High Level Design (HLD)

Phishing Domain Detection

Revision Number: 1.0

Last date of revision: —--

Document Version Control

Date Issued	Version	Description	Author
27/03/2023	1	Initial HLD – V1.0	Santosh Singh

Contents

Abstract

Phishing stands for a fraudulent process, where an attacker tries to obtain sensitive information from the victim. Usually, these kinds of attacks are done via emails, text messages, or websites. Phishing websites, which are nowadays in a considerable rise, have the same look as legitimate sites. However, their backend is designed to collect sensitive information that is inputted by the victim.

This work discusses the implementation of a Machine learning algorithm to identify potential phishing attempts using the URL shared by the users and help them understand if they'll be safe if the link is opened.

We won't be opening the link to identify a potential phishing URL.

.

1 Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:
    - Security
    - Reliability
    - Maintainability
    - Portability
    - Reusability
    - Application compatibility
    - Resource utilization
    - Serviceability

1.2 Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3 Definitions

Term	*Description*
CS	Cyber Security
Database	Collection of all the information monitored by this system
IDE	Integrated Development Environment
AWS	Amazon Web Services

2 General Description

2.1 Product Perspective

The Phishing domain detection solution system is a machine learning-based pattern matching model which will help us identify potential harmful URLs and take necessary action..

2.2 Problem statement

To create an ML solution for phishing domain detection and to implement the following use cases.

- To identify URL based features.
- To identify domain based features.
- To categorize the URL as harmful or safe

2.3 PROPOSED SOLUTION

The solution proposed here is an ML based pattern matching algorithm which can be implemented to perform above mentioned use cases. If the URL is identified as harmful, the user will be informed about the same and would be advised to not click or interact with the URL.

2.4 FURTHER IMPROVEMENTS

This algorithm can be improved to recognize more patterns based on other features of the URL such as protocols.

2.5 Technical Requirements

This document addresses the requirements for detecting the harmful URL shared by the user. A simple input text field should be used for this purpose. The text field can be on a simple web-form.

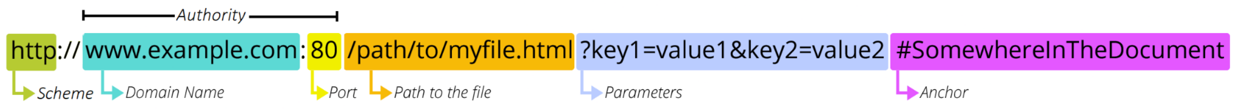
- The form should be able to accept URL

2.6 Data Requirements

Data requirements completely depend on our problem statement.

- We need data that is balanced and must have at least 50000 rows.
- We require at least 20000-30000 URLs for each class label with annotation.
- URL: Web address for locating internet resource
- URL length ranges between 0-8192 characters depending on the browser
- It stands for Uniform Resource Locator
- .It is of string data type: sequence of characters

A URL consists of a number of components. A URL is composed of different parts, some mandatory and others optional. The most important parts are highlighted on the URL below (details are provided in the following sections):


Authority
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument
Scheme
Domain Name
Port
Path to the file
Parameters
Anchor

- **Protocol:** The protocol is the first part of a URL and specifies the method by which the resource is accessed. The most common protocol is "http" (Hypertext Transfer Protocol), which is used for web pages. Other protocols include "https" (secure HTTP), "ftp" (File Transfer Protocol), and "file" (local file system).
- **Domain name:** The domain name is the second part of a URL and specifies the location of the server that hosts the resource. For example, "google.com" is the domain name of the Google website.
- **Port:** The port is an optional part of a URL that specifies the network port used to connect to the server. The default port for HTTP is 80, while the default port for HTTPS is 443.
- **Path:** The path is the part of the URL that specifies the location of the resource on the server. The path can include subdirectories and file names.
- **Query string:** The query string is an optional part of a URL that can be used to pass additional information to the server. The query string is separated from the rest of the URL by a question mark (?) and consists of key-value pairs separated by ampersands (&).

- **Example:**
  http://www.example.com:80/path/to/resource?query=string
  Protocol: http
  Domain name: example.com
  Port: 80
  Path: /path/to/resource
  Query string: query=string

2.7 Tools used

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn are used to build the whole model.

- PyCharm is used as IDE.
- For visualization of the plots, Matplotlib, Seaborn and Plotly are used.
- AWS is used for deployment of the model.
- MySQL/MongoDB is used to retrieve, insert, delete, and update the database.
- Front end development is done using HTML/CSS
- GitHub is used as a version control system.

2.7.1 Hardware Requirements: None

2.8 Constraints

The solution system must be user friendly, as automated as possible and users should not be required to know any of the workings.

2.9 Assumptions

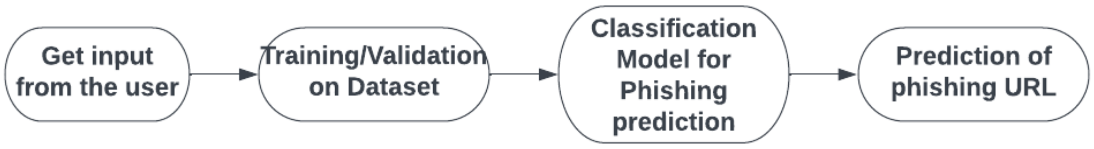
The main objective of the project is to implement the use cases as previously mentioned (2.2 Problem Statement) for a new dataset that comes as a user input in the form of a URL through a webform. A classification model is used for detecting the above-mentioned use cases based on the input data..

3 Design Details

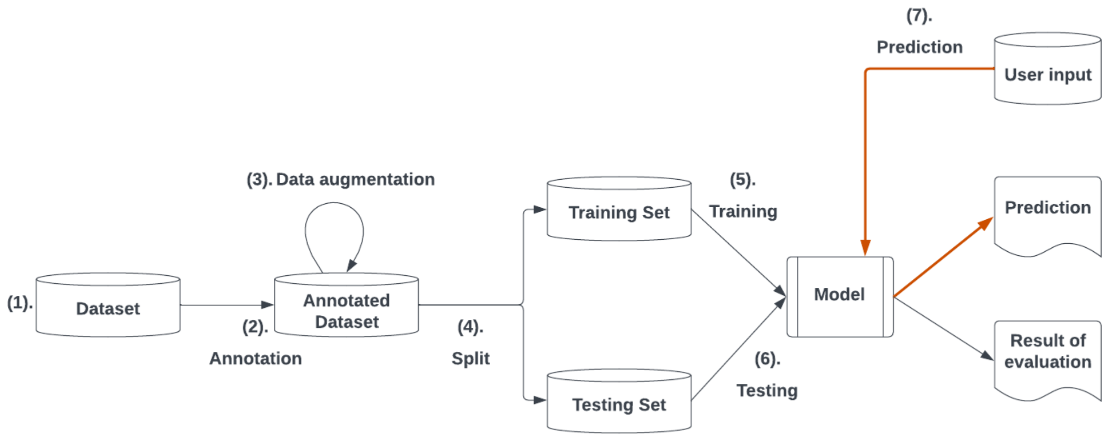
3.1 Process Flow

For identifying the different types of anomalies, we will use a deep learning base model. Below is the process flow diagram as shown below.

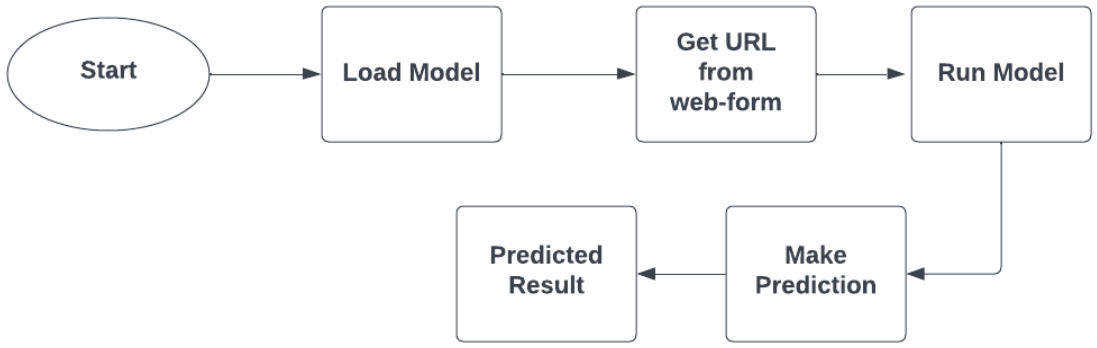
Proposed methodology


Get input from the user
Training/Validation on Dataset
Classification Model for Phishing prediction
Prediction of phishing URL

3.1.1 Model Training and Evaluation



3.1.2 Deployment Process



3.2 Event log

The system should log every event so that the user will know what process is running internally.

Initial Step-By-Step Description:

1. The System identifies at what step logging required
2. The System should be able to log each and every system flow.
3. Developers can choose logging methods. You can choose database logging/ File logging as well.
4. System should not hang even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

3.3 Error Handling

Should errors be encountered, an explanation will be displayed as to what went wrong? An error will be defined as anything that falls outside the normal and intended usage.

4 Performance

A Classification model is used for predicting the URL. When a prediction is made, the user will be informed and decide if they should interact with the URL. It should be as accurate as possible, so that it will not mislead the user into interacting with a malicious URL. Also, model retraining is very important to improve the performance.

4.1 Reusability

The code written and the components used should have the ability to be reused with no problems.

4.2 Application Compatibility

The different components for this project will be using Python as an interface between them. Each component will have its own task to perform, and it is the job of the Python to ensure proper transfer of information.

4.3 Resource Utilization

When any task is performed, it will likely use all the processing power available until that function is finished.

4.4 Deployment

5 Conclusion

The designed phishing domain detection model will detect harmful/malicious URL based on various URL features, so that we can identify URL which should be interacted with and safeguard the user data.

6 Resources

6.1 Dataset:

- dataset_full.csv
  Full variant - dataset_full.csv
  - Short description of the full variant dataset:
  - Total number of instances: 88,647
  - Number of legitimate website instances (labeled as 0): 58,000
  - Number of phishing website instances (labeled as 1): 30,647
  - Total number of features: 111
- dataset_small.csv
  Small variant - dataset_small.csv
  - Short description of the small variant dataset:
  - Total number of instances: 58,645
  - Number of legitimate website instances (labeled as 0): 27,998
  - Number of phishing website instances (labeled as 1): 30,647
  - Total number of features: 111

7 References

1. Phishing Wiki Page