testing. Moreover, to evaluate the use of deception in the domain of web applications, Xiao *et al.*[6] implemented a web deception framework that allows us to introduce deception in any web application. In their experiments, the authors showed that over 36% of attackers who were able to exploit a vulnerability did not set off any of their traps. Their research has shown that while deception is a useful complement to other detection techniques, it is insufficient as a stand-alone protection mechanism.

ML initiatives have produced fresh difficulties that do not exist in conventional software development. Keeping production deployment current, one of these involves tracking input data, data versions, tuning parameters, *etc*. Meanwhile, MLOps was defined as including three parts: ML, Development and Operations (DevOps), and data engineering. The largest effect on MLOps development came from DevOps[7]. To map how MLOps is currently understood and how it compares and differs from related techniques such as DevOps, the article[8] used meta-analysis, document analysis, and triangulation. This research gives up the comment that these related studies[8–11] effectively conceptualize MLOps and demonstrate that it lies at the nexus of software engineering, data engineering, DevOps, and ML. With this characteristic, MLOps is considered a potential solution for continuing to harden the robustness of WAD.

Moreover, S. Garg *et al.*[12] discussed open research problems and provided a detailed representation for automation with DevOps in ML-based applications, called MLOps. The pipeline seeks to achieve the advantages of both contexts by keeping the DevOps pipeline's trademark simplicity and incorporating new circular phases for updating ML. This aims to produce a self-maintaining ML-based development subsystem that may advance concurrently with software development. The authors in Garg *et al.* (2023)[13] also described the three tiers of MLops and open-source platforms that help users easily monitor the workflow of the system, visualize operations, and trace errors when building and operating the ML model.

Unlike the above approaches, our work focuses on web attack detection, which enables a continuous training strategy based on cyber traps and MLOps. The main contributions in this article are summarized as follows:

- First, an ML-based WAD using a stacking classifier is proposed to harness the capabilities of detecting web attacks with better performance than any single model in the ensemble.
- Second, we integrate the MLOps pipeline to formalize a strategy of automatic training ML models frequently.
- Finally, a honeypot system is deployed to collect attack data to diversify the training dataset. In addition, a Service Website is built to take on the role of deploying ML-based WAD to users.

The remainder of this paper is structured as follows. In Section 2, the methodology and detailed workflow of our proposed system are described. The implementation and experiments are presented in Section 3. Finally, we conclude the paper in Section 4.

DESIGN

This section describes our approach to create an ML-based WAD with cyber traps to be proactive in collecting training datasets for the ML process. After that, we integrate the MLOps pipeline in our system detection to enhance automation in the deployment and maintenance of ML models.

A. Overall Model

In this part, we introduce a WAD model consisting of Kubeflow, Data Storage, Honeypot, and Serving Website, as shown in Figure 1. Each of these parts performs a different task and executes on separate hosts. The workflow of MLOps and the Deception model for WAD can be summarized as follows:

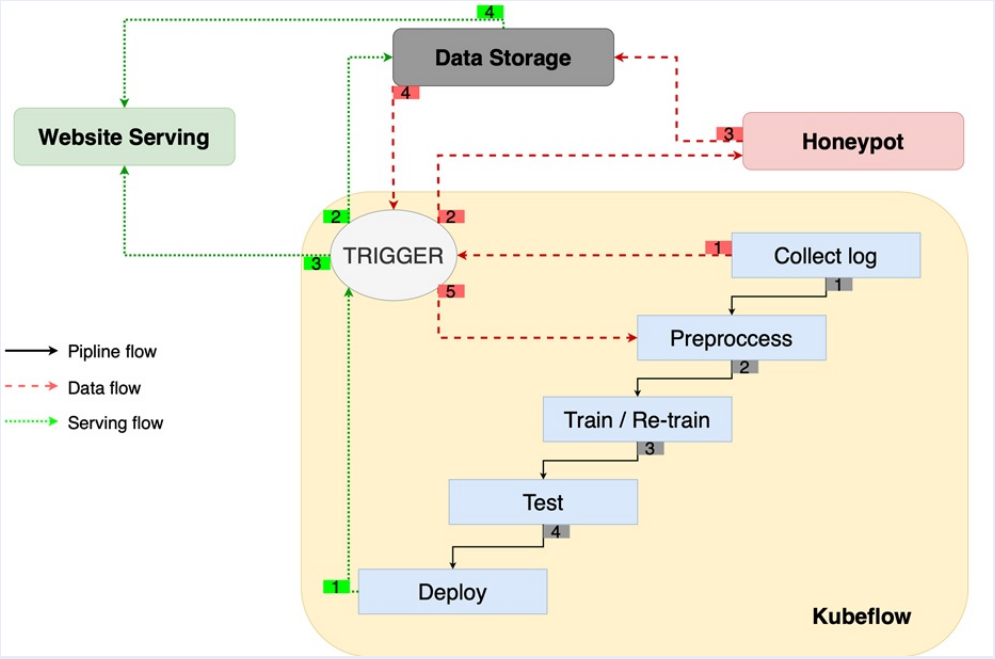

Figure 1: **The overall architecture of our model.**

- First, when executing the workflow, collected logs in Kubeflow send a data flow signal to Trigger.
- Next, a signal is sent to the honeypot to save all the collected data during the trap's working period on data storage.
- Data saved in Data Storage are preprocessed in the Preprocess block of Kubeflow to prepare for the training step.
- Next, the pipeline flow signal is sent to the train/retrain block.
- Finally, a new ML model is deployed on Data Storage. A serving flow signal is sent to the website serving to update the new model.

B. ML-based Web Attack Detection

In this study, our WAD model is built based on the ensemble method to reduce the uncertainty in the generalization performance of using a single algorithm [14]. In the ensemble model, we use 3 well-known ML algorithms in classification, including support vector machine (SVM), logistic regression (LR), and K-nearest neighbors (KNN). In addition, we also built a convolutional neural network (CNN) model to compare the performance of the ensemble method with that of a neural network.

Ensemble Learning (Stacking Classifier) model

Stacking, also known as a voting classifier, is an ensemble learning method. This method improves the model's performance by combining different ML classifiers for classification. Majority voting, a popular voting technique, is used in this approach. It has three scenarios based on unanimous voting, simple majority voting, and plurality voting. Furthermore, hard voting usually refers to plurality voting.

We assume that the decision of the t^{th} classifier is $d_{t,c} \in 0, 1,\ t = 1, \ldots, T$ and c = 1,…,C, where T is the number of classifiers and C is the number of classes. If the t^th classifier chooses class c, then $d_{t;c} = 1$, and 0 otherwise. Finally, mathematical representation-based plurality voting is computed to choose the class desired c^* as follows:

$$\sum_{t=1}^{T} d_{t;c^*} = \max_{c} \sum_{t=1}^{T} d_{t;c} \quad (5)$$

CNN model

In this study, the architecture of the CNN model contains 2 convolutional blocks, as shown in Figure 2. The input x contains features that indicate whether this is a good or bad. x is passed through 2 convolutional blocks denoted as ConvBlock_1 and

ConvBlock$_2$, respectively, as in (6).

$$h_1 = ConvBlock_1(x)$$
$$h_2 = ConvBlock_2(x) \quad (6)$$

Then, the output value of the 2^{nd} convolution block will be flattened as in (7).

$$f = Flatten(h_2) \quad (7)$$

This result f goes through a fully connected layer, as in (8).

$$c = FC_1(f) \quad (8)$$

Finally, the Softmax layer transforms the result c into values between 0 and 1 as the probability $\widetilde{y}$ (9).

$$\widetilde{y} = Softmax(c) \quad (9)$$

The cross-entropy function is used to optimize the loss value between the vector output $\widetilde{y}$ and vector y containing the actual label values.

C. MLOps for Web Attack Detection

In this study, we proposed a methodology for a WAD-based MLOps pipeline to create an automatic training rule for continuously updating ML models. The pipeline workflow is shown in Figure 3.

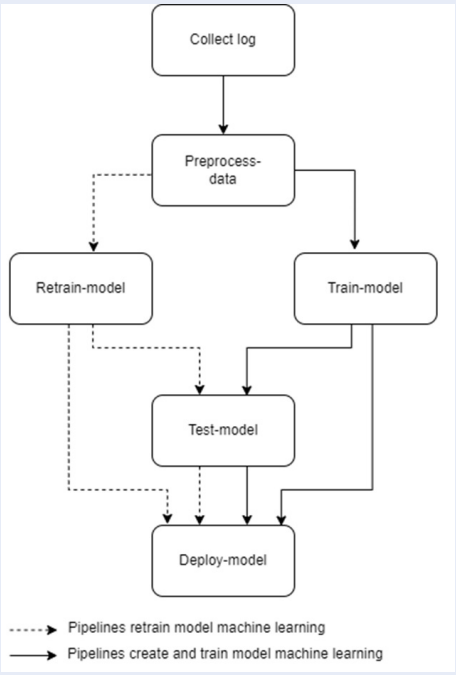

Figure 3: The workflow of WAD in integration with the MLOps pipeline.

The Log Collection Process

The log collection process as in Figure 4 is explained as follows:

The honeypot's daily process is located outside the internet to collect logs.

After an activation signal from Kubeflow, Honeypot turns on the firewall allowing only administrator access, and each step of the log processing is executed sequentially:

- Collect good requests: the process of simulating web page interactions as normal users to collect labeled datasets is normal.
- Collect bad requests: This process will collect the malicious exploit code attacked by the administrator.
- Parse and push log: process collected logs and send them to the Data storage.

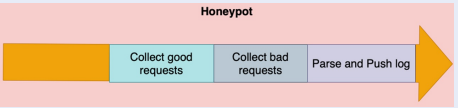

Figure 4: Log Collection Process.

Data Acquisition and Processing

Data acquisition is responsible for collecting enough data to train the ML model. The tasks for data acquisition and processing can be summarized as follows:

- **Data Extraction and Analysis**: Analyzing data to understand the data schema and integrating relevant data to maximize model performance.
- **Data Preparation**: is responsible for cleaning and splitting datasets into training, validation, and test sets. For instance, these outliers detected by the local outlier factor algorithm will be removed from the dataset, or *NULL* values are transformed to zero. This step results in formatted data for training ML models.

When there is not enough data to train or retrain the ML model, two main approaches allow for overcoming the issue as follows:

- **Data Augmentation**: This technique aims to balance non-IID (nonindependent and identically distributed) data. More specifically, when the difference ratio between the attack and normal labels is more than 20%, the conditional generative adversarial network (CGAN) will generate data with a minority label to keep the difference within 20%.

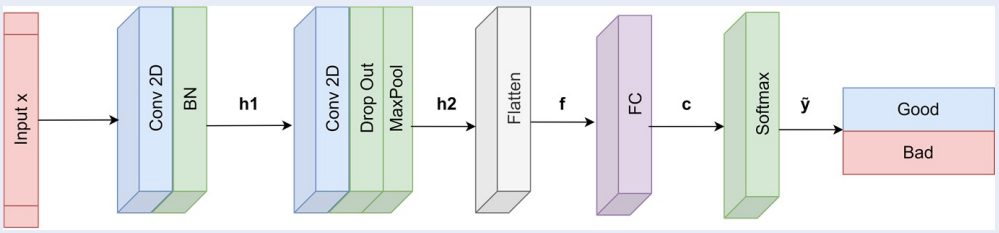

Figure 2: **The architecture of the CNN model.**

- **Incremental Learning (IL)**: which reuses most of the weights of a trained neural network.

Training, Testing, and Deploying the ML (ML) Model

The process of training ML models is an iterative process in which data scientists work with several algorithms, data features, and hyperparameters. The output of this step is a set of metrics for evaluating the quality of the model. Once the best ML model has been chosen, it is saved and deployed to the serving website. Our primary objective is to ensure that we monitor all testing experiments, provide the reusability of code, and uphold the ability to continue updating the robustness of ML-based WAD.

Building Honeypot System and Service Website

Honeypot System Overview

In this study, we apply the Web Honeypot to simulate specific web services and attract particular types of attacks by specific web technologies, as in Figure 5.

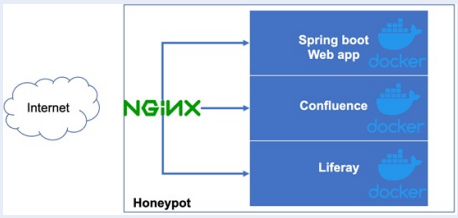

Figure 5: **Overview of the honeypot system.**

Honeypot runs potentially attractive web applications on the Nginx server using Java programming language. The system records all requests to the above web applications. At the same time, honeypots are always listening for trigger signals from Kuberflow to perform tasks of collecting datasets, processing data, and pushing data to Data Storage.

Trigger Processing

A trigger is a flag that tells the system when a recurring run configuration spawns a new run. The following types of run triggers are available:

- **Access Management**: allows objects based on the IP address to access the web application.
- **Nginx website server management**: formatting log data by getting the properties from the HTTP request packet sent by the user, as in Figure 6.
- **Access Log Processing**: As shown in Figure 7, each formatted access log is classified and sorted, and duplicates in the above-formatted dataset are removed. Finally, both goodrequest.txt and badrequest.txt files are pushed to the Data Storage.

Deploying ML Models to the Serving Website:

We are constructing a serving website designed in Figure 8, which continuously listens for updating new machine learning models. Additionally, the serving website will receive files in both.txt and.csv formats, containing good and bad requests. Subsequently, the serving website employs a machine learning model to classify these requests. To achieve the goal of detecting new attacks and avoiding false negatives, we selected a detection threshold of 0.5 for identifying bad requests. Therefore, the predicted label of the detection model is 1 (indicating a bad request) if the predicted value is greater than 0.5, and vice versa.

IMPLEMENTATION AND EXPERIMENT

In this section, we carried out a performance evaluation of the MLOps pipeline in a variety of scenarios. First, we describe data resources and partitions. Then, we concentrate on experimental settings, such as environmental conditions, baseline studies, and performance metrics. Finally, we ran a series of experiments to compare the MLOps pipeline's performance with various ML models.

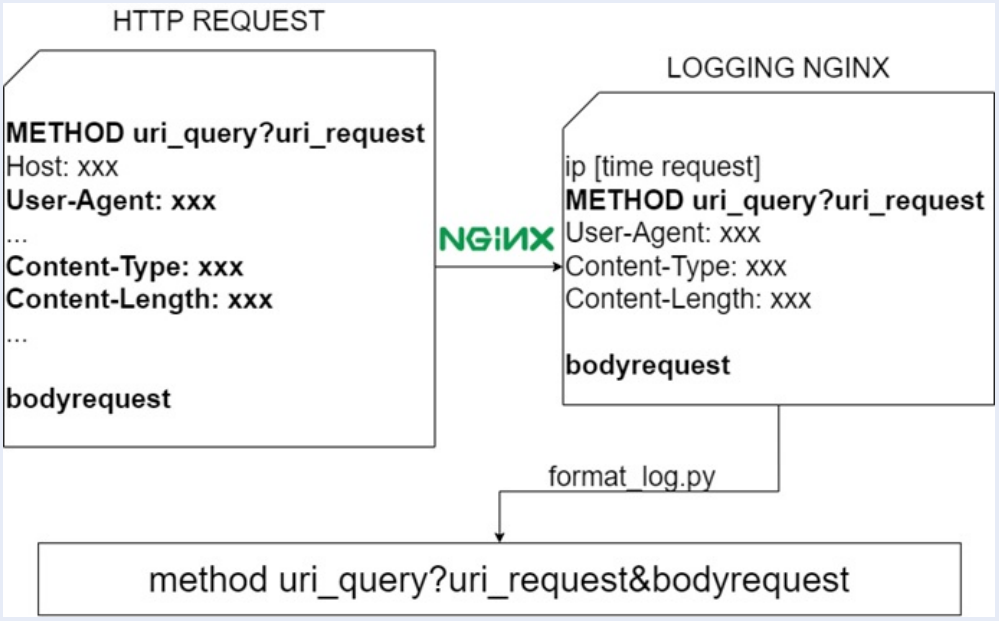

Figure 6: **The process of log formatting**.

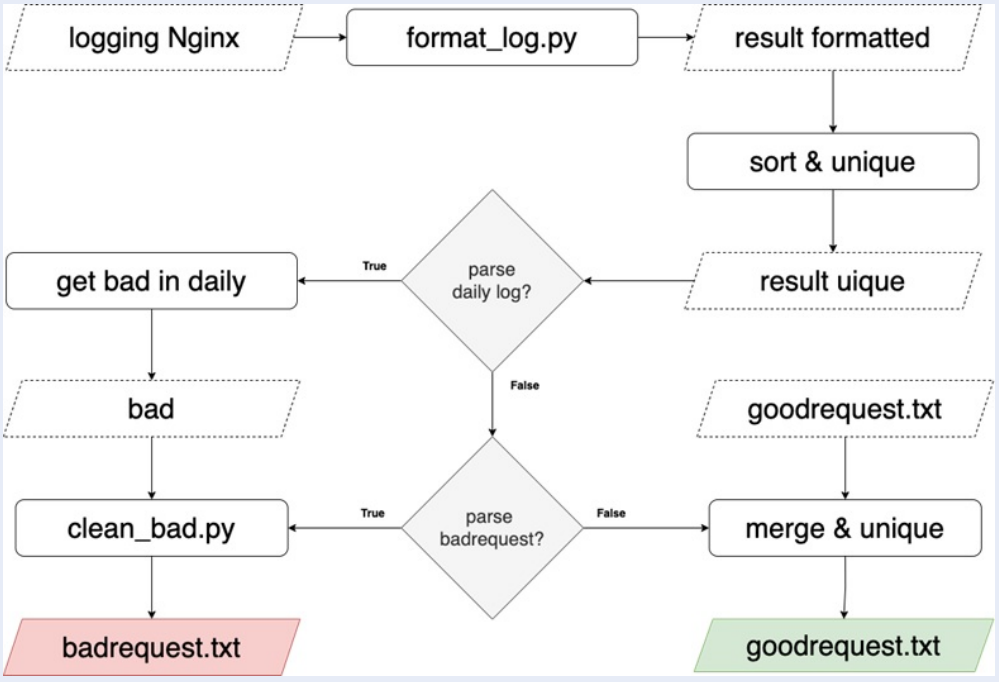

Figure 7: **The workflow of processing the access log**.

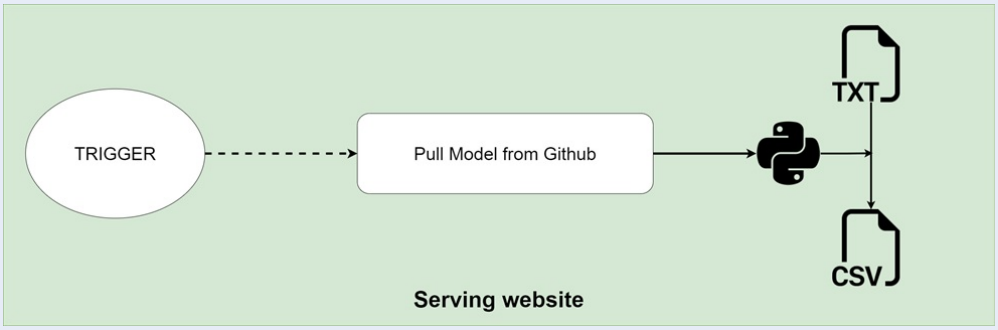

Figure 8: **Serving website for classifying requests**.

A. Dataset

In our experiment, we evaluated the proposed model on 2 datasets in the same feature extraction step. More specifically, we used the public Fwaf dataset to evaluate the effectiveness of the model. We also simulated the model on the real scenario with the dataset collected by our honeypot. We also separated these datasets into 3 parts, including 80% for training, 10% testing and 10% validating, as shown in Table 1 and Table 2. In more detail, the validating dataset is used to evaluate the performance of the model after each epoch in the training and retraining process, as shown in Figure 3. The testing dataset will re-evaluate the model after training, as shown in the test-model block of Figure 3.

Feature extraction

In the WAD scenario, we use queries or requests to detect attacks. Furthermore, we extract all records based on the TF-IDF (Term Frequency - Inverse Document Frequency) numerical statistical method. As in Figure 9, we count the frequency of each word in every request (TF) and count the number of queries/requests in which this word occurs (IDF). These words are the features of the dataset that are equal to the ratio of TF and IDF.

Simulating and collecting attack datasets in Cyber Traps (Our dataset)

As in Figure 5, to realize attacks, we built a honeypot based on 3 web applications:

- **Spring boot web application** is a well-known website built based on the Java programming language.
- **Confluence**: is a web-based corporate wiki that was created in the Java programming language and released for the first time in 2004.
- **Liferay** is a Java-based web app-location platform that provides a toolset for the development of customizable portals and websites.

Furthermore, these access logs in honeypot will be statistics and recorded by Nginx. All bad requests collected belong to XSS, SQLi, Path Traversal, RCE, and Command Injection. Next, they are processed and labeled with Bad and Good classes as in Section II.D.2. The created dataset consists of 26,020 good records and 32,757 bad records.

Furthermore, to prove the effectiveness of the MLOps pipeline in retraining the model with new data, we separated our dataset into 2 phases based on the data collection timeline of honeypots. Each phase that represents the new data update period is also split into 3 sets, including training, validating and testing sets. Detailed data are presented specifically in Table 1.

Fwaf - Web-Application Firewall dataset

Our experiments are also evaluated on the Fwaf dataset. In this dataset, all collected records contain benign traffic and the most up-to-date common attacks related to real-world scenarios. The data were labeled by two types: Bad and Good. Approximately 1,350,000 records have been captured and processed for CSV structure format. Furthermore, with the ratio of good and bad labels less than 2.5%, as shown in Table 2, the Fwaf dataset is completely a non-IID dataset.

Experimental settings

We implemented the MLOps pipeline with the TensorFlow framework. The designed CNN model and 3 ML algorithms are implemented using Keras and Scikit-learn (SKlearn). Our experiments are conducted on a Kali 2022.1 virtual machine (VM) with the CPU AMD Ryzen 5 3500U 4 cores 3.7 GHz with 8

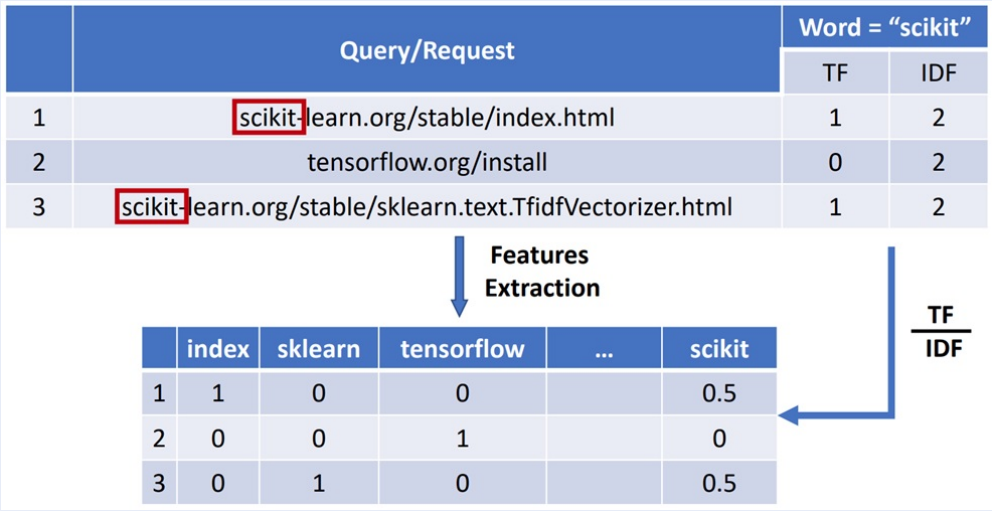

Figure 9: **Examples of query/request extraction with the TF-IDF statistical method**.

Table 1: **Number of samples in 2 phases of our dataset.**

Dataset	Phase	Labels	Number of Samples			
			Total (100%)	Train (80%)	Valid (10%)	Test (10%)
Our	1	Good	13,010	10,408	1,301	1,301
		Bad	16,375	13,100	1,637	1,638
	2	Good	13,010	10,408	1,301	1,301
		Bad	16,382	13,106	1,638	1,638

Table 2: **The number of samples in the training, testing, and validating data of the Fwaf dataset**

Dataset	Labels	Number of Samples			
		Total (100%)	Train (80%)	Valid (10%)	Test (10%)
FWAF	Good	48,126	38,500	4,813	4,813
	Bad	1,294,513	1,035,610	129,452	129,451

GB RAM and 100 GB Disk. Furthermore, the honeypot configuration is built based on Ubuntu 20.04 With Intel(R) Xeon(R) Platinum 8171 M CPU 4 cores 2.6 GHz, 50 GB Disk and 12 GB RAM. Kuberflow uses the cloud version of Arrikto's trail, which is implemented on Ubuntu Server 2020.4 with Intel Xeon E5-2660 CPU 4 cores 2.3 GHz, 60 GB Disk.

In the training process with CNN, we train the model in 2 epochs with a batch size of 128. The loss function is binary cross entropy, and the Adam optimizer is also used with a learning rate of 0.001. In addition, the ML-based attack detectors based on KNN, SVM and LR are described in Table 3.

C. Performance Metrics

In this part, 4 common metrics are used to evaluate the performance of the ML-based detection model, including accuracy, precision, recall, and F1-score.

- **Accuracy**: the results of the model to predict the correct proportion with total samples in the dataset as in Eq. (10).

$$Accuracy = \frac{n_{true}}{n_{total}} \tag{10}$$

- **Precision**: the proportion of samples identified as attacks ($attack_{true}$ + $attack_{false}$) that are indeed attacks $attack_{true}$, as in Eq. (11).

Table 3: List of hyperparameters used in KNN, SVM and LR

Algorithm	Param	Value
KNN	weights	uniform
	algorithm	auto
	leaf_size	30
	n_neighbors	3
SVM	cache_size	200
	kernel	rbf
	gamma	20
	decision_function_shape	ovr
LR	class_weight	balanced

$$Precision = \frac{attack_{true}}{attack_{true} + attack_{false}} \quad (11)$$

- **Recall** is the proportion of all attack samples ($attack_{true}$ + $normal_{false}$) correctly identified as exact types of attacks $attack_{true}$, as in Eq. (12).

$$Recall = \frac{attack_{true}}{attack_{true} + normal_{false}} \quad (12)$$

- **F1-score: This** is calculated by taking the harmonic mean of **precision** and **recall** as in Eq. (13).

$$F_1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (13)$$

D. Experimental Results

Evaluation of MLOps Pipeline with data augmentation scenario:

In this part, to evaluate the effectiveness of the MLOps pipeline in the data augmentation scenario, we completed a total of 2 experiments.

In the first experiment, we train ML models with non-IID data. Then, in the second experiment, we train the ML model with augmented data. More specifically, in this scenario, we evaluate these ML models with the non-IID Fwaf dataset.

Each experiment has the same circumstances with 5 ML models. The numerical results, which are shown in Table 4, illustrate the performance of the ML models in terms of the accuracy, precision, recall and F1-score. It can be easily seen that five types of ML models have also affected the performance of the MLOps pipeline on the dataset with augmentation, with accuracy and F1-score above 99%. However, the performances of the 5 models were significantly decreased when training on the dataset without augmentation, with a lower F1-score of 91.892%. Although the CNN model is the most effective model on the augmentation dataset, with 4 metrics of approximately 99.982%, the CNN model on the dataset without augmentation is not efficient, with precision and F1-scores of approximately 70.117% and 80.607%, respectively. In addition, when training with augmented data, the differences in performance between the CNN model and ensemble are negligible, with less than 0.003% in 4 metrics. Therefore, the ensemble model has proven to be effective and stable in both experiments, with 4 metrics greater than 91%.

Evaluation of MLOps Pipeline with Incremental Learning scenario:

In addition, to determine the effectiveness of these ML models in the first training, we also train and test these models with the dataset in phase 1. With the results shown in Table 6, it is easy to see that the performance of 5 models when trained and tested with our dataset on phase 1 has been very high with values of over 97% on 4 metrics.

In the experiment of model training without IL, when testing a large amount of new testing data in phase 2, the numerical results in Table 5 have proven that the effectiveness of these models has reduced significantly by at least 8% in 4 metrics. The results in Table 5 illustrate the effectiveness of IL in retraining the model with new data. More specifically, when testing with new data in phase 2, the performance of all retrained models reached more than 97% in all metrics.

Furthermore, with the results in Table 5 and Table 6, the Ensemble model has proven stability and effect in

Table 4: The performance of 5 ML models on Fawf

Algorithm	Without Augmentation				With Augmentation			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
SVM	0.98376	0.85145	0.85724	0.85434	0.9998	0.99979	0.99979	0.99979
LR	0.98254	0.83671	0.8489	0.84276	0.9967	0.99941	0.99716	0.99828
KNN	0.97999	0.86152	0.80265	0.82636	0.9979	0.99958	0.99687	0.99822
Ensemble	0.99096	0.91581	0.92205	0.91892	0.9998	0.99979	0.99979	0.99979
CNN	0.98113	0.70117	0. 94788	0.80607	0.9998	0.99982	0.99982	0.99982

Table 5: The performance of 5 ML models on phase 1 of our dataset

Algorithms	Accuracy	Precision	Recall	F1-Score
SVM	0.98129	0. 98474	0. 98174	0. 98324
LR	0.98469	0. 98474	0. 98775	0. 98624
KNN	0. 98809	0. 99084	0. 98783	0. 98933
Ensemble	0. 98979	0. 99084	0. 99084	0. 99084
CNN	0. 97958	0. 98168	0. 98168	0. 98168

Table 6: The performance of 5 ML models on our dataset

Algoritl	Without Incremental				With Incremental			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
SVM	0.88091	0.90842	0. 88152	0.89477	0.97652	0.98168	0.97632	0.97900
LR	0.87615	0.89867	0.88144	0.88996	0.97686	0.97985	0.97866	0.97926
KNN	0.86662	0.86203	0.89480	0.87811	0.98027	0.98352	0.98112	0.98232
Ensemb	0.90201	0.91514	0.90959	0.91236	0.98197	0.98352	0.98412	0.98382
CNN	0.86458	0. 87790	0.87897	0.87844	0.97822	0.97985	0.98105	0.98045

the context of WAD even under the condition that the model is neither retrained nor regularly updated, with Accuracy and F1-score always greater than 90%.

Moreover, to demonstrate the effectiveness of the system in terms of predictability, we also summarize some experimental evaluation results in this realistic scenario with the ensemble model. According to the comparison of Real and Predicted (Pred) Label in Table 7, all the abnormal requests are classified correctly. In particular, the 11^{th} record is one of the Confluence's latest exploits assigned as CVE-2022-26134. It has been detected by our system.

CONCLUSION

To effectively detect unknown web attacks, ML models must be trained using a large, up-to-date dataset. As such, deceptive tactics such as installing honeypots and cyber traps within the network can prove to be a valuable strategy for attracting attackers to exploit vulnerable servers. The logs and requests obtained from these deceptive resources can then be used to train the ML-based web attack detection model. The experimental results on two datasets demonstrate the feasibility and effectiveness of the proposed method for constructing high-performance machine learning models for detecting web attacks. In particular, our ensemble model also proves more efficient and stable than SVM, KNN, LR and CNN and is perfectly suitable for deployment on the Serving website. Furthermore, by utilizing triggers defined in the MLOps pipeline, these models can be regularly updated with new indicators identified in the cyber trap system, enabling them to recognize harmful requests directed toward web servers. Our experiments involving the setup of a web honeypot, an MLOps pipeline, and evaluations on two datasets demonstrate the benefits of proactive, deceptive, and ongoing approaches in the realm of web attack detection.

Table 7: The experimental evaluation results

#	Request	Labels	
		Real	Pred
1	post/rest/tinymce/l/content/65628 /comment?actions=true&html=l <set/xmlns='urn:schemas-mic...	Bad	Bad
2	post/pages/doenterpagevariables .action?querystring=\\ u0027+{ class.forname(\\u0027javax. script...	Bad	Bad
3	get/admin	Good	Good
4	get/images/-min.jpg	Good	Good
5	get/admin/browseshortcuts.action	Good	Good
6	get/c/portal/logout	Good	Good
7	post/rest/tinymce/l/content/65628 /comment?actions=true&html= <a href="javas\\x02cript:javasc...	Bad	Bad
8	get/c/portal/layout	Good	Good
9	get/	Good	Good
10	get/images/bg-01.jpg	Good	Good
11	get/${(#a=@org.apache.commons ,io.IOUtils@toString(@java.lang. Runtime@getRuntime().exec("ic..	Bad	Bad
12	post/api/jsonws/invoke?cmd={"/ expandocolumn/add-column":{}} &p auth=o3lt8qlf&formdate=lE...	Bad	Bad
13	get/css/edit-info.css	Good	Good
14	get/css/buy-product.css	Good	Good
15	get/images/icons/avatar group 48.png	Good	Good

ABBREVIATIONS

CGAN: conditional generative adversarial network

CNN: convolutional neural network

DR: detection rate

KNN: K-nearest neighbors

LR: logistic regression

LSTM: long short-term memory

ML: Machine learning

MLOps: ML operations

non-IID: nonindependent and identically distributed

RNNs: recurrent neural networks

SVM: support vector machine

TF-IDF: Term Frequency - Inverse Document Frequency

VM: virtual machine

WAD: website attack detection

ACKNOWLEDGMENTS

None

AUTHOR'S CONTRIBUTIONS

Van-Hau Pham and Nghi Hoang Khoa engaged in a discussion and put forth the key concepts of this research. Nguyen Huu Quyen and Van-Hau Pham carried out the experiments and composed the paper. Phan The Duy discusses the ideas and contributed to writing the paper. All authors discussed and edited the manuscript.

All authors read and approved the final manuscript.

FUNDING

None

COMPETING INTERESTS

The authors declare that they have no competing interests.

REFERENCES

1. What Is a Web Application Attack and how to Defend Against It [Internet];Available from: https://www.acunetix.com/websitesecurity/web-application-attack/.
2. Lima Filho FS de, Silveira FAF, de Medeiros Brito Junior A, Vargas-Solar G, Silveira LF. Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning. Secur Commun Netw. 2019 Oct 13;2019:1-15;Available from: https://doi.org/10.1155/2019/1574749.
3. Liang J, Zhao W, Ye W. Anomaly Based Web Attack Detection: A Deep Learning Approach. In: Proceedings of the 2017 VI International Conference on Network, Communication and Computing - ICNCC 2017 [Internet]. Kunming, China: ACM Press; 2017 [cited 2023 Jan 6]. p. 80-5;Available from: https://doi.org/10.1145/3171592.3171594.
4. Zhang Y, Lu J, Jin S. Web Attack Detection Based on User Behaviour Semantics. In: Qiu M, editor. Algorithms and Architectures for Parallel Processing [Internet]. Cham: Springer International Publishing; 2020 [cited 2023 Jan 6]. p. 459-74. (Lecture Notes in Computer Science; vol. 12454);Available from: https://doi.org/10.1007/978-3-030-60248-2_31.
5. Stevanović N, Todorović B, Todorović V. Web attack detection based on traps. Appl Intell. 2022 Sep;52(11):12397-421;Available from: https://doi.org/10.1007/s10489-021-03077-9.
6. Han X, Kheir N, Balzarotti D. Evaluation of Deception-Based Web Attacks Detection. In: Proceedings of the 2017 Workshop on Moving Target Defense [Internet]. Dallas Texas USA: ACM; 2017 [cited 2023 Jan 6]. p. 65-73;Available from: https://doi.org/10.1145/3140549.3140555.
7. Testi M, Ballabio M, Frontoni E, Iannello G, Moccia S, Soda P, et al. MLOps: A Taxonomy and a Methodology. IEEE Access. 2022;10:63606-18;Available from: https://doi.org/10.1109/ACCESS.2022.3181730.
8. Mäkinen S, Skogström H, Laaksonen E, Mikkonen T. Who Needs MLOps: What Data Scientists Seek to Accomplish and How Can MLOps Help? 2021 [cited 2023 Jan 6];Available from: https://doi.org/10.1109/WAIN52551.2021.00024.
9. Pölöskei I. MLOps approach in the cloud-native data pipeline design. Acta Tech Jaurinensis. 2021 Apr 9;15(1):1-6;Available from: https://doi.org/10.14513/actatechjaur.00581.
10. Renggli C, Rimanic L, Gürel NM, Karlaš B, Wu W, Zhang C. A Data Quality-Driven View of MLOps. 2021 [cited 2023 Jan 6];Available from: https://arxiv.org/abs/2102.07750.
11. Muralidhar N, Muthiah S, Butler P, Jain M, Yu Y, Burne K, et al. Using AntiPatterns to avoid MLOps Mistakes. 2021 [cited 2023 Jan 6];Available from: https://arxiv.org/abs/2107.00079.
12. Moreschini S, Lomio F, Hastbacka D, Taibi D. MLOps for evolvable AI intensive software systems. In: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) [Internet]. Honolulu, HI, USA: IEEE 2022 [cited 2023 Jan 6]. p. 1293-4;Available from: https://doi.org/10.1109/SANER53432.2022.00155.
13. Garg S, Pundir P, Rathee G, Gupta PK, Garg S, Ahlawat S. On Continuous Integration/Continuous Delivery for Automated Deployment of Machine Learning Models using MLOps. In: 2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE) [Internet]. Laguna Hills, CA, USA: IEEE; 2021 [cited 2023 Jan 6];Available from: https://ieeexplore.ieee.org/document/9723793/.
14. Pham NT, Foo E, Suriadi S, Jeffrey H, Lahza HFM. Improving performance of intrusion detection system using ensemble methods and feature selection. In: Proceedings of the Australasian Computer Science Week Multiconference [Internet]. Brisband Queensland Australia: ACM; 2018 [cited 2023 Jan 6]. p. 1-6;Available from: https://doi.org/10.1145/3167918.3167951.