International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

PhishSentinel: Machine Learning Based Phishing Detection System

Nitish Kumar[1], *Harsh Raj*[2], *Nihal Kumar*[3], *Harsh Raj*[4], *Prof. Preeti Kushwah*[5]

[1]B. Tech Student, Oriental Institute of Science and Technology, Bhopal (Enrolment No: 0105CS231141)
[2]B. Tech Student, Oriental Institute of Science and Technology, Bhopal (Enrolment No: 0105CS231084)
[3]B. Tech Student, Oriental Institute of Science and Technology, Bhopal (Enrolment No: 0105CS231135)
[4]B. Tech Student, Oriental Institute of Science and Technology, Bhopal (Enrolment No: 0105CS231083)

A B S T R A C T

The use of phishing techniques is increasing. Attackers use this method to convince users to provide their private information such as user names, passwords, and bank account information. Phishing websites are designed to be nearly identical to legitimate websites so that it becomes difficult for regular Internet users to identify that they are at risk of being scammed. Conventional Security Filters (blacklists, WHOIS database information and verification of the domain age) only work after the phishing website has been reported, which could potentially lead to thousands of victims being affected before the exposure has been remedied. With the above mentioned in mind, a new concept in the detection of phishing websites has been developed that is based on Machine Learning. The name of this tool is PhishSentinel. PhishSentinel has been designed to detect phishing websites prior to their being reported or classified. The system accomplishes its tasks by analysing various features associated with a given URL, including (but not limited to) domain structure, link length, characters, content patterns and web page characteristics. Rather than depending on other sources for verification, the model of detection is based on examples of both phishing and legitimate websites. The design of the tool utilized Python and related libraries (Pandas, NumPy, Scikit-learn and Beautiful Soup) to apply several different algorithms (Logistic Regression, Support Vector Machine, Random Forest and XGBoost) for the purpose of training and evaluating the different models developed. Following experimentation, the best performing model is the Random Forest classifying model that showed the greatest level of accuracy and least number of misclassifications. In addition to detecting malicious URLs, the objective of PhishSentinel is to improve speed and adaptability to allow for on-time detection, as well as being able to detect phishing attempts. If improvements can be made to increase the amount of data used in training the model through methods such as using browser extensions with real-time scanning capabilities and deep learning methods, then the PhishSentinel application will become a useful tool for general users, including students and business professionals. PhishSentinel will continue to be built upon and will ultimately help to eliminate cybercrime as a result of phishing scams and to provide a safe browsing space for the general public.

1. Introduction

With an increase in the volume of digital services, online transactions and website interaction, there has also been an increase in the number of phishing scams being perpetrated. The advanced techniques used by phishing sites make it more difficult for users to determine which sites are real and which sites are phishing scams. Traditional means of detecting phishing scams are to check if the website URL is on a blacklist, conduct a WHOIS lookup and check the reputation of the domain that the URL resolves to. These methods of detection reactively check URLs for phishing scams and, as such, will not identify newly created phishing URLs fast enough to prevent harm to consumers during the early phases of a phishing scheme. Recent developments in the use of machine learning technologies have made it possible to detect phishing attacks proactively through the analysis of the structural features and lexical characteristics of URLs. By using structural features and lexical characteristics of URLs, machine learning algorithms have been able to distinguish between safe and unsafe hyperlinks without relying on external databases of phishing threats. The method for detecting phishing attacks through the use of machine learning is implemented in PhishSentinel, which combines multiple types of supervised learning algorithms to create an efficient, accurate, real-time method for detecting phishing attacks.

2. Literature Review

2.1 Phishing Detection Methods in the Beginning

Early detection of phishing relied almost entirely on WHOIS database checks and blacklist systems. As these methods were slow to update and could not prevent any newly created phishing URLs from being discovered, there was little or no real-time protection available.

2.2 Techniques for Feature Extraction from URLs

Researchers began looking into examining intrinsic URL features such as length and number of subdomains, suspicious characters, patterns of tokens used and sequences used for redirection, which allowed detection of phishing in its earliest stage without having to examine webpage content

2.3 Using Machine Learning to Classify URL

The use of machine learning, specifically Logistic Regression, Support Vector Machines, Random Forest and XGBoost, showed excellent ability to differentiate phishing from non-phishing URLs. The accuracy of tree models and boosting models were superior due to their ability to capture complex non-linear relationships.

2.4. Using Content to Detect Phishing & Hybrid Approaches

Researchers found that they could improve performance by incorporating HTML parsing, JavaScript analysis, and inspecting web page behaviour, however, this resulted in more computational requirements. Therefore, this method was not suitable for real-time detection situations where speed is necessary.

2.5. Using Ensemble Learning Models for Real-Time Phishing Detection

Recently, researchers have been reporting on the dependability of using ensemble methods by combining several classifiers together in order to decrease the rate of false negatives and increase the degree of robustness. URL-based methods are preferred for real-time scanning due to their superior speed and ability to function independently of threat feed inputs.

3. Methodology

A well-defined multi-stage workflow has been developed in the proposed phishing detection system consisting of, but not limited to: Dataset Preparation, Feature Engineering, Machine Learning Model Development and Training, Ensemble Modelling, and Real-time (URL) Verification's (With Complete Frontend and Backend Components).

3.1 Dataset Acquisition and Pre-Processing

A dataset of phishing and legitimate URLs, labelled, was obtained from public repositories & trusted sources, ensuring its validity.

To ensure that the data met quality standards, the data was cleaned up by eliminating duplicate records and filling in any information that was missing & confirming the correctness of all label values. All URLs were converted into a standardized (abbreviated) format during feature extraction process, enabling the same method for analysis of all URLs. To ensure equal representation of the class distribution, minor modifications (if necessary) were applied using balancing techniques. Once the dataset was cleaned up, it was divided into 80% for training and 20% for testing to assess the performance of the models created from the respective datasets.

3.2 Feature Extraction and Engineering

Using a script written in Python, a feature extraction method was created to generate useful numeric representations of unstructured URLs. Some features being analysed include lexical (e.g., the length of a URL, the length of its domain, how many special characters exist in the URL, if there are suspicious symbols present such as "@", "-", and multiple "/", etc.) and whether or not the URL points to an IP address/domain. On the other hand, structural characteristics are also being analysed (e.g., the type of protocol being used [HTTP/HTTPS], how many subdomains are contained within the URL, how many levels of redirection exist with the URL, and how complicated the path of the URL is).Feature scaling and normalization techniques for models sensitive to the number assigned to features were employed. Redundant features or features with low impact were filtered out during the training process to help increase efficiency and reduce noise levels during training.

3.3 Machine Learning Model Training

We identified four machine learning algorithms as good candidates for this project: Logistic Regression, SVMs, Random Forests and XGBoost, because they together provide a broad range of both linear (Logistic) and margin based (SVM) and ensemble based (Random Forest/XGBoost) learning methods.

With Google Collab. we were able to conduct our training in an efficient manner and manage resources more easily. Hyper-parameter tuning through Grid Search and Cross Validation enabled us to optimise the performance of our models and minimise the risk of overfitting. To assess the models' performance against one another, we compared them using various evaluation metrics: accuracy, precision, recall, F1 score and confusion matrices, and made further refinements to the models' behaviour. A thorough analysis showed us that not all models perform equally on all URL patterns, indicating that an ensemble approach would be necessary to achieve maximum overall accuracy.

3.4 Ensemble Prediction Mechanism

Using only their learned parameters, each of the trained models operated separately in determining whether the input URL was a phishing site or a legitimate one based on its training data. A majority vote was taken on the output of all the models that were utilized as an ensemble and combined into one reliable prediction of whether the URL was phishing or legitimate. By using an ensemble approach, this technique reduced the errors associated with the limitations of individual models and created a more consistent overall performance when testing with various data samples. By employing this methodology, it provided superior detection capabilities overall compared to using individual classifiers alone.

3.5 Backend Integration Using Flask

All final models were trained, exported and saved as pickle files to be used on the Flask back-end. The Flask back-end receives and processes every request and serves up each of the trained models with the processed feature vector in real time. The final response sent to the UI is generated based on the results of the voting mechanism. The architecture of the back-end is designed for modularity, maintainability, and efficiency for the support of real-time detection.

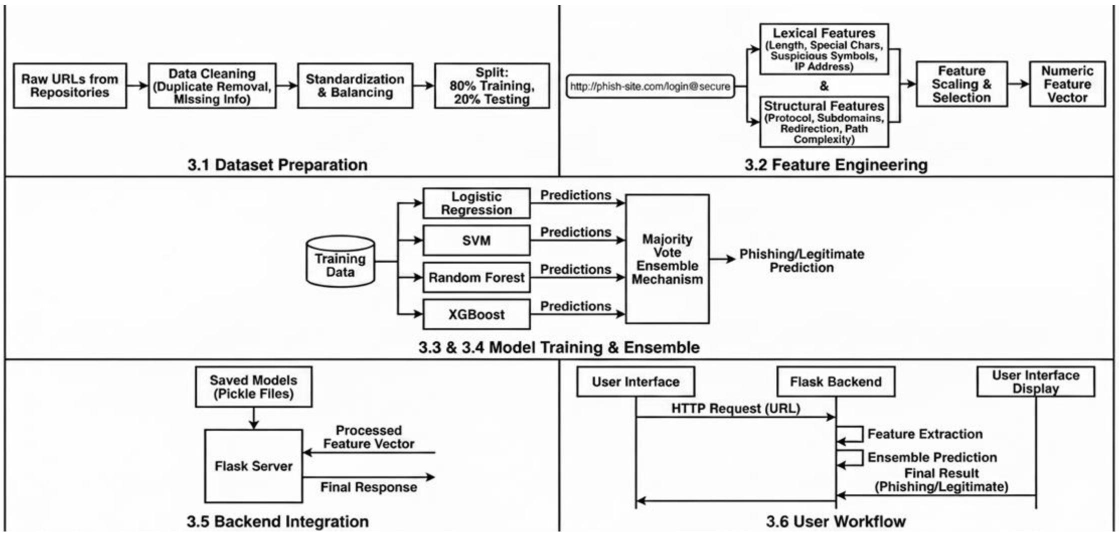
Raw URLs from Repositories
Data Cleaning (Duplicate Removal, Missing Info)
Standardization & Balancing
Split: 80% Training, 20% Testing
3.1 Dataset Preparation
http://phish-site.com/login@secure
Lexical Features (Length, Special Chars, Suspicious Symbols, IP Address)
&
Structural Features (Protocol, Subdomains, Redirection, Path Complexity)
Feature Scaling & Selection
Numeric Feature Vector
3.2 Feature Engineering
Training Data
Logistic Regression
SVM
Random Forest
XGBoost
Predictions
Majority Vote Ensemble Mechanism
Phishing/Legitimate Prediction
3.3 & 3.4 Model Training & Ensemble
Saved Models (Pickle Files)
Flask Server
Processed Feature Vector
Final Response
3.5 Backend Integration
User Interface
Flask Backend
User Interface Display
HTTP Request (URL)
Feature Extraction
Ensemble Prediction
Final Result (Phishing/Legitimate)
3.6 User Workflow

Figure 1 Phishing website detection system architecture

3.6 Frontend User Workflow

Using an easy to use interactive web interface allows the user to input a possible harmful URL into the system and submit it for investigation through an HTTP request sent to the backend Flask server. The results of that investigation will be shown to the user as soon as the Flask server finishes the investigation and will allow the user to easily determine if the URL being investigated is either Phishing or Legitimate. Combining the capabilities of the two workflows provides a usable system in the real world from both a design and ease of use point of view.

4. Result and Evaluation

Table 1 Classification Table – Logistic Regression.

	Precision	Recall	F1-Score	Support
bad	0.97	0.91	0.93	31355
good	0.96	0.99	0.98	78515
accuracy			0.96	109870
macro avg.	0.96	0.95	0.95	109870
weighted avg.	0.96	0.96	0.96	109870

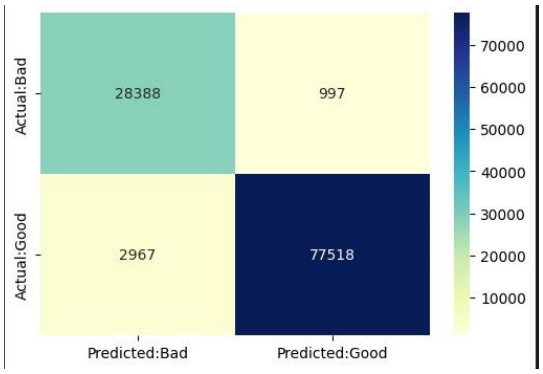

Figure 1 confusion Matrix - Logistic Regression

Table 2 Classification Table – Randon Forest

	Precision	Recall	F1-Score	Support
bad	0.91	0.89	0.90	30062
good	0.89	0.91	0.90	29938
accuracy			0.90	60000
macro avg.	0.90	0.90	0.90	60000
weighted avg.	0.90	0.90	0.90	60000

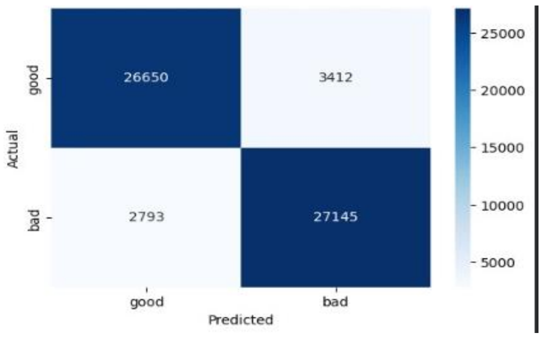

Figure 2 Confusion matrix - Random Forest

Table 3 Classification Table – XgBOOST

	Precision	Recall	F1-Score	Support
bad	0.75	0.94	0.83	25103
good	0.98	0.91	0.94	84767
accuracy			0.91	109870
macro avg.	0.86	0.92	0.89	109870
weighted avg.	0.93	0.91	0.92	109870

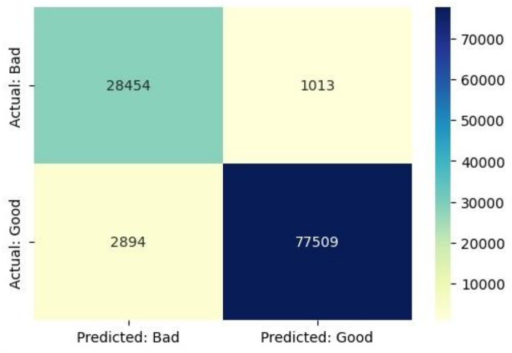

Figure 3 Confusion Matrix - XgBoost

All major evaluation metrics show that Logistic Regression outperformed both Random Forest classifiers and the baseline model. An examination of the classification report and confusion matrix results corroborate that Logistic Regression was both reliable and applicable in real-world settings.

Logistic regression, with its balance of high precision and recall and low rates of error, is the most accurate model for this dataset. The fact that logistic regression has a high linearity compared to other models suggests that adding complexity (such as through using advanced types of machine learning) may not produce a significant boost in predictive ability, thus making it an efficient and effective method of analyzing the data at hand.

4.Challenge

As PhishSentinel was under development, many practical issues emerged. Many of the URLs used to develop a clean, sorted dataset that included an equal amount of information occurred multiple times (duplicate) or were labelled incorrectly (mislabeled). Extracting useful features from the URL's required careful filtering to ensure that only the pertinent patterns of interest were extracted to create a clean and balanced dataset. Some of the machine-learning algorithms over-fitted during training, and ultimately required optimization to enable real-time prediction. Some legitimate URLs were incorrectly categorized as phishing URLs, requiring additional fine-tuning to ensure accuracy.

5.Conclusion

PhishSentinel, uses an analytic method, including a combination of models to improve overall accuracy and stability. In addition, by integrating browser extensions, using larger datasets, and incorporating new deep learning methods into the design, this software has the potential to be even more successful in cybersecurity-related applications.

6. Reference

[1] APWG report available at: https://apwg.org/trendsreports/ Last accessed on Nov 15, 2023

[2] K. Greene, M. Steves and M. Theofanos, "No Phishing Beyond This Point," *Computer*, vol. 51, pp. 86-89, June 2018.

[3] Fette, Ian, Norman M. Sadeh and Anthony Tomasic, "Learning to detect phishing emails," The Web Conference, 2007.

[4] A. N. Shaikhikh, A. M. Shabut and M. A. Hossain, "A literature review on phishing crime, prevention review and investigation of gaps," 10th International Conference on Software, Knowledge, Information Management Applications (SKIMA), Chengdu, China, pp. 9-15, 2016.

[5] Sami Smadi, Nauman Aslam, Li Zhang, "Detection of online phishing email using dynamic evolving neural network based on reinforcement learning," *Decision Support Systems*, vol. 107, pp. 88-102, 2018.

[6] L. Ma, B. Ofoghi, P. Watters and S. Brown, "Detecting Phishing Emails Using Hybrid Features," 2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing, Brisbane, QLD, Australia, pp. 493-497, 2009.

[7] Abu-Nimeh, Saeed, Dario Nappa, Xinlei Wang and Suku Nair, "A comparison of machine learning techniques for phishing detection," APWG Symposium on Electronic Crime Research, 2007.

[8] Khan, R., McLaughlin, K., Laverty, D. & Sezer, S., "STRIDE-based threat modeling for cyber-physical systems," 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe, ISGT-Europe 2017, vol. 2018-January, pp. 1-6, 2017.

[9] Y. Zhang, J.I. Hong, L.F. Cranor, "Cantina: A content-based approach to detecting phishing web sites," Proceedings of the 16th International Conference on World Wide Web, WWW '07, ACM, New York, NY, USA, pp. 639-648, 2007.

[10] G. Xiang, J. Hong, C.P. Rose, L. Cranor, "CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites," *ACM Transactions on Information and System Security*, vol. 14 (2), pp. 1-28, 2011.

[11] A. Le, A. Markopoulou, M. Faloutsos, "Phishdef: URL names say it all," 2011 Proceedings IEEE INFOCOM, pp. 191-195, 2011.

[12] E. Buber, B. Diri, O.K. Sahingoz, "Detecting phishing attacks from URL by using NLP techniques," 2017 International conference on computer science and Engineering, pp. 337-342, 2017.