Malicious URL Detection

A dissertation report
submitted in fulfillment of the requirement for the degree of
Bachelor of Technology
by

Sumit **(2017UCP1323)**
Himanshu Rawat **(2017UCP1230)**
Rajat Gedam **(2017UCP1254)**

Under the Supervision of **Dr. Smita Naval**

Department of Computer Science and Engineering
Malaviya National Institute of Technology
Jaipur 302017 (India)

May, 2021

Certificate

We,

Sumit (2017UCP1323)
Himanshu Rawat (2017UCP1230)
Rajat Gedam (2017UCP1254)

Declare that this project titled, "Malicious URL Detection" and the work presented in it are our own. I confirm that:

- This project work was done wholly or mainly while in candidature for a B. Tech degree in the Department of Computer Science and Engineering at Malaviya National Institute of Technology Jaipur (MNIT).
- Where any part of this project has been submitted for a degree or any other qualification at MNIT or any other institution, this has been clearly stated. Where we have consulted the published work of others, this is always clearly attributed. Where we have quoted from the work of others, the source is always given. Except for such quotations, this dissertation is entirely our work.
- We have acknowledged all main sources of help.

Signed: _____

Date: _____

Dr. Smita Naval
Assistant Professor
Department of Computer Science and Engineering
Malaviya National Institute of Technology Jaipur

Abstract

Websites are the basic building blocks of the internet, and information is an important source of knowledge. There are 4.66 billion active internet users worldwide, and all of their personal information can only be shared, made public, or used for other purposes with their permission. This is the responsibility of the website owner. However, some of the web sites try to go the other way, by injecting malicious code into the user's computer. Malware is a popular security problem that users face while surfing the internet. Providing malicious links on the web page could result in installing ransomware, spyware or a virus. To research the actions of malware in a page, features such as the internet protocol, port, Google index, HTTPS token, and web traffic are critical. This may lead to the leaking of sensitive information.

We created a machine learning model to predict whether a URL is malicious or benign.We have used various types of features and compare the performance of possible combinations and share our findings. For the application purpose, we create a chrome extension to make use of the model. It can be installed by the users into their browser. Every site they visit will be checked by this extension and predicted as malicious or benign. The final model used in the implementation of the system is Random Forest with Lexical Features. The use of such static features is safer and faster since it does not involve execution. Random Forest had 95% test accuracy, detecting large numbers of malicious Web sites from their URLs, with only modest false positives. Recommendations are provided for best practices that will aid in the detection of malware in the future.

Acknowledgement

It gives us a great sense of pleasure to present the report of the B.Tech Project undertaken during B. Tech Final year. We owe a special debt of gratitude to our supervisor Dr. Smita Naval, Department of Computer Science and Engineering, Malaviya National Institute of Technology for her constant support and guidance throughout our work. Her cognizant efforts have been a constant source of inspiration for us.

Sumit	Himanshu Rawat	Rajat Gedam
201UCP1323	201UCP1230	2017UCP1254

Table of Contents

List of Figures

List of Tables

Important Terms and Concepts

1. **Web Crawler**
   A web crawler is an automated script used to index websites based on their content. They are widely used by search engines for wide content results.

2. **Seed URL**
   It is a starting point for a web crawler.

3. **Malicious URL**
   A malicious URL is a potentially harmful link. Commonly used for delivering malware to targets.

4. **Benign URL**
   URLs that are not malicious are termed benign URLs.

5. **Lexical Features**
   Lexical features are extracted from the properties of the URL string.

6. **Host-based Features**
   Host-based features are extracted from the properties related to the hostname of the URL.

7. **Content-based Features**
   Content-based features are the features extracted from the source code of the web page. It needs to download the entire webpage first to extract the features.

8. **Online Learning Algorithm**
   Online Learning Algorithms include highly scalable algorithms. It can be used where the data is available sequentially and can be used to update the predictor for future prediction.

9. **Accuracy**
   The correct predictions on the test data in percentage are termed Accuracy.
   Accuracy = (Correct Predictions) / (Total Predictions)

10. **Precision**
    The percentage of the relevant results is termed Precision.
    Precision = (True Positive) / (True Positive + False Positive)

11. **Recall**

    The percentage of correctly classified relevant results is termed recall.

    Recall = (True Positive) / (True Positive + False Negative)

12. **API**

    Application Programming Interface is an interface that allows interactions between multiple software by defining the different kinds of calls or requests that can be made.

Chapter 1

Introduction

1.1 The Internet

In this era of the Internet, visiting various websites has become a much frequent activity. Everything is dependent on smart devices. Nothing is possible without the use of internet. Approximately 4.5 billion people have access to the internet. The Internet has such a wide and powerful functionality that it can be used for almost any information-based purpose, and it is open to anyone who connects to one of its constituent networks.It facilitates human contact through social media, electronic mail (e-mail), "chat rooms", newsgroups, and audio and video transmission, as well as allowing people to collaborate remotely. Every information is available to us at a single click. While we enjoy the luxury of getting information on the tips of our fingers, the use of malicious websites for various purposes is quite prevalent today. Users may click a link thinking they are visiting a genuine website but end up visiting a malicious website. An adversary may host a malicious website or send a malicious link via email. Either way, a user is at risk of getting exposed to a potential threat, which may cause damage or leak sensitive information. For protection against such malicious URLs, various methods have come into practice. Anti-virus software checks each URL that a user encounters and tells whether the link is malicious or not using a blacklist. Operating systems also can check malicious activities.

1.2 Malware

Malware, short for malicious software, is a broad term that encompasses viruses, worms, trojans, and other malicious computer programs that aims to destroy systems and

gain access to sensitive data. In this era of internet, use of malware for various unethical purposes has become prevalent. The purpose may be to gain unauthorised access to a sensitive data, for financial gains or for the sake of destruction of a system. A system can get exposed to a malware via a malicious email or a malicious link. The attacker may also manually install a malware on device either by physical access or by privilege escalation to gain remote administration control. Following are the common types of malware -

1.2.1 Types of Malware -

Virus :

Viruses bind their malicious code to clean code and wait for an unwitting user or an automated method to execute it. They can spread rapidly and widely, much like a biological virus, causing harm to systems' core functionality, corrupting data, and locking users out of their computers. Typically, they are stored inside an executable format.

Worms :

Worms are named for the way they infect computers. They thread their way through the network, starting with one infected machine and connecting to subsequent machines to spread the infection. This form of malware has the ability to rapidly infect entire networks of computers.

Trojan :

This form of malware hides inside or disguises itself as legitimate software, similar to how Greek soldiers hid in a giant horse to deliver their attack. It can bypass protection invisibly by creating backdoors that allow other malware variants easy access.

For protection on an individual level against malware, a good antivirus software and sufficient education about malwares is necessary. For protection at an organisation level, educating the employees about malwares is important. Also, installing a robust antivirus software which constantly monitors the networks against any infection is necessary.

1.3 Motivation

Clicking links and visiting websites is a popular practise done by users in this era of the internet. However there are many malicious websites and links present in the internet, clicking which may harm the data of the user. Sometimes a user is tempted to visit a

website because it appears to be attractive. In some other cases, a user clicks a link because a warning message related to presence of virus is shown. An uneducated user would click such links and install malwares on his/her computer. This would lead to loss or unauthorized access of sensitive data, financial loss, etc. In such scenarios, knowing whether a link is malicious in advance before clicking it becomes essential.

1.4 Malicious and Benign

Malicious URLs are those URLs which may cause harm to users data whereas benign URLs are those URLs which are safe to visit. These are the two categories in which our classifier classifies the URLs into.

1.5 Objective and Scope

We have developed a system to report malicious URLs in real-time while browsing using extensions in browsers. We have created a blacklist of URLs that will be dynamically updated. Our system works in many parts - First, our extension will send the current URL to the server, our server will check these URLs in the blacklist (or white list) and send back the response to the user. Second, we have created an ML-based model to predict the genuineness of URLs, based on which we have created our blacklist and whitelist. Third, we have used a web crawler that is continuously fetching URLs from the web and storing it in a database which will be used by our model to create our blacklist.

1.6 Outline

The rest of the report is organised as follows: We offer a brief summary of the relevant papers and literature that we reviewed or used as part of our literature survey in Chapter 2. In Chapter 3, we discuss about the architecture of our model both of the server-side as well as the client-side. We give an overview of the machine learning models and the types of features that we have used in our experiments in Chapter 4. In Chapter 5 we present the experimental results obtained using various models. We have also discussed the future work in our work in Chapter 5.

Chapter 2

Literature Survey

In this chapter we are going to discuss the existing work in the domain of malicious URL detection. we will formulate the malicious URL detection problem as a machine learning task. then we will discuss various works on feature representation. We will divide it in three categories, detection based on lexical features, host-based features and content-based features.

2.1 Problem formulation

Malicious URL detection can be formulated as binary classification task (i.e. malicious or benign). Mathematically, given dataset with T URLs $\{(u_1, y_1), ..., (u_T, y_T)\}$ where u_t for $t = 1, ..., T$, represent URLs in training dataset and $y_t \in \{0, 1\}$ is corresponding label for URLs. $y_t = 1$ represent URL is malicious and $y_t = 0$ represent URL is benign.Malicious URL detection is a two-fold process:

1. *Feature Representation:* extracting feature vector from URLs, i.e $u_t \longrightarrow x_t$ where $x_t \in R^d$ is a d-dimensional vector representing the URL; and

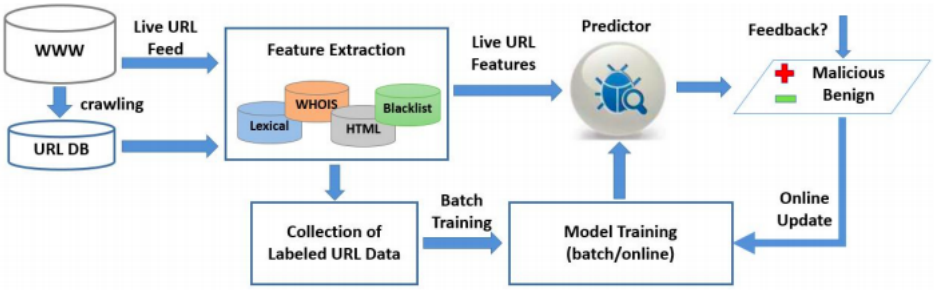

Figure 2.1: General framework for malicious URL detection using machine learning.

2. *Machine learning:* Learning a prediction function $f : R^d \longrightarrow \{0, 1\}$ which predicts whether URL is malicious or benign using its feature representation.

Our goal is to maximize the predictive accuracy. Both of above mentioned fold is important to achieve the goal.Figure 2.1 shows the general work flow for malicious URL detection system.

2.2 Feature Representation

Success of machine learning model depends on the quality of training data and the quality of feature representation. Given a URL $u \in U$ where U is set of all valid URLs, Goal of feature representation is to find a mapping $g : U \longrightarrow R^d$, such that $g(u) \rightarrow x$ where $x \in R^d$ is a d-dimensional vector, which will be used to train our machine leaning model. Feature representation is about

1. *Feature collection* ,which aims to collect relevant information about the URLs on the basis of which our model can classify URLs as benign and malicious.

2. *Feature preprocessing* ,which aims to convert unstructured information about URLs to a numerical vector which can be used by our machine learning model.

2.2.1 Blacklist feature

Trivial approach to identify malicious URLs is to use blacklists. Once a URL is detected as malicious it will be added to the list.Despite of its simplicity and ease of implementation, it suffers from high false negative due to problem in maintaining exhaustive up to date lists.But we can use presence in blacklist in another way.As Ma *et al.*[7] used presence in blacklist as a feature. They uses 6 blacklist from different service provider for this purpose. When used in conjunction with other features, they significantly improved the performance of prediction model.

2.2.2 Lexical feature

Lexical features are features obtained by properties of URL string. For example many malicious URLs look like benign by mimicking there names with minor variations.

Kolari *et al.*[6] suggested extracting words from the URL string. The string was processed such that each segment delimited by a special character (e.g. "/", ".", "?", "=", etc.) comprised a word. Based on all the different types of words in all the URLs, a dictionary was constructed, i.e., each word became a feature. If the word was present in

the URL, the value of the feature would be 1, and 0 otherwise. This is also known as the bag-of-words model.

Bag of word model causes loss of information about order in which words occur in URLs. Ma *et al.*[7] used similar lexical features with modification to account for the order in which word appeared. Ma *et al.*[8] made the distinction between tokens belonging to the hostname, path, top-level domain and primary domain name.Blum *et al.*[2] enhanced the lexical features by considering the usage of bi-gram features, i.e., they construct a dictionary, where in addition to single-words the presence of a set of 2-words in the same URL is considered a feature.

2.2.3 Host based feature

Host based features is obtained from host-name properties of URLs. They keep track of location, identity and properties of malicious hosts.

McGrath and Gupta[9] studied impact of host based feature on maliciousness of URLs.They found that phisher uses short URL services and time to live from registration of domain, in case of malicious URLs, was almost immediate.Ma *et al.*[7] and Ma *et al.*[8] further suggested various several host based features(like, WHOIS information, location, Domain name propertise).

2.2.4 Content based feature

Content based features are extracted from the content of web pages (i.e HTML,JS or CSS code). These are obtained by downloading entire web page.So these features are "heavy weight" and it took lot of time to extract content based features.

Hou *et al.*[5] proposed feature like length of documents, word count, average length of words, links to remote script,invisible object,word count per line, unsymmetrical HTML tags. Choi *et al.*[4] and Canali *et al.*[3] also suggested similar features with minor variations(number of iframe tag,presence of double documents, number of hyperlinks, number of hidden elements,number of elements with suspicious content,number of out of placed elements).Canali *et al.*[3] proposed features several more descriptive feature which were aimed at minor statistical properties of the page.

Features	Collection difficulty	Collection time	Processing time
Blacklist	Moderate	Moderate	Low
Lexical	Moderate	Low	Low
Host	Easy	High	Low
Content	Easy	High	Low

Table 2.1: Properties of different feature representation for malicious URL detection.

Chapter 3

Architecture

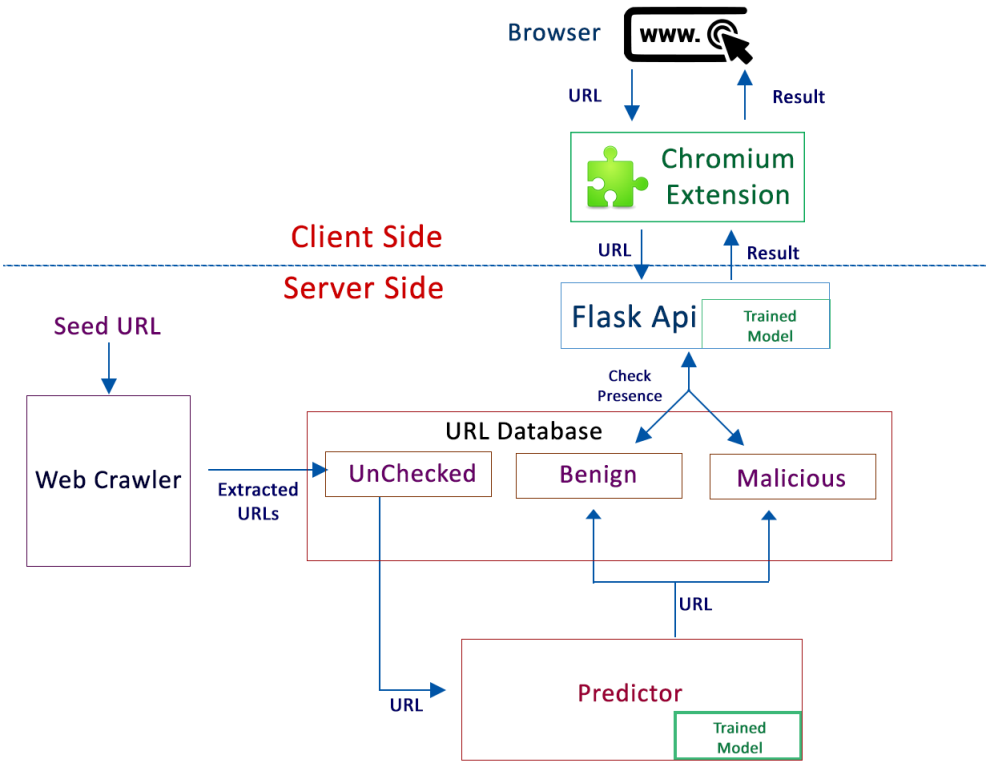

Figure 3.1: Complete Architecture of the Proposed Model

We have developed a system to report malicious URLs in real-time while browsing using extensions in browsers. We have created a blacklist of URLs that will be dynamically updated. Our system works in many parts - First, our extension will send the current URL to the server, our server will check these URLs in the blacklist (or white list) and send back the response to the user. Second, we have created an ML-based model to predict the genuineness of URLs, based on which we have created our blacklist and whitelist.

Third, we have used a web crawler that is continuously fetching URLs from the web and storing it in a database which will be used by our model to create our blacklist.

3.1 Server-Side

3.1.1 Database

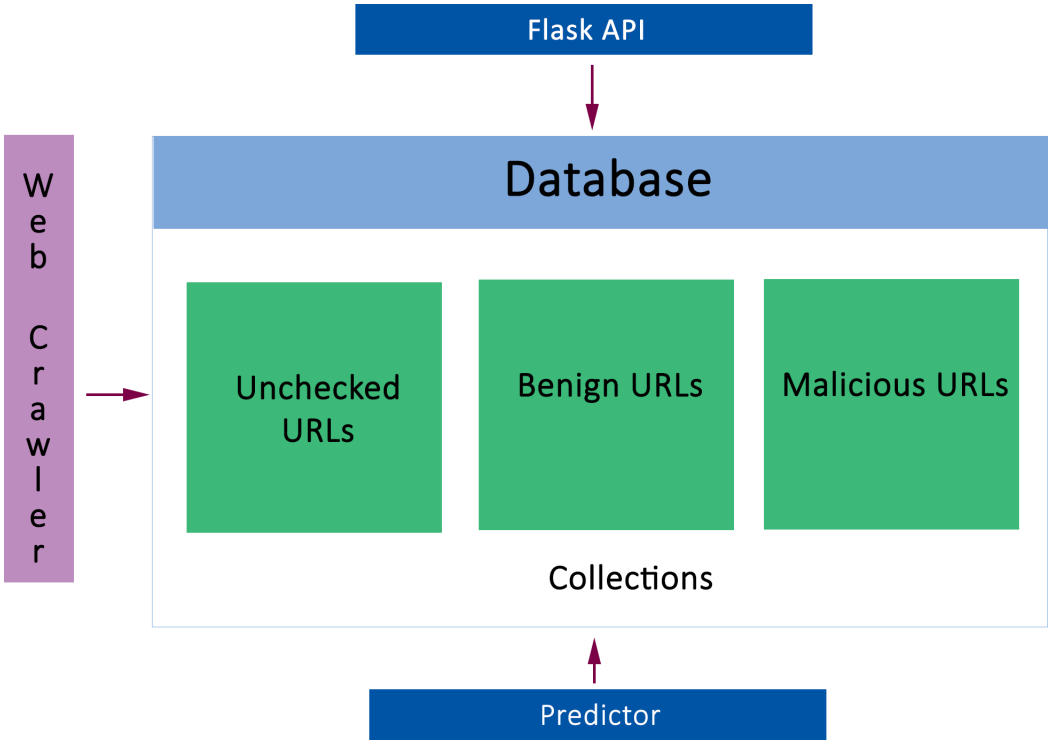

Figure 3.2: Database Architecture

The URLs to be processed could be in a huge amount. This would have resulted in large time complexity for processing them in the system. To make things fast and efficient, we made use of Database Systems. Database systems offer various functionalities from the start which increases efficiency, scalability, concurrency of the system.

In our proposed system, we have used MongoDB as our database. The followings are the collections we created in our database:

- **Unchecked URLs Collection:** These are the URLs inserted by Web Crawler so that they can be classified into Benign and Malicious URLs.
- **Benign URLs Collection:** These are the URLs from the Unchecked URLs Collection which are predicted safe by our model.

- **Malicious URLs Collection:** These are the unsafe URLs from the Unchecked URLs Collection which are predicted unsafe by our model.

We made use of the databases to skip the use of the machine learning model on the same URL for the second time. We store the predicted URLs into respective collections and make use of them if any URL arrives in the system for the second time.

Component	Collection from Database used by Component	Operation Type on the Collection
Web Crawler	UnChecked URLs	Insert
Flask API	Benign URLs, Malicious URLs	Read, Insert
Predictor	Benign URLs, Malicious URLs	Read, Insert, Update

Table 3.1: Database Collections accessed by Components

3.1.2 Web Crawler

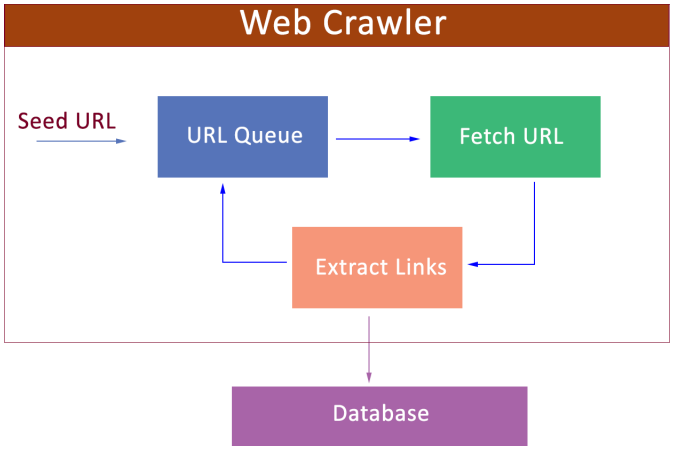

Figure 3.3: Web Crawler Architecture

The Web Crawler is used to fetch URLs from the web. Its architecture is shown below. It will take a seed URL and store that in a queue. Then it will pop a URL from the queue and fetch all the URLs present on that page and store them in the queue too, adding them in the database at the same time for our model to check.

3.1.3 Predictor

Predictor is a python file that is responsible for classifying the crawled webpage URLs stored in the Unchecked URLs Collection. It runs on the server-side continuously

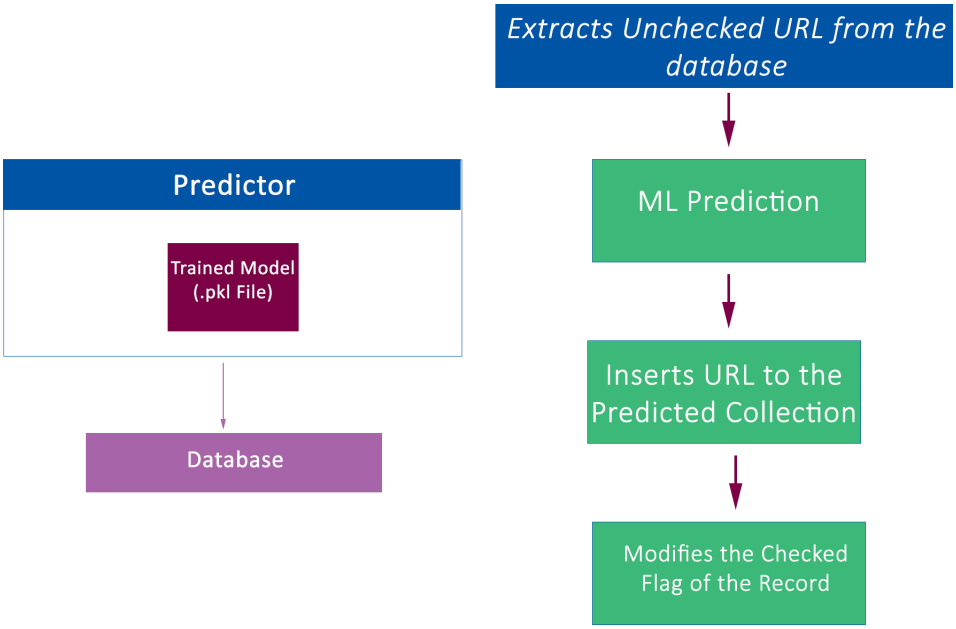

Figure 3.4: Predictor Flow

and checks for the Unchecked URLs in the collection. It makes use of the trained model stored as a .pkl file. Whenever it finds an unchecked URL, it extracts the features of that URL and tests it on the trained model.

The output is given as benign or malicious from the prediction. As per the output value, the URL checked is inserted into the corresponding collection of the database - either Benign URLs Collection or Malicious URLs Collection.

The URL is then marked checked in the database so that in the next iteration, it does not predict the same URL and skips from decreasing efficiency. Through this way, we reduce the number of operations on the database eventually resulting in low CPU usage and increasing efficiency of the system.

3.1.4 Flask API

Flask is a web application development framework. In our work, we used the Flask framework for building REST API. It manages the HTTP requests easily.

In our system, the chromium extension makes an HTTP request to the REST API created through Flask.

The REST API has the trained model file with it. It uses the model file for the prediction. The REST API has access over the database to check the URL in the Benign URLs Collection and Malicious URLs Collection.

Both the collections have URLs predicted by the model. We made use of these collections to skip the prediction process for the second time. The prediction process

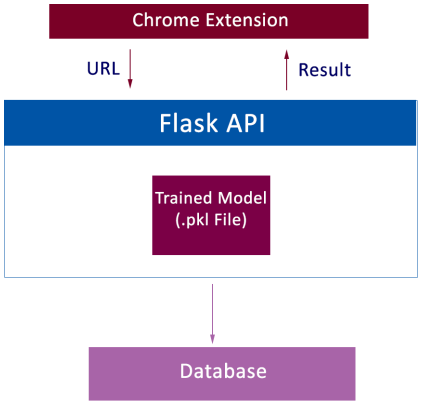

Figure 3.5: Flask API Architecture

involves extracting features from the URL and then testing it through the model and it takes a lot of time as compared to accessing databases.

In the end, the output is returned to the Chrome extension.

The figure 3.6 shows the flow of the REST API on being called through the HTTP Request by the Chromium Extension.

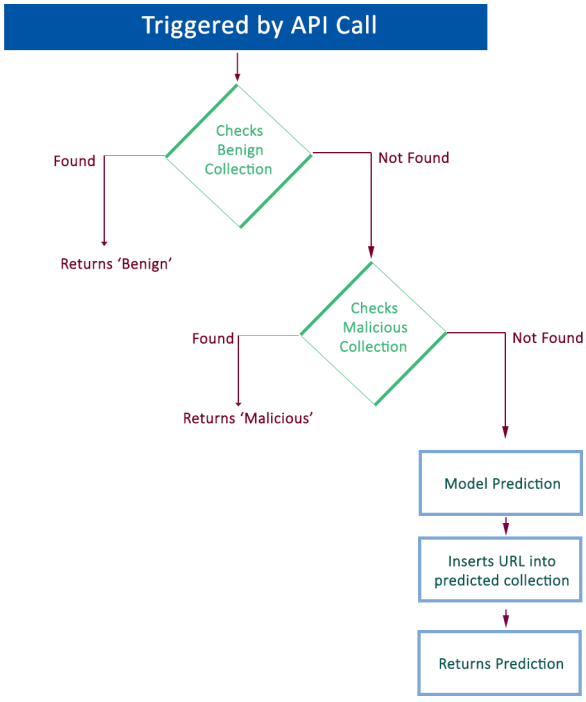

Figure 3.6: Flask API Flow

On triggering through the HTTP request, REST API extracts the URL from the HTTP request parameter. Predictor searches the URL in the two collections of the database - Benign URLs Collection and Malicious URLs Collection. If the URL is found

in any one of these, the corresponding result is sent to the chromium extension - ‘Benign’ or ‘Malicious’.

If none of the collections have an entry ‘URL’, the URL behavior is predicted through the Machine Learning Model. On prediction, the URL is inserted into the collection decided by the result of the ML model. After insertion, the same returned value is sent to the chromium extension.

3.2 Client-side

3.2.1 Chrome Extension

The interaction of the browser with our server-side components happens through the chromium extension. In our work, the chromium extension has 2 JS Scripts - background.js and main.js.

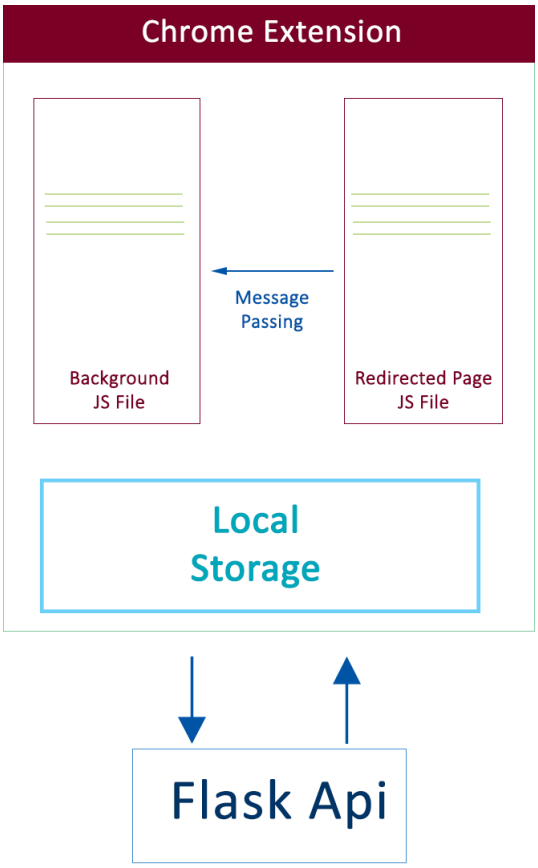

Figure 3.7: Chrome Extension Architecture

background.js

In our work, we need to check every URL the user visits on the browser. We found two events that were related to the URL Requests.

- **webRequest.onBeforeRequest**

  When a request is about to be made, this event gets triggered. This event reacts to every web request the page makes, from image request to JS file request. This can result in numerous calls on the server and can be a victim of a DOS attack. Also, the event blocks the loading of the page until it returns. This can become faulty if our server is in maintenance mode or when the response is taking too long.

- **tabs.onUpdated**

  This event is fired on the update of the tab. There could be a change in URL, title, or favicon. We filtered the event on the URL updates and used the same in our work. This resulted in only meaningful API calls to the server, only when there is an update in the address bar URL. Also, the event doesn't block the user which makes it user responsive and the user doesn't have to wait for the response.

Local Storage stores the URLs which were predicted malicious by our server but the user still wants to visit them. This means that the user has whitelisted the malicious URL. This could happen when the webpage is informational to the user or the model predicted wrong as it is based just only on the prediction.

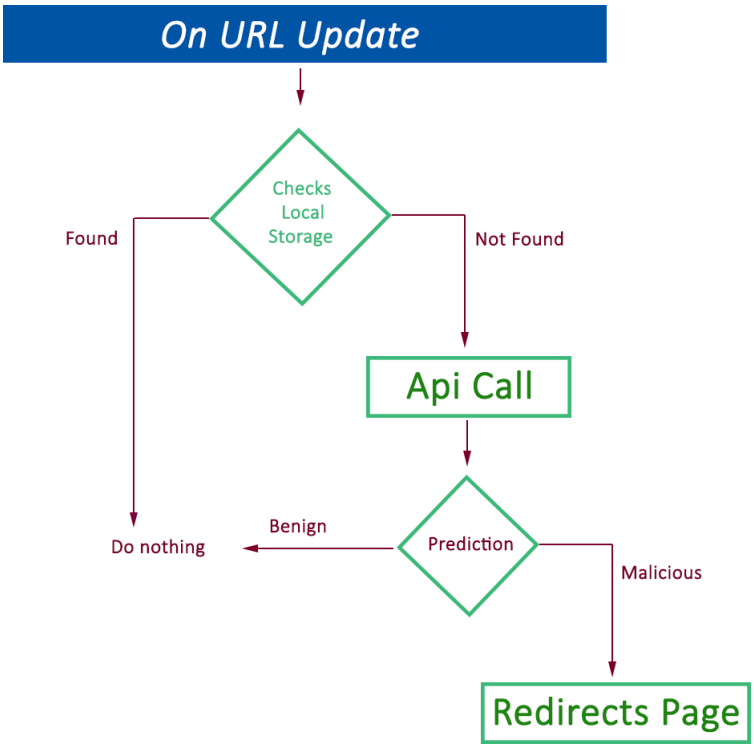

Figure 3.8: Background.js on URL Update Event

The URL is first checked on the local storage. If It is there, then the extension does nothing but if it's not there, the extension makes an HTTP request to the flask REST Api which then returns the prediction. Based on the prediction, we perform the following:

- If the predicted value is benign, the extension does nothing, the loading of the webpage continues.
- If it is predicted as malicious, then the extension forces the tab to redirect to a pre-defined webpage. The redirected page runs the main.js file.

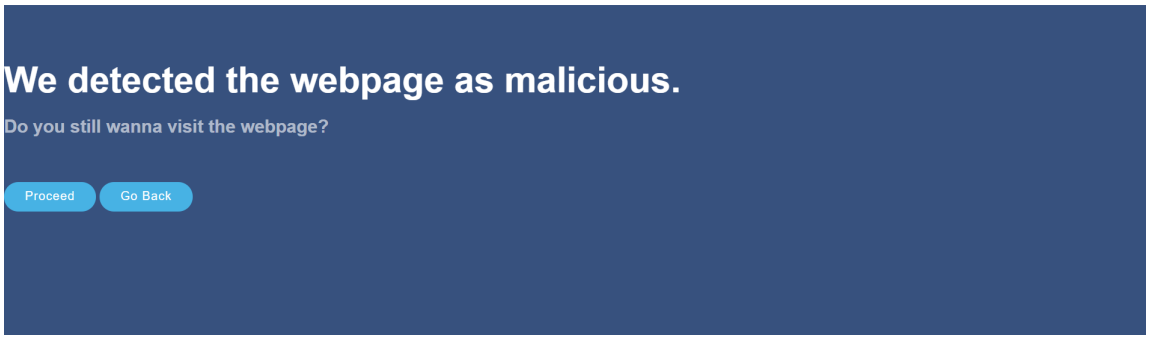

Figure 3.9: Redirected Page

main.js It is the JavaScript file of the redirected page. As it resides inside the extension directory, it can make use of extension libraries and methods. Two options are given to the user, whether to proceed or to go back to the safe page.

- Clicking the 'Go Back' button, the user goes back to the page from where it was directed to the malicious page.
- Clicking the 'Proceed' button means that the user wants to visit that page, and hence we need to store that URL somewhere so that the user doesn't have to do the same process the next time they visit the same URL.

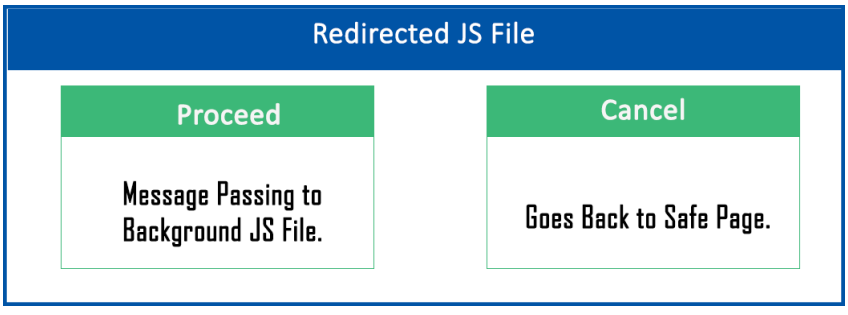

Figure 3.10: main.js

For this, the main.js sends a message to the background.js to save the URL in local storage. We didn't change the databases on the server because the page could be useful to that user only and that's why they want to visit or a wrong click by the user should not affect the experience of other users using the extension.

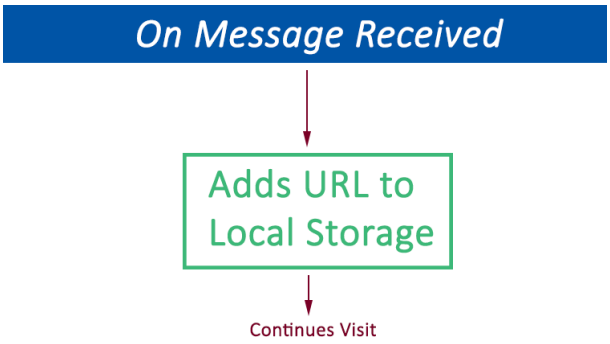

Figure 3.11: Background.js on Receiving Message

Chapter 4

Overview of Machine Learning Model

4.1 Dataset Creation

- **PhisTank.org**[1] - It is a website that contains malicious URLs. We have downloaded malicious URLs from their database. We got 10035 malicious URLs (which are currently active).
- **DMOZ** - we got 3494 benign URLs from DMOZ.
- **Kaggle**[10] - It contains 1056397 URLs.

We mixed all these URLs and randomly shuffled them. We have a total of 1070466 URLs (benign- 1003494, malicious- 66972). Our dataset is highly skewed as the ratio of malicious to benign URLs is about 0.0667.
For Content-based features, we want currently active URLs. In our dataset, we found around 3494 malicious URLs which are currently active and we have added the same number of benign URLs to it. So, we have 6988 URLs for content-based features.
We have divided our dataset into two sets: training (70%) and test set (30%). We have randomly shuffled our dataset to avoid any types of bias.

4.2 Feature Representation

We have used the following features in our model: -

4.2.1 Lexical Features

Lexical features are extracted from the properties of the URL string. The malicious nature of a URL is predicted based on how it looks. The length of delimiters, count of

Lexical Feature	URL Component
URL Length	URL
Dots Count	URL
Suspicious TLD	Top-Level Domain
Hyphen Count	Primary Domain
Subdir Count	Path
Domain Length	Primary Domain
IP Present	Domain
Double Slash Count	Path
URL Shortening Service	Domain
Count Subdomain	Subdomain
Count Queries	Query
Count '@' Symbol	Primary Domain

Table 4.1: Lexical Features

special characters, and length of URL are some examples of traditional lexical features. Advanced Lexical Features include domain-related features, file-name features, and argument features. The different lexical features we used in our work are shown in table 4.1.

4.2.2 Host-based Features

Host-based features are those features that are obtained from the host-name properties of the URL. There are several host-based features associated with a URL. However, the feature of importance to us is the duration for which the URL was active. This information can be obtained from the whois server. We obtain the creation-date and the expiration-date from the whois server. Using these two parameters we obtain the duration for which the website was active.
Duration becomes an important feature because malicious URLs are short-lived while benign URLs remain active for a longer time. Hence duration becomes an important feature for our model to distinguish between the two types of URLs.

4.2.3 Content-based Features

We have used beautifulSoup to traverse the content of pages. A total of 7 content-based features are used as suggested in [12]:

Lexical Feature
Number of <iframe> tags
Number of hidden tags
Number of script elements
Presence of meta refresh tag
Number of object tag
Number of included URLs
Presence of double documents

Table 4.2: Content-based Features

The raw content of a page acts as statistical information and content-based features are based on these statistical features. Most of them are self-explanatory. Some of them are explained below: -

- The number of included URLs - the number of elements with their source location. It could be object, form, frame, script, iframe. External content in a webpage can be included through these.
- Presence of double documents - It indicates the existence of a compromised web page that contains the body, head, and HTML tags twice or more.

4.3 Machine Learning Algorithms

In batch learning, the entire dataset is used to train the machine learning model. The learning ends here and it is shipped to the production. It is popularly known as offline learning.

The model has to be trained from scratch all over again if we need to update the model on some new data. We made use of two popular batch learning algorithms in our work: Support Vector Machine and Random Forest.

4.3.1 Support Vector Machine

Support Vector Machine is the most popular supervised learning method. It is based on the approach of maximum margin. It creates a hyperplane in an N-dimensional space that can distinctly classify the two classes. The two factors of any hyperplane, orientation, and position are calculated through the closest data points. These data points are known

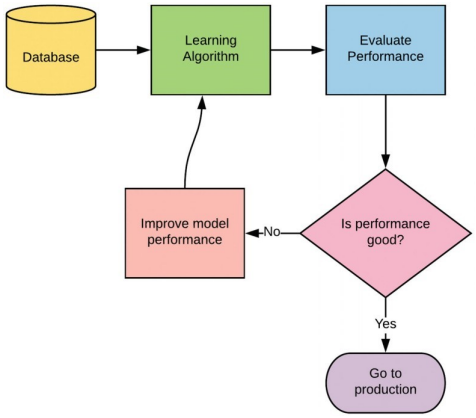

Figure 4.1: Batch Learning Algorithm

as Support Vectors. We used linear SVM from the sklearn library for the implementation of the Machine Learning Algorithm in URL Classification.

4.3.2 Random Forest

Random Forest is the most widely used learning algorithm. Random Forest is a combination of many decision trees. Feature Randomness and Bagging are used during the build of the forest of trees. Each tree gives a prediction and the majority prediction becomes the model's prediction.

4.3.3 Online Learning

Online Learning is a sequential form of learning that is much more scalable and efficient. While collecting data we observed that the ratio of malicious URLs to benign URLs collected was quite low. Even if we collect a sufficient number of malicious URLs, it was observed that many of them were not active, and collecting host-based and content-based features was not possible. This reduces the ratio even more.

We explore several methods to create a balanced dataset for our purpose. One way would be to collect the available and active malicious URLs from various sources and use an approximately equal number of benign URLs. The dataset would not be quite large but it is workable. A better solution would be to use an online machine learning algorithm. It will continuously accept a stream of data and the model would get updated with each stream. We can train the model for some time and then start predicting the labels for the incoming stream. Or we may predict the labels and update the model simultaneously. Either way, after a sufficient amount of time, the model will start giving better results because it would have then encountered sufficient numbers of malicious and benign URLs and learned the features. The only thing we need to take care of is that the number

of malicious and benign URLs encountered by the model should be approximately the same. To accomplish this, we will accept only five percent benign URLs. This is because we observed that the ratio of malicious URLs encountered to benign URLs encountered is approximately five percent.

Another reason for using online learning is that most malicious URLs do not have an active website. Because of this reason, it is not possible to extract content-based features from such websites. However newer malicious websites will help us in obtaining content-based features because they are still active. This will greatly enhance the learning and accuracy of our model with time. Using the strategy suggested above will help in creating a more robust and accurate model. This model of online learning can be used along with a chrome extension and web crawler. Our chrome extension and web crawler will encounter several URLs every day and our model will predict the nature of those URLs. It will simultaneously train the model with all the malicious URLs and five percent of benign URLs. This would lead to a much more accurate model.

Chapter 5

Experimental Results

5.1 Observations

1. Lexical Features
2. Host-based Features
3. Content-based Features

Features	Accuracy	Precision	Recall
(1)	95%	96%	95%
(2)	80%	57.14%	74.22%
(3)	75%	70.54%	97.84%
(1) and (2)	82.85%	92%	76.67%
(1) and (3)	86.19%	86.53%	88.53%
(2) and (3)	77.14%	79.41%	75%
(1), (2) and (3)	90.47%	85.57%	90.81%

Table 5.1: Observations on combinations of Features

We tested our model on different combinations of the features and obtained the accuracy shown in table 5.1. Random Forest with Lexical Features had the highest accuracy.

We have a huge dataset for lexical feature which result in higher accuracy. But for content based feature we need currently active malicious URLs which are very few in number and hence lower accuracy in case of content based features.

Host-based and content-based features are hard to obtain when using in a real-time application. It took around a minute for obtaining all kinds of features of a single URL. The user response would be highly affected on using these as our deciding factors.

When we trained the models using only one type of feature, we observed 6425 true positives and 246 false positives(total URLs = 13644) in case of model using only lexical features.We have a test set of 2096 URLs(30% of 6988 active URLs) for host and content based model. In case of model host-based features only, we observed 722 and 436 as true positives and false positives respectively. and in case of content-based model we observed a 911 true positives and 349 false positives.

As a next step, we started using the features in a combination of two. We observed 923 and 77 true positives and false positives respectively in a combination of lexical and host-based features. In case of combination of lexical and content-based features, the true positives and false positives were 604 and 72 respectively. Lastly, in case when host-based and content-based features were used together, the observed true positives and false positives were 810 and 210.

When we used a combination all three types of features we observed 920 true positives and 130 false positives.

Online Learning will not be feasible in real-time applications like chrome extensions. Online learning requires the correct labels so that it can improve for the future predictions. The user needs to respond to extension again and again eventually resulting in the bad user experience.

For the chromium extension in our work, we used Lexical Features and Random Forest for precise results considering user experience at the same time. This chrome extension can be used widely and can be a great tool because of increasing attacks and vulnerabilities on the sites.

5.2 Scope of Improvement

5.2.1 Model Architecture

Our model is predictive as we have used ML-based techniques to classify URLs. But to increase its accuracy and hence its reliability we can add honeypot in our systems

to further check malicious links dynamically. Honeypot is a virtual computer system that runs code in a virtual environment to check whether it's malicious or not. They are resource extensive software so our ML model can help in reducing the task of honeypots systems. As we will only check those URLs using honeypot which is flagged as malicious by our ML model. For that, we have to decrease the false-negative value of our model so that no malicious URLs are being left.

5.2.2 ML Model

- Including Javascript and CSS-based features, we have only included HTML-based features in our model, but by using CSS and javascript based features we can further improve the effectiveness of our model.

- Dataset, one of the major problems with using content-based features is the unavailability of active malicious websites to fetch content for training our model. Creating a dataset that will contain malicious links along with page content is challenging. One possible way could be to store calculated features only but in that case, it will be difficult to use other features in the future.

References

[1] , 2013 May, "phishtank, data and information about phishing on internet," `http://phishtank.org/`

[2] Blum, A., Wardman, B., Solorio, T., and Warner, G., 2010, "Lexical feature based phishing url detection using online learning," in *Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security*, AISec '10 (Association for Computing Machinery, New York, NY, USA). p. 54–60.

[3] Canali, D., Cova, M., Vigna, G., and Kruegel, C., 2011, "Prophiler: A fast filter for the large-scale detection of malicious web pages," in *Proceedings of the 20th International Conference on World Wide Web*, WWW '11 (Association for Computing Machinery, New York, NY, USA). p. 197–206.

[4] Choi, H., Zhu, B. B., and Lee, H., 2011, "Detecting malicious web links and identifying their attack types," in *Proceedings of the 2nd USENIX Conference on Web Application Development*, WebApps'11 (USENIX Association, USA). p. 11.

[5] Hou, Y.-T., Chang, Y., Chen, T., Laih, C.-S., and Chen, C.-M., 2010, "Malicious web content detection by machine learning," *Expert Systems with Applications* **37**, 55–60.

[6] Kolari, P., Finin, T. W., and Joshi, A., 2006, "Svms for the blogosphere: Blog identification and splog detection," in *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*

[7] Ma, J., Saul, L. K., Savage, S., and Voelker, G. M., 2009, "Beyond blacklists: Learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09 (Association for Computing Machinery, New York, NY, USA). p. 1245–1254.

[8] Ma, J., Saul, L. K., Savage, S., and Voelker, G. M., 2009, "Identifying suspicious urls: An application of large-scale online learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09 (Association for Computing Machinery, New York, NY, USA). p. 681–688.

[9] McGrath, D. K., and Gupta, M., 2008, "Behind phishing: An examination of phisher modi operandi," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, LEET'08 (USENIX Association, USA).

[10] Urcuqui, C., 2018 Jan., "kaggle, online platform for machine learning competitions.." `https://www.kaggle.com/xwolf12/malicious-and-benign-websites/discussion/64942`