|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PROJECT REQUIREMENTS SPECIFICATION  Web Page Classification for Safer Browsing  UE18CS390A – Project Phase – 1  ***Submitted by:***   |  |  | | --- | --- | | Manav Agarwal | PES2201800025 | | Rishab Kashyap | PES2201800065 | | Shreya Yuvraj Panale | PES2201800117 | | Shreya Venugopal | PES2201800688 |   Under the guidance of   |  | | --- | | **Dr. N Mehala**  Designation  PES University |   **January - May 2021**  **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  FACULTY OF ENGINEERING  **PES UNIVERSITY**  (Established under Karnataka Act No. 16 of 2013)  Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India |

TABLE OF CONTENTS

|  |  |
| --- | --- |
| 1. Introduction | 3 |
| 1.1 Project Scope and Motivation | 3 |
| 1. Literature Survey or Existing System | 3 |
| 1. Product Perspective | 3 |
| 3.1 Product Features | 3 |
| 3.2 User Classes and Characteristics | 3 |
| 3.3 Operating Environment | 3 |
| 3.4 General Constraints, Assumptions and Dependencies | 3 |
| 3.5 Risks | 4 |
| 1. Use Case Diagram 2. Functional Requirements | 4 |
| 1. External Interface Requirements | 4 |
| 5.1 User Interfaces | 4 |
| 5.2 Hardware Requirements | 4 |
| 5.3 Software Requirements | 4 |
| 5.4 Communication Interfaces | 5 |
| 1. Non-Functional Requirements | 5 |
| 6.1 Performance Requirements | 5 |
| 6.2 Safety Requirements | 5 |
| 6.3 Security Requirements | 5 |
| 1. Other Requirements | 5 |
| Appendix A: Definitions, Acronyms and Abbreviations | 5 |
| Appendix B: References | 5 |

# Introduction

This Document consists of the requirements and basic explanation of a system that will be used to detect malicious URLs. This will include a brief description of each requirement and its categorization based on its importance.

# Project Scope

The project will not only identify malicious URLs but also be able to derive some behavioral similarities in these URLs. Improvement over the detection schemes existing currently. A possible shortcoming, may be dealing with a static set of web URLs where it is difficult to see if a website is still safe or malicious in real time. Also the product may be computationally expensive since it needs more resources to access the entire network of URLs on analyzing in real time.

# Literature Survey or Existing System

The survey gives a general overview of all the possible approaches that can be used to do this project:

|  |  |  |  |
| --- | --- | --- | --- |
| Paper Details | Objective of paper, Techniques/Methods | Advantages | Limitations |
| Malware Detection using DNS Records and Domain Name features[1] | * A mix of Proactive Phishing Based and **DNS** to see which are the most corrupt TLDs based on previous datasets * 10 fold Cross-Validation used | * The feature selection is done in a good way since priorities have been assigned to the features based on effect they have on the accuracy | * Low Accuracy. * Attackers might change methods over time and new features need to be recognized. |
| Web Phishing Detection Using a Deep Learning Framework[2] | * Analyze real IP flows from ISP and propose a detecting method based on two types of features * Original * Interactive | * No need the content of the website as prior knowledge but only the properties and visiting behavior between user and website. * Combines proactive phishing approach also for better results | Mathematically complex |
| A heuristic technique to detect phishing websites using TWSVM classifier[3] | * Hybrid features including basic URL-based features, hyperlink-based features and similarity-based feature * Finds the degree of dissimilarity between the log-in page of phishing site and homepage of compromised domain. | * Can detect zero-day phishing attacks * Detects phishing sites hosted on compromised web servers * Search engine independence, page ranking independence, database independence * Parallelized feature extraction and classification to reduce time complexity | * False positive rate of 2.23% due to the non-availability of home page for the visited URL. * Legitimate sites which are hosted on free websites are also classified as phishing due to the high dissimilarity score between the free hosting domain and legitimate site * Fails to detect low-content web-pages such as single sign-on, however the designed phishing page with low target content has low success rate as it might create suspicious behavior to the online user. |
| A Machine Learning Approach for Detecting  Malicious Websites using URL Features[4] | * Compares multiple algorithms against combination of different attributes | * Feature Selection Techniques applied yielded good results as malicious URLs were accurately detected using only 8 features. | * Class imbalance reduces the accuracy of these algorithms, hence removing the class imbalance will be a problem to look into |
| Phish Haven—An Efficient Real-Time AI Phishing  URLs Detection System[5] | * Make use of URL HTML and URL Hit along with parallelly computed ensemble methods for detecting various attacks. * Summarizes the various processes to perform proactive phishing along with visual based similarity using multiple techniques. | * Efficient and fast detection of URLs based on the types of symbols * Can detect both human generated as well as AI generated malicious URLs * Faster computation with Machine Learning models using parallel threads. * Detection of malicious tiny URLs | * Not the most efficient when it comes to analysis of zero-day attacks * Real time third party dependencies |

Several other studies[6] use similar approaches to solve the given problem statement.

# Product Perspective

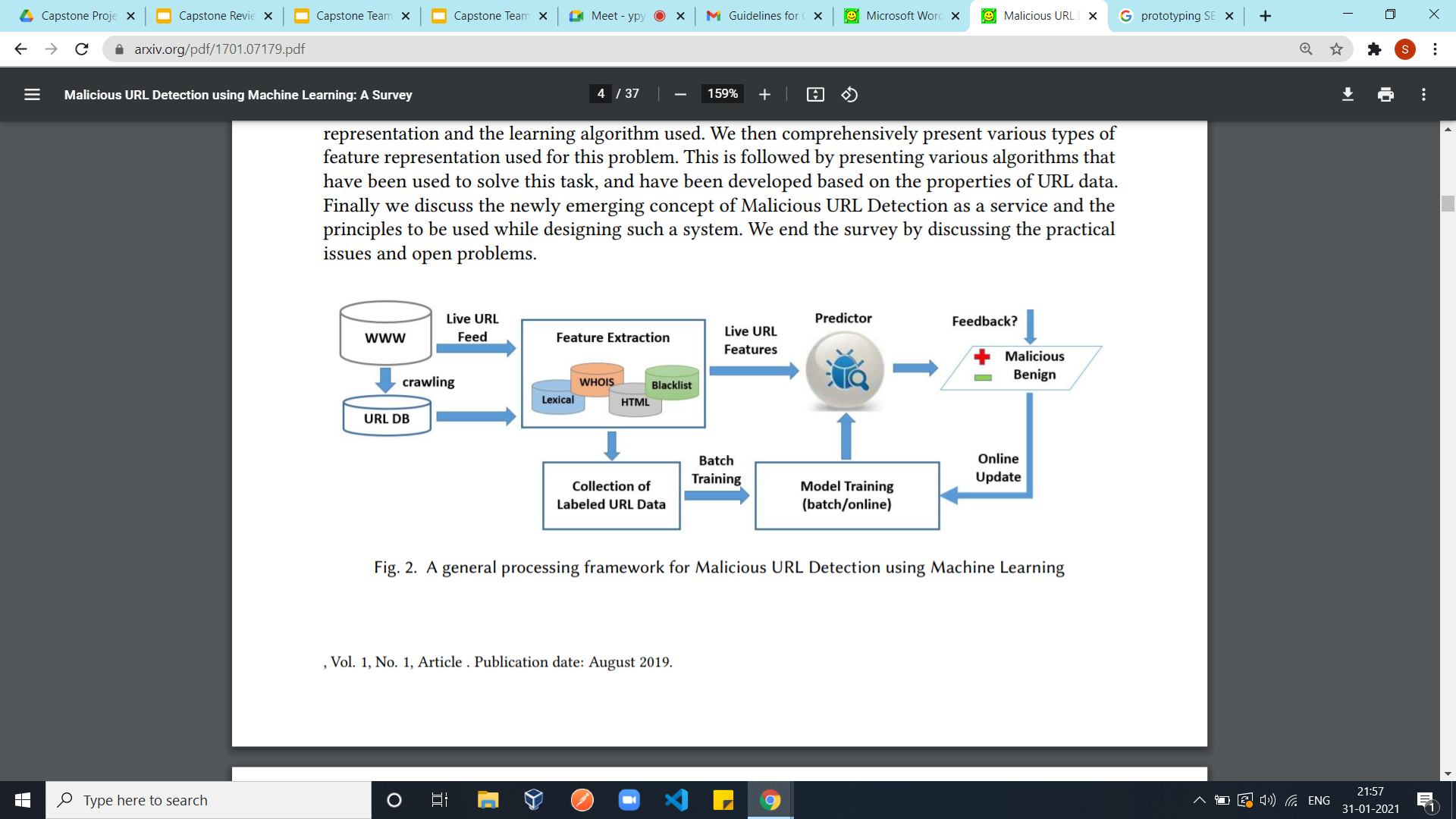
Malicious websites are responsible for a majority of the cyber-attacks and scams today. Malicious URLs are delivered to unsuspecting users via email, text messages, pop-ups or advertisements. Clicking on or crawling such URLs can result in compromised email accounts, launching of phishing campaigns, download of malware, spyware and ransomware that may result in severe monetary losses.

The standard and fastest way to identify malicious URLs is by comparing URLs against blacklists. However, blacklists are never exhaustive and lack the ability to detect newly generated URLs

This product however deals with identification of URLs posing a threat to browsing and web searches using a combination of various website detection schemes.

# Product Features

The set of functions performed by the product along with its sequence are given in the figure below:



* **Dataset Creation:**
  + Involves the URLs and the datasets from various sources
  + A database is created by combining all sources
  + A source may include URLs obtained by web-crawling also
  + From the dataset a sample is selected such that it has almost equal proportion of both training and testing data
  + Sample should also be diverse in terms of the domains it comprises of
* **Feature Extraction:**
  + All possible features are mined. These features maybe:
    - Search Engine Based: PageRank, Number of search results on querying key words, Rank of URL found when its key words are queried etc.
    - URL based: Lexical analysis counting the frequency of special characters, number of domain names, extracting keywords, http/https, presence of IP address, length etc.
    - DNS based: IP address, TLD, file system, information given by the Google API etc
    - HTML contents: For tags such as <p>, <h1>,<description>,<div> etc
    - CSS properties: For page layout properties like height, width, padding, etc
    - Other general properties such as whether the home page for the web page exists and whether it leads to a login page that leads back to the same homepage
  + Some of the above features such as the search engine based, HTML and CSS related ones may not be available since the URL may be blocked by the browser
  + Hence these mined features must be consolidated and valid preprocessing must be done
  + Using a machine intelligence algorithm such as Decision trees with or without a wrapper or some form of attribute selection, statistical significance etc.
* Dataset Splitting into testing and training
* **Training Module:**
  + Will apply multiple models and check which gives best accuracy
  + Models may be:
    - Classification models such as SVM
    - Random Forests
    - Neural Networks and its variants
    - Genetic Algorithms such as Harmony Search
  + These may be coupled with similarity metrics like the Jaccard Similarity
  + They may include bagging and boosting techniques for better accuracy
  + Combinations of multi-level or various combination of models may be experimented
  + All these will be evaluated based on some performance metrics and the one which is evaluated as best will be finalized
* **Predictor:**
  + Passing the testing data into the training model
  + Classifying live URLs
  + Propagate the classification to the URLs that are closely associated with the predicted URLs thereby improving the performance

# User Classes and Characteristics

The system can be used mainly by two kinds of users about which the details are given below:

|  |  |  |
| --- | --- | --- |
|  | **Naive Web User** | **Network Security Engineer** |
| **Definition** | Person who is using the internet to access a web page and needs to be protected from malicious URLs | Person trying to research in this field or using this tool for commercial purposes or trying to enhance it. Also includes organizations that keep BlackList records |
| **Frequency of Use** | Each time a web-page is encountered tool is used to classify | Depends on the motive behind use. Example, a researcher will only look into the working once, a maintainer as and when required for enhancements etc. |
| **Technical Capability** | No technical knowledge | Specialized technical knowledge in computer science |
| **Security Levels** | Should only have the UI visible | Back-end must not be entirely visible as it can fall in the wrong hands but information regarding structure of product is known |

# Operating Environment

* Hardware Platform
  + Any system that supports browsing
  + System that allows the usage of a keyboard for data or URL entry
  + Monitor or device must allow all forms of notifications to be seen so as to receive warnings from the model
* Software Platform
  + Must allow browsing platform of HTTP/1.1
  + Should be allowed to parallely allow the model to take in input data
  + Permissions granted to access suspicious html pages if necessary
  + Respond to requests sent by the model to block a URL
* Operating System
  + Any Operating System that supports browsing

# General Constraints, Assumptions and Dependencies

* **Access to some attributes are limited**

Phishing URLs that will be used for building the model mainly may not be available for search queries as it has been blocked by the browser. Hence the following features will not be accessible:

* + Contents of the web-page
  + Page-rank and web-graph connectivity related features: This is mainly because may of the blocked URLs do not appear on giving a search query
  + DOM characteristics
  + CSS characteristics
* **Cost is specific to client system for some attributes**

For attributes such as those given below the time taken to obtain information regarding it depends on the speed of the machine and the internet connection. These may cause unwanted delays in some cases.

* + DNS related attributes
  + Search Query Results
* **Feature Extraction methods are time consuming**

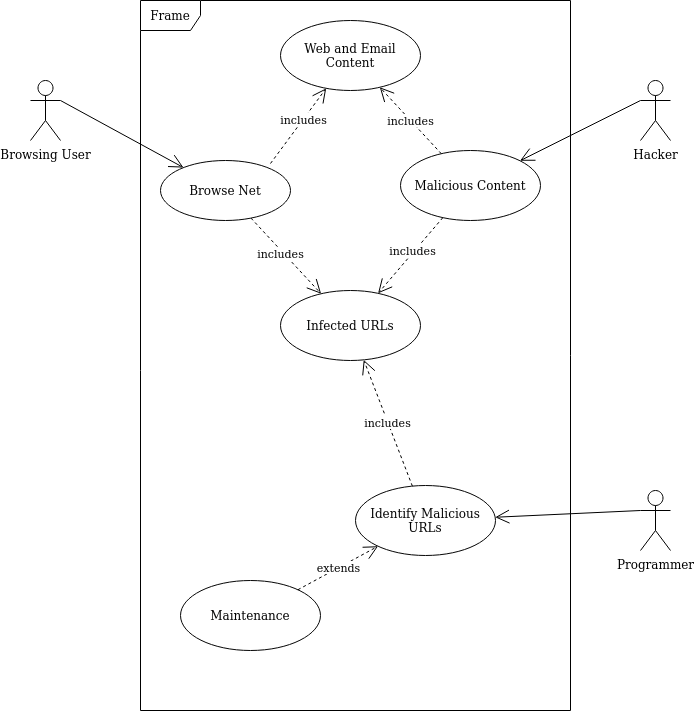
Some processes for feature extraction are tedious and since feature selection methods will be applied all the features extracted may not even be used. Some such processes are:

* + Lexical Analysis: Every character of the element must be traversed and sometimes re-traversed to obtain significant information.
  + CSS characteristics: Not only do the properties need to be extracted, those that are useful for the product must also be selected
  + Web-crawling to look into search engine related features: Each link has several paths that leads from it and each of them must be traversed
  + HTML tag data extraction: Traversing the DOM tree of a webpage is very tedious and time consuming
* **Effectiveness depends on the best kind of sampling done on the dataset to ensure no bias**
  + Proper proportion of benign and malicious URLs must be taken
  + Data must not be related to web pages of only a certain type. There should be diversity
  + Must not result in overfitting
* **It is assumed that:**
  + URLs given in the BlackList or WhiteList have been classified correctly
  + Malicious URLs have some sort of similarity i.e. follow a certain set of rules unintentionally
  + Reputed Pages are not malicious

# Risks

* Too much time taken for classification leading to a crash
* If the dataset is not sampled properly the results may be inaccurate or biased
* May have more time and space complexity than existing methods
* Complete feature extraction may lead to a sparse dataset leading to inaccuracies
* Heuristics or rules generated may not be consistent with the current malicious URLs
* The product may be hacked or face some form of cyber security threat such as Denial of Service attacks

# Use Case Diagram



# Functional Requirements

Req1: Each Input must be a URL

Req2: There should be almost equal proportions of benign and malicious URLs

Req3: The occurrence of special characters must be extracted from the URLs

Req4: For all inputs the proportion of URLs that are not blocked by the browser must be calculated

Req5: For all inputs the proportion of those that give results on querying in a search engine must be calculated

Req6: If the proportion of inputs that satisfy Rule5 are greater than 50% Page Rank related attributes must be taken into account

Req7: Keywords from the URL must be extracted

Req8: Additional domains if present in the URL must be extracted

Req9: URL must be checked for HTTP or HTTPS

Req10: Number of outlinks from the URL must be extracted

Req11: Outlinks of the URL must be extracted

Req12: PageRank of the URL must be found

Req13: Length of the URL should be extracted

Req14: Preprocessing of data must be such that null characters are handled if all attributes for a URL cannot be extracted

Req15: At least 3 URL feature selection methods must be applied

Req17: At Least 3 training and testing split methods should be applied

Req19: At Least 5 URL classification models must be applied

Req20: At least 10 possible model combination performance must be explored

Req22: A genuine accuracy level must be achieved before approving any model combination

Req23: Each Phase must have multiple unit tests (Stub-Drive approach) before implementation

Req24: Must satisfy the Hardware and Software requirements before deploying the model as given in the below section

# External Interface Requirements

# User Interfaces

# Requires a browsing tab or a textbox to enter the URL to be analyzed

# Option for description of the working of the model

# Messages for malicious URL will be displayed to the user before it can be accessed.

# Hardware Requirements

> Basic system supporting a simple browsing system

> Any screen or device that is compatible to display searches and handle the model

> Model Requirements:

* Any working computer system with any OS
* Keyboard to enter the URL required
* Mouse to operate and select the required entities

# Software Requirements

* Python 3.6 and above
* Jupyter Notebooks (Colab)
* Access to network
* Access to any browser such as Google or Firefox
* Data Storage Services for URL analysis
* HTTP/1.1 for browsing analysis

# Communication Interfaces

* Must have a decent internet speed to allow enough power for the model to identify the type of URL
* Browser must allow access rights to examine the user’s entered URL
* Permission to access html in web pages for analysis.

# Non-Functional Requirements

# Performance Requirement

* Req16: Highest Accuracy URL feature selection method for the dataset must be applied
* Req18: Highest Accuracy train-test split method for the dataset must be applied
* Req21: For a new URL the output must be given in 0.2s
* Req25: Should function smoothly without any hitches
* Req26: Must display only the model combinations that have the highest accuracy.

# Safety Requirements

* Req27: The model should not disrupt the browsing mechanism
* Req28: Users must not be blocked from the browsing system if the model faces any form of failures
* Req35: Maintainers should check for the health and performance of the system constantly on deployment
* Req36: Changes made to the system must be made offline and new modifications must be tested before allowing users to make use of it.

# Security Requirements

* Req29: Only maintainers or developers of this project must have access to the underlying code.
* Req32: Users must have the proper system requirements to access the model to allow web detection to prevent any issues inflicted on the application in this process.
* Req33: The system must be prepared to defend itself against any corrupt files holding viruses.
* Req34: Any issues to the system must immediately alert the maintenance crew to check for bug fixes and eradicate issues.

# Other Requirements

Req30: Product should not have more than 0.5% difference in accuracy from the existing accuracy

Req31: Product accuracy should not have more than 0.5% for different browsers

# Appendix A: Definitions, Acronyms and Abbreviations

URL: Uniform Resource Locator

HTTP: HyperText Transfer Protocol

HTTPS: Secure HTTP

BlackList: Set of Malicious URLs

WhiteList: Set of Non-Malicious URLs

# Appendix B: References

* [1] <https://dl.acm.org/doi/abs/10.1145/3231053.3231082>
* [2] <https://www.hindawi.com/journals/wcmc/2018/4678746/>
* [3] <https://link.springer.com/article/10.1007/s00521-020-05354-z>
* [4] <https://ieeexplore.ieee.org/document/8821879>
* [5]<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9082616>
* [6]<https://docs.google.com/document/d/1BAQRMT5p6TjQG2OHG7_4-q-JdHGFQ_Mmcz_jrXvcuFs/edit?usp=sharing>