

Prevention of Webshell Attack using Machine Learning Techniques

Satish Y C¹, Praveen M Naik² and Dr. Bhawana Rudra³

¹⁻³Department of Information Technology, National Institute of Technology Karnataka Surathkal, Mangaluru, INDIA--575025

Email: satishyc@hotmail.com, mr.praveen.m.naik@gmail.com, bhawanarudra@nitk.edu.in

Abstract—Webshell is a web vulnerability and a security threat to any user or a server that can be accessed by attackers to control our system. And also, they may use our system as a command control device to attack other systems. It is difficult to monitor and identify such threats because attackers always tried to attack in different methods and new technologies. However, we can detect the webshell with Machine Learning Techniques with better accuracy; all we need is more number of samples. With this project, we presented a PHP based webshell detecting model. We used different ML algorithms: Logistic Regression(LR), Random Forest(RF), Support Vector Machine(SVM) and K-Nearest Neighbour(KNN). Addition to this PHP file's standard statistical features, we also added an opcode sequence from the PHP files, consisting of the TF-IDF Vector and the Hash Vector. Depending upon these features, we trained with different machine learning models(SVM, RF, LR, KNN). In these models, we got better results with Random Forest having an accuracy of 96.45% with a false-positive rate of 3.5%, which is good results compared to several popular detection techniques.

Index Terms— Machine Learning; Logistic Regression; Random Forest; Support Vector Machine; K-Nearest Neighbour.

I. INTRODUCTION

In today's scenario, web apps have been used widely, and it plays a vital role in current society, such as e-commerce websites, news portals, social media, and banking so on. Despite that, web attacks truly impact web security in different manners. Webshell is a web vulnerability and a security threat to any user or a server that can be accessed by attackers to control our system. It is going to be a severe threat to all users as a result of webshell providing easy access for hackers. They can tamper or steal the information stored in the database, monitoring web activity, installing malicious files into servers, changing source code, spreading computer worm, and exploiting web vulnerabilities, and so on.

Web apps have developed in languages like PHP, Active Server Pages(ASP), Java Server Pages(JSP), Perl, Ruby, Python, etc. We have mainly centered on PHP-based applications, and popular websites like Facebook, Yahoo, Wikipedia, Wordpress, etc. are developed using PHP[1]. Webshell furthermore supports to control the system used as a command control device to attack other systems. It is beneficial in building a monitoring application in the web application domain. Still, this is also correct that hackers always intend to create indirect access. Hackers used to attack web servers; one of the popular ways of webshell is to run commands, browse

files, upload files, raise benefits, and turn to different targets. Therefore, the character of a web server administration is essential to safeguard the security and privacy of all web pages. The webpage contains thousands of line codes, or in some cases, it is more than that. It is tough to monitor or to verify the code manually to detect webshell. Hence, detecting and identifying the webshell inside a web page is a tough job.

There exist a few familiar tools to identify the webshell. Namely: There is a UNIX command called Grep to identify whether a given string or a regular expression is there or not. NeoPI[2], which is having the advantage to detect, find the complicated along with encrypted content inside text files along with scripts by measuring and registering a few indices like Longest Word, Entropy, Coincidence, Signature; subsequently, this is going to return a position for every record and top-10 documents among the most elevated position: Yet, it has a disadvantage which doesn't produce a threshold value concerning indices to identify whether the given file is webshell infected or not. PHP Shell Detector[3] primarily is a known webshell signature database stored in a web server. It is useful in detecting known webshell, but if a new webshell is discovered, which is not updated in the database, it is hard to find these kinds of webshells.

However, numerous NGOs and Government organizations are trying to inform the importance of information security, and these types of attacks are happening all over the world endlessly. In recent times[4], a firm in the Government sector has identified a misconfiguration in one of their internet-facing servers. It permitted attackers to upload their web shell; with this, the opponent gained hold for further compromise. Then the organization appointed Microsoft's Detection and Response Team (DART) to handle the full incident rejoinder and rectify the problem sooner it could create additional harm. DART's examination indicates that the hackers inserted a webshell script in different directories at a web server, directing them to further steal the details of user account details. This information allowed the hackers to conduct monitoring using net.exe, search for new target systems using nbtstat.exe, and ultimately move alongside using PsExec.WSO Web Shell[5] "WSO" seemingly abbreviated as "Web Shell by orb," a note that sometimes occurs in WSO source code. The WSO webshell provides administration access to those who know its URL. It's also seemingly known as the FilesMan backdoor, although it's a good deal more than a little backdoor. WSO needs a password in addition to more access. Passwords are sent on to different WSO instances: '45000', 'nzerf', 'aspx22', 'FDY27c', 'gatamorgfana' The password has run over md5() and analyzed with string literal in the original code. These types of string literals often appear on the web: searching on the web for any of them get corresponding plain-text and password.

Although the increase in the usage of technologies like artificial intelligence and ML algorithms looked continuously begun to be utilized in the field of cybersecurity for the managing of security concerns like brute force login attacks, intrusion detection systems, malicious third-party apps, sensitive data leak, smart firewalls, and so on. Machine learning is going to help to detect the webshell when they come up with enough samples and appropriate features. Moreover, to improve the detecting performance, we also need to choose the relevant machine learning algorithm. Consequently, my project mainly focuses on the improvement of machine learning classification algorithms and webshell useful futures. We refer to a few related research on webshell detection through machine learning; also, we analyzed their benefits and drawbacks in detail. Subsequently, our work is on detecting PHP webshell through various machine learning models, which are Support Vector Machine(SVM), Random Forest(RF), Logistic Regression(LR) and K-Nearest Neighbour(KNN). Besides, we added many practical features: Information Entropy[6], Index of Coincidence[7], usage of eval() function, signatures matched, data compression ratio, text features of opcode like the TF-IDF Vector and the Hash Vector. Finally, we have created the dataset then trained with different machine learning classification algorithms. We got a good result for the random forest with an accuracy of 96.45%, and the false-positive rate is 3.5%.

The arrangement of the rest of the chapters of this project report follows. Chapter 2 represents related work. Chapter 3 explains the dataset and feature extraction. Chapter 4 introduces implementations with machine learning algorithms. Chapter 5 is having details about the experiment and its results. Finally, Chapter 6 will have details of the conclusion and future work.

II. RELATED WORK

We summarize the appropriate outcomes from similar work about the detection of webshell and point out contrasts with our model. B. Hu [8] Their specialty is the detection of the PHP webshell. They also chose similar features that we are also using, like they used some statistical features, which are an index of coincidence, entropy, data compression ratio, etc. They have selected a few samples in their model, i.e., only 600 samples; These samples are very less; with this, we can't rely on their research model. And features what they have chosen surely we can't incorporate the PHP webshell exactly, i.e This kind of features also helpful in detecting other types of webshells, like ASP or JSP webshells. J. Hu, Z. Xu, D. Ma, J. Yang[9] They have used a decision tree to

detect the PHP webshell, and they have used some necessary attributes like counting the number of words, lines, and named functions. We have much question mark whether using those features will be essential or helpful in training our predictive model. D. Truong, G. Cheng, X. Guo and W. Pan[10] They have mostly fixed static features to detect the webshell. These static functions do not help identify because if an attacker used the pattern already mentioned in the static feature, only we can identify else we can't.

W. Zhu, Y. Hu[11] They have used SVM for the detection of webshell. However, they have chosen fewer statistical features such as length of the most significant word, index of coincidence, data compression ratio, entropy, so on. However, the NN-SVM model demonstrates great generalization ability due to the SVM's upgrading. NN-SVM (NearestNeighborSupportVector Machine) needs to prune the training set depending on the distinction among each sample also its closest acquaintance group. Selected features are not sufficient to detect web shell. T. Li, C. Ren, Y. Fu, J. Xu, J. Guo and X. Chen, [12] Their research work characterizes the conventional hidden means and hazards of the webshells, and they have examined the techniques of traditional ML algorithm detection techniques. Conventional ML algorithms primarily utilize features of the word plus static features to detect and classify. Despite that, these methods have some drawbacks, like incomplete feature lexicon, bypassing feature lexicon, and the outcome of detection adequately on the extraction of feature words. H. Zhang et al.[13] They have figured out webshell attacks through web-traffic and tried to obtain anonymous webshell-attacks. They have come up with a character based feature extraction method for the sequential data. This model obtains input in the form of a feature vector to the combined model of LSTM and CNN. They have not analyzed the Webshell behavior pattern. F. M. M. Mokbal, W. Dan, A. Imran, L. Jiuchuan, F. Akhtar and W. Xiaoxi,[14] They have proposed an ANN-based model to identify cross-site scripting(XSS) based web-attacks. They have conducted experiments effectively on test data and compared with existing methods. They Have not tested in real time environment. Om Kumar, C.U., Sathia Bhama, P.R.K.[15] They have proposed a method to identify the Distributed Denial of Service(DDoS) using machine learning techniques. They have implemented using many supervised ML algorithms. Despite that, those techniques need comprehensive network expertise to choose relevant features from the dataset. Typically, they remain restricted to only one or some DDoS vectors. Besides, they need frequent updates of the system to retain it running in distinct conditions. K. Kamtuo and C. Soomlek[16] They have introduced the framework of preventing the SQL injection attack in different SQL Queries, which are mainly used in server-side scripting by using compiler platform and ML algorithm techniques. Selected dataset is very small.

Li, Q., Li, W., Wang, J., & Cheng, M.[17] This method is based on classical ML, which is challenging to process various and irrelevant features. In contrast, the methods built on deep learning have multiple hyper-parameters, and it's obvious to let overfitting. Their implementation is only for a dataset not for a real time environment and they have not analyzed the behaviour of the webshell attack.

III. DATASET AND FEATURE EXTRACTION

A. Dataset

We have downloaded several projects from the web, and this includes webshell projects and Content Management System(CMS). They have malicious and non-malicious examples. To maintain the steadiness of the webshell infected and non-webshell infected, we took it almost equally, i.e., 4900 webshell samples and 5100 non-webshell samples collected through several Github resources. Some of them are mentioned in the below table. We have not used every sample in which we downloaded specific samples we couldn't extract. Table I has details of dataset repositories.

B. Feature Extraction

In this part, we've defined the distinctive introduction of each decided feature used in our examination. Firstly we have extracted standard features of the PHP files, which comprise an index of coincidence, entropy, size of the longest word, the ratio of data compress, etc. along with certain standard features, we got a 6-dimensional vector which has features.

Still, certain features are simple to compile, which are useful in training to predict the model. Feature vectors dimension comparatively low, therefore we might not get an exceptional prediction model, and with these small features, we can't decide our model performance will be excellent. Also, some valid information in the code, such as annotations, will also add the calculations of certain features; it may commence lowering classification effects.

So, we have extracted the operational code(i.e., opcode) sequence from every sample. We got 2 necessary text features of the opcode, the TF-IDF (Term Frequency-Inverse Document Frequency) vector, and Hash Vector.

TABLE I. COLLECTED SAMPLES LINKS

Sample	Source
Webshell	https://github.com/JohnTroony/php-webshells
	https://github.com/xl7dev/WebShell
	https://github.com/ysrc/webshell-sample
	https://github.com/tdifg/WebShell
	https://github.com/tennc/-WebShell
	https://github.com/tanjiti/-WebShell-Sample
	https://github.com/JohnTroony/-PHP-WebShells
CMS	https://github.com/doorgets/CMS
	https://github.com/croogo/croogo
	https://github.com/craftcms/cms
	https://github.com/johnshen/-PHPcms

Firstly we tried with different dimensions of TF-IDF and Hash Vector. According to our observations, we got a good result when choosing 146 dimensions of the TF-IDF vector and 200 dimensions of Hash Vector. Along with this, we will have a 352 dimension input vector.

C. Information Entropy

Information Entropy is an area of study involved with comparing data for communication. It is also a subfield of mathematics, and it is involved in topics like data compression and limits of signal processing. This concept was developed and proposed by Claude Shannon[18] at Bell Labs. Entropy involves the representation of data compactly and transforming and storing it in a robust manner, which is robust to the errors.

Calculating the information for an event

Calculating the information is the basic building block in the field of information entropy. Instinct beyond the calculation of data is the concept of estimating how much astonishment there in an event. These events which are not common(lower probability) remains astonished hence they have more information for those events which are common(higher probability)

Rare events remain vague or higher astonishing, and it needs higher information to present them in current events. If we want to determine the amount of information of any event with utilizing the probability of an event we call it as "Shannon Information," we also call this one as "information," and we can calculate for any discrete event y as follows:

$$\text{Information}(Y) = -\log(p(y))$$

Here $\log()$ is log base-2, and $p(y)$ is the probability of event y .

The reason behind the selection of log base-2 is the measuring of information is in bits. We can explain this in the information processing insight as to the number of bits required for the representation of an event.

The information entropy calculated as $k()$;

$$k(y) = -\log(p(y))$$

D. Index of Coincidence

Index of Coincidence(IC) can be calculated using the probability that whenever we pick two letters from text without any text replacement, then that two letters are the same. Every random part of the text with every letter has a probability of $1/26$, and then IC also is $1/26(0.0385)$.

If we know the frequency of letters, and the sum of all frequencies equal to one. Then we can use this formula to calculate the IC of any language.

$$I.C = \sum \{i = A\} \{i = Z\} (Fi)^2$$

F_i is the frequency of a letter in the text of any language.

E. Length of the longest word

There are some proofs for any encrypted webshell; we may contain some unusual words or strings with considerable length and are unreadable. So, by calculating the longest word's length in a script, we can conclude whether it is a webshell infected or not.

F. Count of Matched Signatures

Webshell holds some strange signatures that accidentally occur in regular files; these kinds of signatures have been mentioned in the below table. We will be able to calculate the occurrence of these signatures and their frequencies, and we list out this; it is going to identify the webshell. All these signatures are mentioned in Table II.

G. Data Compression Ratio

The data compression ratio estimates the relative decrease in the size of the information description generated by an information compression algorithm. It is commonly expressed as the division of uncompressed size by compressed size.

$$\text{Compression Ratio} = \frac{\text{Uncompressed Size}}{\text{Compressed Size}}$$

H. Use of eval()

The eval() is a kind of 1-word trojan, which is a variety of the webshell in a little volume, eval() is common in the formation of @eval(\$post[xxxxx]). So, eval() frequency can add to the dataset in the method of ML algorithm training.

TABLE II. LIST OF SIGNATURES TO BE MATCHED

Signature
assert
base64 decode
eval
exec
file_put_contents
passthru
pcntl
popen
proc_open
python_eval
shell
system

I. Text features of opcode sequence

Operational code, also known as opcode, is part of the instruction to a machine code, which is utilized to indicate what all the operations need to perform[19]. It is more effective to extract a few textual content functions from the opcode sequence than at once extracted via PHP source code due to the fact that opcode can act as filters to a few noises in PHP source code, like a few annotations. In our evaluation, we usually extracted the opcode sequences of all accumulated samples. After this, we extract two kinds of textual content features from the opcode sequence via Scikit-learn, which is the TF-IDF vector and the Hash vector, respectively. These pairs of vectors have distinctive policies and have their advantages. So, if we combine all of these features, our predictive model will be more effective.

TF-IDF Vector

Counting words are a useful beginning point, though it is fundamental. There are several issues with simple counts. If we take a few words like "these," it will have appeared many times in the text, and there will be a large number of occurrences in the text, which is not meaningful to encode in the vectors.

- **Term Frequency:** The number of times the mentioned word appears in any document.
- **Inverse Document Frequency:** This one reduces the words that look a lot over the documents.

TF-IDF is a word frequency score that tries to focus more on new words like frequently occur in a document but not in the entire document. TF-IDF is the outcome of TF and IDF. TF-IDF opcode vector flow can be a useful feature based on a fact that, if we have the frequency of opcode samples in different variations like either the given frequency of opcode samples is vast or flat, these kinds of the opcode will assist us rightly to identify among all other samples. Hence, the opcode of the TF-IDF vector will help predict the model to achieve better accuracy.

Hash Vector

Frequencies and counts will be helpful; however, one shortcoming of these types is the vocabulary will turn to large. In this case, it requires a significant vector to encode the documents and requires vast memory, and it will also slow down the algorithms.

An inventive workaround is the use of one-way hashing of words to convert them to integers. The amusing part is that vocabulary is not required, and you can choose a random-long fixed length of a vector. The Drawback is Hashing is a one-way function, so there is no method to get back to original. We can call this hashing trick; it determines that it can utilize the hashing function to define the eigenvectors index position; hence we don't need to create an extensive dictionary.

IV. IMPLEMENTATION WITH MACHINE LEARNING MODELS

Machine learning algorithms usually comes with two categories: supervised learning and unsupervised learning[20]. Supervised algorithms consider the function that describes a set of the input features to labels (also called “targets”). And it is directly associated with prediction tasks. Unsupervised learning which not require labels and reviews internal patterns or structures inside a dataset. We chose Supervised learning to train our model, where we train our model with the labeled dataset. It suggests a few data already marked with the proper solution. It can be used to learn which happens in the familiarity of a supervisor or a master. Supervised ML algorithm learns from the well-labeled dataset for training, and it also supports you in predicting with abnormal input data. For successful developing, balancing, and expanding the right supervised ML algorithm in data science, we require a reasonable amount of time and technical expertise in the field of data science. The training data is trained by using four different ML algorithms. i.e., Random Forest, K-Nearest Neighbour, Logistic Regression, and Support Vector Machine. Each algorithm explained in detail.

A. Random Forest

It is a supervised ML algorithm technique utilized for this pair of classification and regression. Though, it is most of the time used in classification problems only. As we are aware that the forest is formed of trees, and if we have more trees, the forest density is more; it implies a healthy forest. This random forest ML algorithm creates many decision trees that are based on data samples and obtains predictions from each tree; then, it finally selects the best solution through voting, which we call the ensemble voting classifier. Because it is more reliable than a single decision tree, and it also reduces the over-fitting by averaging the result.

B. Logistic Regression

It is a supervised ML algorithm and utilized to forecast target variable probability. The characteristic of the target variable is dichotomous, and this means only two classes are possible. In simple terms, the target variable holds binary data, either 1 or 0 (which implies failure or success). In terms of mathematics, we can derive the logistic regression model that predicts $P(Y=1)$ as a function of Y . It is one of the better and simple ML algorithm, which is used in many classification problems, and used in many fields like cybersecurity, medical, etc.

C. Support Vector Machine(SVM)

SVMs are a more robust but adaptable supervised machine learning algorithm which is used for both regression and classification. In general, they are used for classification problems. Recently, SVMs are trendy due to their capacity to supervise multiple categorical and continuous data. SVM algorithm works on the notion of pickup a hyperplane which best and divides the data-set into the two separate classes.

D. K-Nearest Neighbour(KNN)

KNN is a supervised ML algorithm that can be utilized for both classifications and predictive regression challenges. Still, it is primarily applied for the classification of the predictive problems in the application. The following two sections describe more about KNN –Lazy learning algorithm – It is a lazy learning algorithm because it doesn't hold a specialized training phase, and it uses all the training data during classification.

Non-parametric learning algorithm – It is also a non-parametric learning algorithm because it does not find anything regarding the underlying data.

F. Detailed Training Process

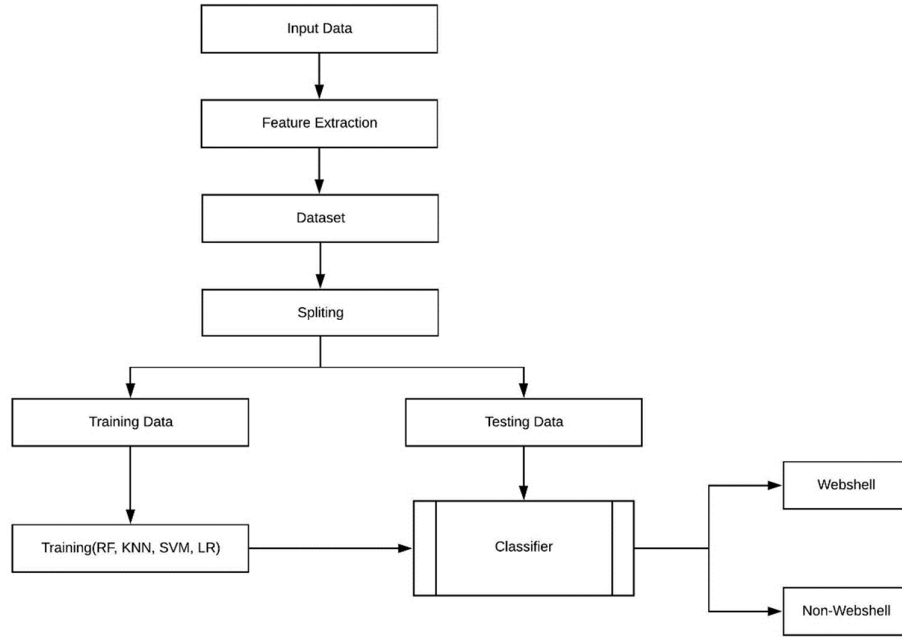


Figure 1: Detailed Training Process

Input Data: Data is collected from webshell files through various sources, which includes webshell and non-webshell.

Feature Extraction: Extract all statistical features and all opcode sequences from the input data.

Dataset: All extracted features stored in CSV files with labels like webshell infected and webshell non-infected.

Splitting: The dataset divided into two parts. One part is for testing another for training in the ratio of 9:1, respectively.

Training: We used different machine learning algorithms for training, which we already mentioned earlier.

V. EXPERIMENT AND RESULTS

A. Evaluation Method

We Chose Cross-validation [21] does a resampling method applied to evaluate ML models on a smaller dataset. The process has a single parameter named as K, which means how many sets we are splitting from the dataset. This is the reason we call it k-fold-cross-validation. If we choose our K value as 5, then it becomes 5-fold-cross-validation. In general, we use 4 parts for training, and 1 part is for testing.

Cross-fold-validation is initially employed in applied ML to evaluate the ability of an ML model against hidden data. This is to use a short sample to predict how the model is presumed to work in common when utilized to obtain predictions on data not applied while the model's training.

This is a standard practice because it is simple to learn. Compared to other models, it is less optimistic; also, we call it less biased. It has a simple train/test split.

The overall technique is as follows:

- Rearrange the dataset randomly.
- Divide dataset into K groups.
- For every different group:
 - Use a certain number of groups of your choice for the testing.
 - Then remaining groups will be used for training.
 - Train the model using a training dataset and evaluate it using the test dataset.
 - Hold the evaluation score and drop the model.
- Review the ability of the model by applying the sample of model evaluation scores.

We evaluated by outlining ROC (ReceiverOperating Characteristic) Curve [22] in our analysis. And we took maximum advantage by choosing many values to confer our evaluation: precision, F1-measure, accuracy, recall, FPR (False Positive Rate), and AUC (Area Under ROC) [23]. We briefly explained these predictions below.

The first thing we notice here is the ROC curve. We can define if our ROC curve is right or not by viewing the AUC(Area Under the Curve) along with additional parameters of the confusion matrix. A confusion matrix is a table that usually utilized to describe a classification model's performance on a collection of testing datasets for which the actual values are identified.

True Positive and True Negatives are the rightly predicted remarks and hence displayed in the green color. For an ideal model, False Positives and False Negatives need to be less, so we need to reduce it; that's why they are displayed in red color. So let's define all these terms in detail.

True Positive (TP) The actual class is "Yes," and predicted class is "Yes," then this kind of prediction we call it has True Positive.

True Negative (TN) The actual class is "No," and predicted class is "No," then this kind of prediction we call it has True Negative.

False Positive (FP) The actual class is "No," and predicted class is "Yes," then this kind of prediction we call it has False Positive.

False Negative (FN) The actual class is "Yes," and predicted class is "No," then this kind of prediction we call it has False Negative.

B. Experiment Results

The precise training procedure of our model is explained in Chapter 4. To start with results we experimented with different dimensions of input size, firstly we trained with six statistical features with various machine learning algorithms, then examined the TF-IDF vector values and Hash vector values separately trained with machine learning algorithms, and finally, we combined all statistical features and top keywords in TF-IDF vector and Hash vector where we got good results for Random Forest algorithm with 96.5%. All remaining details have been mentioned in the Table III.

TABLE III. EVALUATION DETAILS

Feature	Algorithm	Accuracy	Precision	Recall	F1	FPR
6-dimension statistical features	RF	92.36%	92.78%	91.77%	92.45%	6.8%
146-dimension TF-IDF vector	RF	91.76%	92.38%	91.07%	91.85%	7.1%
200-dimension hash vector	RF	93.55%	93.78%	92.07%	93.27%	6.4%
All extracted Features	RF	96.55%	96.27%	97.18%	96.72%	3.55%
All extracted Features	SVM	88.91%	92.09%	87.61%	89.79%	9.44%
All extracted Features	LR	87.93%	91.62%	86.4%	88.93%	10.11%
All extracted Features	KNN	91.13%	95.34%	88.74%	91.92%	5.71%

From the Figure 2, we plotted a graph for TPR vs. FPR; we compared the Random Forest algorithm with 6 statistical features and all extracted features, which includes top keywords from TF-IDF and Hash Vectors. In the above graph, AUC of 6 statistical features will be lesser than all extracted features. This shows that it is inefficient to compare with other one.

C. Comparison with other models

We have downloaded several webshell projects from the internet and compared them with our model and Table IV shows the comparison.

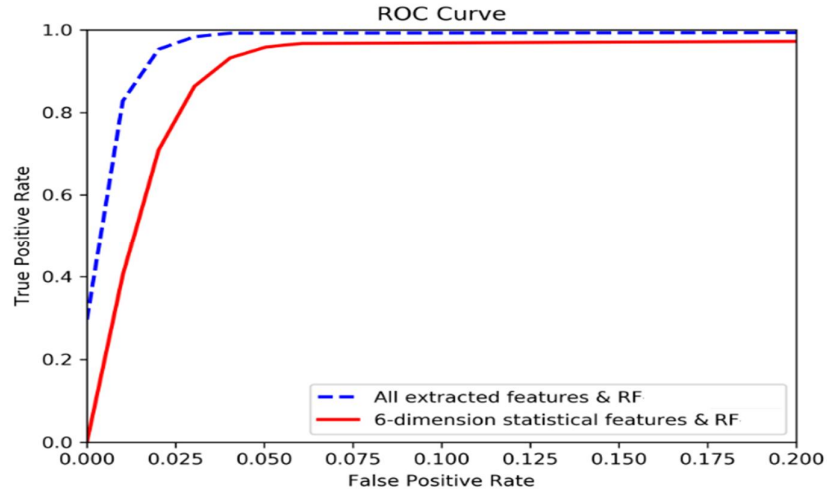


Figure 2 : Roc Curve Comparisons

TABLE IV. COMPARISON WITH OTHER MODELS

Detector	Version	Accuracy
Our Model RF	--	96.55%
php-malware-finder	--	69.606%
D Shield	V2.09	92.11%
Webshell Killer	V3.3.0.2	74.942%
Web Shell Detector	V1.1 in Python	83.991%

VI. CONCLUSION AND FUTURE WORK

In this project, we examined different machine learning algorithms like Random Forest, Logistic Regression, Support Vector Machine, and K-Nearest Neighbours. And also, we have approached different methods like statistical features, opcode sequences. We combined this both where we got better results for the Random Forest Algorithm with 96.55% accuracy and 3.5% of false positive rate and 98.7% of AUC. This shows better, among other algorithms. As future work, there are different attacks which are also helpful to insert webshell for that we can combine to write a single application to detect the attack.

REFERENCES

- [1] Top 10 Php Based Popular Websites Over Internet <https://www.w3trainingschool.com/top-10-php-based-popular-websites-internet> (2020). Accessed 15th June 2020
- [2] Scott, B. and H. Ben, Web Shell Detection Using NeoPI.[R/OL]: <http://resources.infosecinstitute.com/~web-shell-detection>
- [3] Luczko, P., PHP Shell Detector [R/OL]: <https://github.com/emposha/PHP-Shell-Detector>
- [4] Ghost in the shell: Investigating web shell attacks <https://www.microsoft.com/security/blog/2020/02/04/ghost-in-the-shell-investigating-web-shell-attacks> (2020). Accessed 15th June 2020
- [5] PHParasites: WordPress Malware <http://www.stratigery.com/phparasites/wso.html> (2013). Accessed 15th June 2020
- [6] C. E. Shannon, "A mathematical theory of communication," Bell System Technical Journal, vol. 27, no. 3, pp. 379-423, 1948.
- [7] W.F. Friedman, "The index of coincidence and its applications in cryptology," Department of Ciphers. Publ 22. Geneva, Illinois, USA:Riverbank Laboratories, 1922.
- [8] B. Hu, Research on webshell detection method based on bayesian theory, Science Mosaic, vol. 6, pp. 66-70, 2016
- [9] J. Hu, Z. Xu, D. Ma, J. Yang, Research of webshell detection based on decision tree, Journal of Network New Media, vol. 6, pp. 15-19, Jan.2012.
- [10] D. Truong, G. Cheng, X. Guo and W. Pan, Webshell detection tech-niques in web applications, in Fifth International Conference on Com-puting, Communi-cations and Networking Technologies, Hefei, 2014, pp.1-7.
- [11] W. Zhu, Y. Hu, Webshell detection based on NN-SVM, Communica-tion In-formation Technology, vol. 2, pp. 55-58, 2015.

- [12] T. Li, C. Ren, Y. Fu, J. Xu, J. Guo and X. Chen, "Webshell Detection Based on the Word Attention Mechanism," in IEEE Access, vol. 7, pp. 185140-185147, 2019.
- [13] H. Zhang et al., "Webshell Traffic Detection With Character-Level Features Based on Deep Learning," in IEEE Access, vol. 6, pp. 75268-75277, 2018.
- [14] F. M. M. Mokbal, W. Dan, A. Imran, L. Jiuchuan, F. Akhtar and W. Xiaoxi, "MLPXSS: An Integrated XSS-Based Attack Detection Scheme in Web Applications Using Multilayer Perceptron Technique," in IEEE Access, vol. 7, pp. 100567-100580, 2019.
- [15] Om Kumar, C.U., Sathia Bhama, P.R.K. Detecting and confronting flash attacks from IoT botnets. J Supercomput 75, 8312–8338 (2019). <https://doi.org/10.1007/s11227-019-03005-2>
- [16] K. Kamtuo and C. Soomlek, "Machine Learning for SQL injection prevention on server-side scripting," 2016 International Computer Science and Engineering Conference (ICSEC), Chiang Mai, 2016, pp. 1-6
- [17] Li, Q., Li, W., Wang, J., & Cheng, M. (2019). A SQL Injection Detection Method based on Adaptive Deep Forest. IEEE Access, 1–1. doi:10.1109/access.2019.2944951
- [18] Claude Shannon https://en.wikipedia.org/wiki/Claude_Shannon
- [19] D. Sklar, "Understanding PHP internals," Essential PHP Tools: Modules, Extensions, and Accelerators, Berkeley, CA: Apress, 2004, PP. 265-274.
- [20] C. M. Bishop and N. M. Nasrabadi, Pattern Recognition and Machine Learning. New York, NY, USA: Springer-Verlag, 2006.
- [21] StratifiedKFold v.s KFold v.s StratifiedShuffleSplit \url{https://medium.com/@xzz201920/stratifiedkfold-v-s-kfold-v-s-stratifiedshufflesplit-ffcae5bdfd} (2020). Accessed 15th June 2020
- [22] Davis, M. Goadrich, "The relationship between precision-recall and ROC curves," in Proceedings of the 23rd International Conference on Machine Learning, ACM, pp. 233-240, Jun. 2006.
- [23] J. Huang, C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," in IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 3, pp. 299-310, Mar. 2005.