IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Impact Factor: 4.819
Volume 6, Issue 1, June 2021

# Preventing Phishing Attacks in Real-Time Using Machine Learning - Phishr

**Om Sapate[1], Sumit Kolhe[2], Shantanu Taro[3], Vishal Kumar Kashyap[4]**
Students, Department of Computer Engineering[1,2,3,4]
Sinhgad Institute of Technology, Pune, India

**Abstract:** *Today, there's an exponential growth of e-services requiring the exchange of private and sensitive information over the web . Phishing techniques are emerging because the simplest solution to interrupt the weakest link of the security chain: the highest user. Social engineering attacks are deployed by financial/cyber criminals at a really low cost to induce naïve Internet users to reveal user ID, passwords, checking account and master card numbers. This problem must be addressed within the mobile field also , thanks to the massive diffusion of mobile platforms like smartphones, tablets, etc. Google has reported a 350% increase in phishing attacks from January to May of this year and as much as up to 500% in India. During this paper we propose a totally unique framework for phishing detection in mobile devices which, on the one hand exploits well-known techniques already implemented by popular web browsers plug-in, like public blacklist search, and, on the opposite hand, implement a machine learning based engine to make sure zero-hour protection from new phishing campaigns.*

**Keywords**: Mobile Phishing, Proxy, Social Engineering Attack, Phishing Attacks, Machine Learning

## I. Introduction

Social Engineering based attack leverages psychological manipulation of individuals , tricked into performing actions or disclosing tip . Phishing is one among the more known social engineering attacks and aims at exploiting weaknesses in system processes caused by user's behaviour. Indeed, a system are often secure enough against password theft (e.g., the client-server channel is encrypted), but nothing are often done against a naive user threatening the safety of the system by revealing her/his password to a fake Website reached, for example via an email-embedded HTTP link [3].

With the mainstreaming of mobile devices, an enormous chunk of the world's population has gone mobile and with this, the event of mobile malwares and phishing attack schemes possesses a huge boost. Despite the urging need for a protective environment from phishing activities for the mobile ecosystem, very less has been done about it. Even the studies and researches that dived into this field haven't necessarily converged towards solutions running on mobile devices.

## II. Methodology

### 2.1 Existing System

"Traditional" phishing detection concerning the specific subject of phishing detection on mobile devices, is proposed a phishing detection taxonomy for mobile environment, trying to depict all possible scenarios of phishing attacks and related countermeasures. Leaving aside all possible attacks related to very specific vectors (e.g., SMS, Bluetooth and Vishing), the paper stresses the lack of solutions dedicated to mobile devices other than black/white lists. The implementation of sample phishing attacks on the Android platforms demonstrated that attackers can spoof legitimate applications with high accuracy, suggesting that the danger of phishing attacks on mobile platforms is greater than it has previously been appreciated.

Most of the systems use a server-side component to send the observed data or to perform the learning process offline and plant the learned models back to the devices for the detection process.

### 2.2 Drawback of Existing System

- Although, it protects user from entering into phishing sites on Chrome Browser, it fails to do so on other browsers.
- It cannot detect newly created phishing sites (Someone has to report it).
- Doesn't work in other mobile browsers.
- Doesn't scan all the links opened by the device (Mobile).
- Cannot intercept all network traffic like GET requests.

### 2.3 Proposed System

To the best of our knowledge, none of the solutions proposed so far, shown a unified approach across different environments across subsets of the above-mentioned aspects. This is mainly due to the fact that, each solution often needs a number of prerequisites that are: i) difficult to remove, in case some detection technique is covered by other third-party software hard to adapt to other contexts, enabling the exploitation of the same detection technique for other threats iii) and burdensome to merge in a single tool that can exploit different targets, sources or approaches. Therefore, in this paper, we propose a unified reference model and present the PHISHR (Preventing Phishing Attacks in Real-time using Machine Learning).

It is based on Machine learning classification model and can detect phishing attacks in mobile. The machine learning model is kept on server and the client device is provided with an app which can communicate with this ML server and intercept the web pages and apps [2], [4].

### 2.4 System Architecture

The architecture represents mainly flow of requests from users to database through servers. In this scenario overall system is designed in three tiers separately using 3 layers called presentation layer, business logic layer and data link layer.

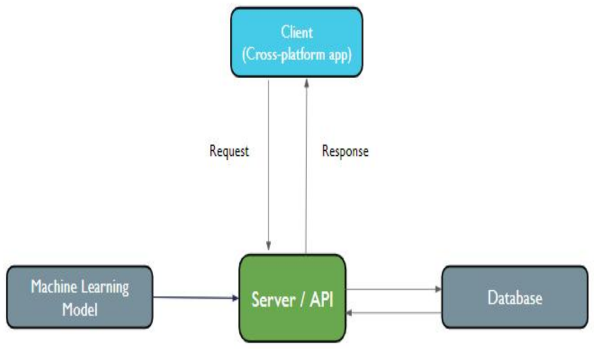


**Figure 1:** System Architecture

### 2.5 Process Flow

#### A. Data Collection

Multiple varied datasets containing the instances of URLs that are found to be malicious in nature will be collected. Value of attributes is in the form of integer -1, 0, and 1; 1 represents phishing, 0 denotes suspicious and -1 denotes legitimate [2], [5]

#### B. Machine Learning Model Training

With the collected data, models will be trained using multiple different algorithms based on literature survey such as linear model, decision tree, Random forest, etc [1], [2], [3], [4].

### C. API Creation

An API will be created with endpoints to serve related information about the user supplied URLs and their nature which will be intercepted by the client application. The API will serve as an interface for the trained machine learning models for as to supply the required information to the application.

### D. End-User Application Creation

To consume the so created API endpoints, a client app (Mobile) will be created that can, in real-time, scan the URLs visited by the users and intercept and classify them as safe / unsafe depending on the data fetched from the trained models via the API.

## III. Machine Learning Model

### 3.1 Dataset

We used the dataset from Kaggle (Phishing Website Dataset) and also from Phish tank repositories. Our dataset contains 31 columns with 30 different features and last column represent result [5].

The dataset contains 11000+ different observations where 70% of the data is used to train the ml model and other 30% is used for testing and predicting the accuracy of the model. Dataset consist of integer values 1, 0, -1 values (1: Legit, 0: Suspicious, 1: Phishing) [5].

### 3.2 Training Model

1. Load Dataset
2. Extract Features from dataset
3. Divide dataset into training and testing set
4. Run simulation of model to train Machine Learning model
5. Predict accuracy with test dataset

### 3.3 Machine Learning Algorithm

### A. Random Forest Classifier

1. Pick N random records from the dataset.
2. Build a decision tree based on these records.
3. Choose the number of trees for forest and repeat steps 1 and 2.
4. Each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the bulk vote.

### 3.4 Mathematical Model

The problem of Phishing URL detection is a case of binary classification problem as URL is either Legit or Phish.
To predict whether the URL is legit or not, we require features of the URL.
Let $w$ be the request of features of URL that need to be classified i.e.

$w \rightarrow^{x} \{legit, phish\}$ *equation (1)*

$X$ will be the system that takes features

Let fi be the feature as fi$\in w$

So, $w = \sum^{n}{}_{i} f_i n > 0$ *i.e.,* $\mathcal{W}$ *is non empty set* *equation (2)*

Therefore, the request w contains at least one feature on which prediction is to be done. As these features vary from simple to complex, the Model or System will be affected.

So, model $X = \{x_1, x_2, x_3, \ldots, x_n\}$ assigns label $y$ where,

$y = \{ {}_{1 \text{ (i.e. Phishing)}}^{-1 \text{ (i.e. legit)}}$ … *equation (3)*

gives,

$x_I : \mathrm{f}(w) \rightarrow y$ *from equation (1),(3)*

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 6, Issue 1, June 2021
Impact Factor: 4.819

In terms of dataset D, $(w_1, w_2,\ldots,w_n) \in$ D and each $w_i$ contains a set of features $f_i$. Also, the dataset is set of classes $C$=($C$1, $C$2) which represents legit and phishing URL's such that:

$$C1=\{w_i, f_i | w_i \in D,\ y = legit,\ i=1...m\}$$

$$C2=\{w_i, f_i | w_i \in D,\ y = phishing,\ i=m+1...p\}$$

As $w_i$ and $c_i$ ( $c_i \in C$ ), both belong to dataset D and $c_i$ is associated with $w_i$ in the dataset, we can represent $(w_i,(c_i))$ as a pair.

## IV. UML DIAGRAMS

### 4.1 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of the use case [7].

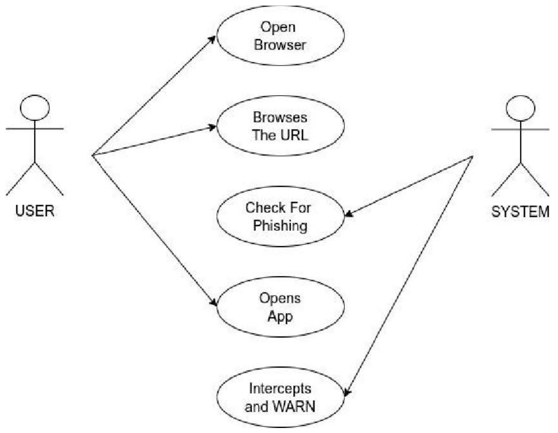


**Figure 1:** Use Case Diagram

### 4.2 Class Diagram

In software Engineering, A class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects translating the models into programming code [6], [7].

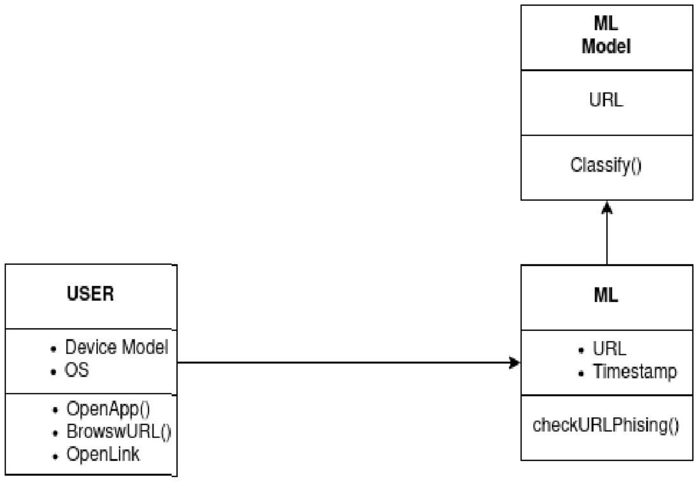


**Figure 3:** Class Diagram

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 6, Issue 1, June 2021
Impact Factor: 4.819

## 4.3 Activity Diagram

An activity diagram is a behavioural diagram i.e., it depicts the behaviour of a system. We use Activity Diagrams for instance the flow of control during a system and ask the steps involved within the execution of a use case [7].

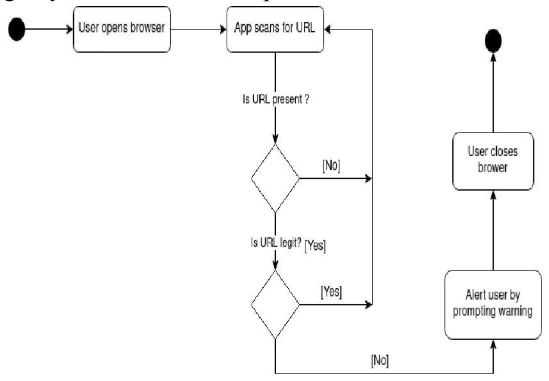


**Figure 4:** Activity Diagram

## 4.5 Component Diagram

Component diagrams are used in modelling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering [7].

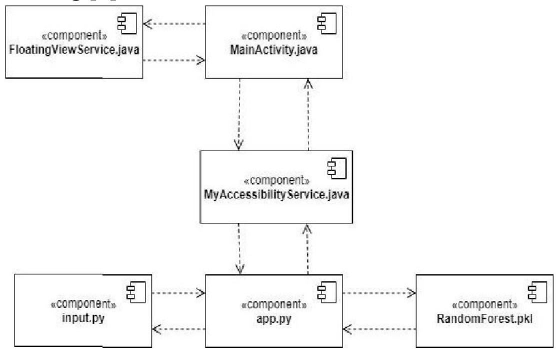

«component» FloatingViewService.java
«component» MainActivity.java
«component» MyAccessibilityService.java
«component» input.py
«component» app.py
«component» RandomForest.pkl

**Figure 5:** Component Diagram

## 4.6 Deployment Diagram

A deployment diagram is the configuration of run time processing nodes and the components that lie on them [7].

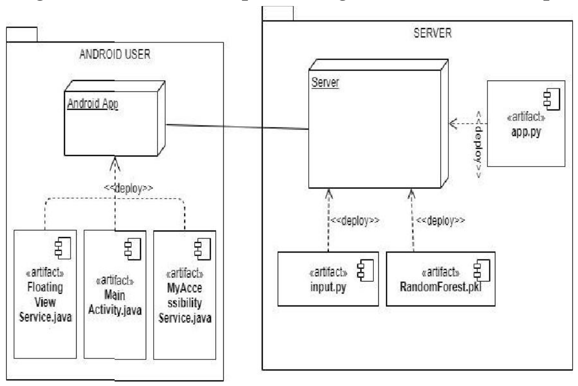

ANDROID USER
Android App
<<deploy>>
«artifact» Floating View Service.java
«artifact» Main Activity.java
«artifact» MyAccessibility Service.java
SERVER
Server
«artifact» app.py
<<deploy>>
<<deploy>>
<<deploy>>
«artifact» input.py
«artifact» RandomForest.pkl

**Figure 6:** Deployment Diagram

## V. Android App Working

1. Install the android app 'Phishr' in user's android device which created in this project.
2. Grant permissions asked by the app.
3. Whenever user opens URL in browser, 'Phishr' app monitors the browser for URL.
4. App send URL to API created and API gives the result where URL is safe or not.
5. If URL is not safe, 'Phishr' app shows warning on user's screen that URL is not safe else user will not interrupted by 'Phishr' app.

## VI. Testing And Validation

### 6.1 Introduction

Application Testing is the process used to help identify the correctness, completeness, security, and quality of developed user application. Software testing is widely used technology because it is compulsory to test each and every software before deployment.

Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors. Quality is not an absolute; it is value to some person.

With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and behaviour of the product against a specification.

### 6.2 Test Levels

There are many different testing levels which help to check behaviour and performance for software testing. These testing levels are designed to recognize missing areas and reconciliation between the development lifecycle states. In SDLC models there are characterized phases such as requirement gathering, analysis, design, coding or execution, testing, and deployment. All these phases go through the process of software testing levels.

- Unit testing tests the minimal software component and sub-component or modules by the programmers.
- Integration testing exposes defects in the interfaces and interaction between integrated components (modules).
- Functional testing tests the product according to programmable work. System testing tests an integrated system to verify/validate that it meets its requirements.
- Acceptance testing can be conducted by the client. It allows the end-user or customer or client to decide whether or not to accept the product. Acceptance testing may be performed after the testing and before the implementation phase.
- Beta testing comes after alpha testing. Versions of the software, known as beta versions, are released to a limited audience outside of the company. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

### 6.3 Test Cases

Guidelines for Test Cases

**GUI Test Cases :**

- Total no of features that need to be check Look and Feel
- Look for Default values if at all any (date & Time, if at all any require) Look for spell check.

**API Test Cases:**

- Check whether API works or not.
- Check for Legit website whether it gives correct output i.e. returns integer -1.
- Check for Suspicious website whether it gives correct output i.e. returns integer 1.

## VII. Result

### 7.1 App View and Working

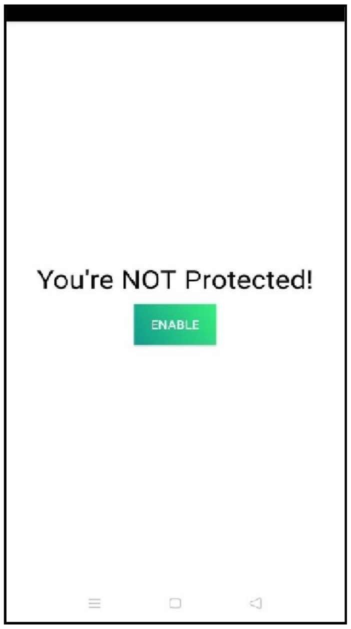


**Figure 8:** App UI Before Enable Protection

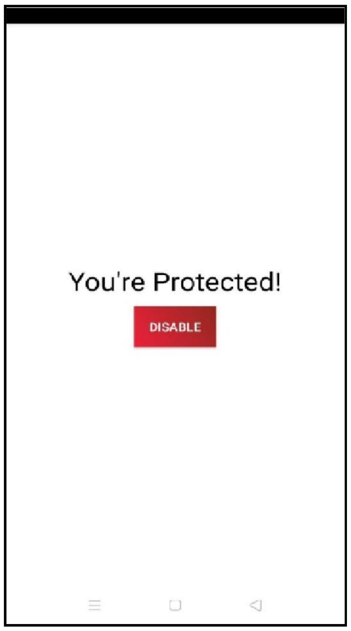
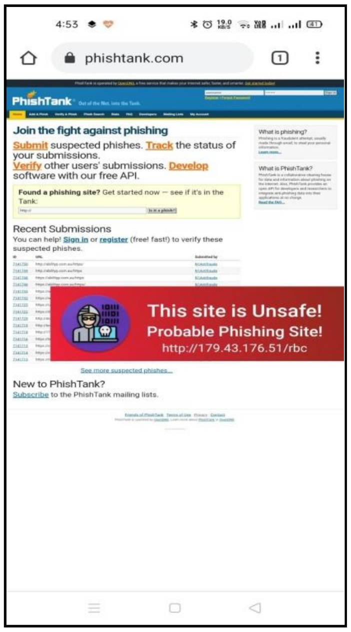


**Figure 9:** App UI After Enable of Protection

Volume 6, Issue 1, June 2021


4:53

phishtank.com

PhishTank

Join the fight against phishing

Submit suspected phishes. Track the status of your submissions.
Verify other users' submissions. Develop software with our free API.

Found a phishing site? Get started now — see if it's in the Tank:

Recent Submissions

You can help! Sign in or register (free! fast!) to verify these suspected phishes.

This site is Unsafe!

Probable Phishing Site!

http://179.43.176.51/rbc

See more suspected phishes...

New to PhishTank?

Subscribe to the PhishTank mailing lists.

**Figure 10:** App Pop-up After Accessing Phishing URL

Designed system working as expected and it will helpful to user to prevent from visiting malicious URL's.

## VIII. Conclusion

By using the PHISHR system, we solve the problem of detecting phishing sites on mobile devices in real-time. Now, the users are able to identify phishing sites/links without interacting. Phishr system itself shows floating warning sign before entering such websites**.**

## References

**[1]** Kahksha Jalal and Sameena Naaz, "Detection of Phishing websites using Machine Learning Approach", International Conference on Sustainable Computing in Science, Technology & Management (SUSCOM-2019) February 26 - 28, 2019

**[2]** Ram Basnet, Srinivas Mukkamala and Andrew H. Sung, "Detection of Phishing Attacks: A Machine Learning Approach", Springer -2008, 10.1007/978-3-540-77465-5_19

**[3]** Muhammet Baykara and Zahit Ziya Gurel, "Detection of Phishing Attacks", IEEE - March - 2018, 10.1109/ISDFS.2018.8355389

**[4]** Anupama Aggarwal, Ashwin Rajadesingan and Ponnurangam Kumaraguru, "PhishAri: Automatic Real Time Phishing Detection on Twitter", IEEE – March 2013, 10.1109/eCrime.2012.6489521

**[5]** https://www.kaggle.com

**[6]** https://www.tutorialspoint.com

**[7]** https://en.wikipedia.org/wiki/Unified_Modeling_Language