AN EFFICIENT AND USABLE CLIENT-SIDE CROSS PLATFORM COMPATIBLE PHISHING PREVENTION APPLICATION

by

S.BEN STEWART 2016103513
N.DHANUSH 2016103021
G.SANTHOSH 2016103057

A project report submitted to the

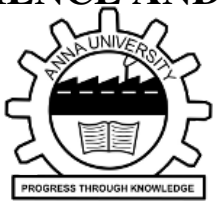
FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY, CHENNAI – 25

APRIL 2020

BONAFIDE CERTIFICATE

Certified that this project report titled **AN EFFICIENT AND USABLE CLIENT-SIDE CROSS PLATFORM COMPATIBLE PHISHING PREVENTION APPLICATION** is the *bonafide* work of **S.BEN STEWART (2016103513)**, **N.DHANUSH (2016103021)** and **G.SANTHOSH (2016103057)** who carried out the project work under my supervision, for the fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering. Certified further that to the best of my knowledge, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on these or any other candidates.

Place: Chennai
Date:

Dr.Angelin Gladston
Associate Professor
Department of Computer Science and Engineering
Anna University, Chennai – 25

COUNTERSIGNED

Head of the Department,
Department of Computer Science and Engineering,
Anna University Chennai,
Chennai – 600025

ACKNOWLEDGEMENT

We express our utmost gratitude to our guide **Dr.Angelin Gladston, Associate Professor** for guiding us with patience throughout the project. We thank her for leading us in the right direction through useful discussions and ideas. She also encouraged us during every phase of the project to come out with different perspectives to solve the problems. We value and appreciate her knowledge, principles and ability to bring out the best in us. Without her constant support and appreciation, we could have never done it to this extent.

We are very thankful to **Dr.S.Valli, Professor**, Head of the Department of Computer Science and Engineering, Anna University, Chennai-25, for providing us with the facilities of the department and for the constant support.

We are grateful to the panel of reviewers **Dr.A.P.Shanthi, Professor** , **Dr.V.Vetriselvi, Associate Professor** and **Dr.B.L.Velammal, Associate Professor** for their insightful suggestions and critical reviews throughout the course of our project.

We also thank our parents, family and friends for bearing with us throughout the course of our project and for providing the opportunity for undergoing this course in such a prestigious institution.

S.Ben Stewart **N.Dhanush** **G.Santhosh**

ABSTRACT

Phishing is a crime where the victim is contacted by an attacker posing as a trustworthy source and lure them into providing sensitive information like credit card details and personal identification numbers. These attacks are currently being blocked by web browsers that have a list of such phishing links. It takes several days and intense computing resources to prepare the list. Having a time lag in this process means that many victims are vulnerable at that point in time to such an attack. Inorder to make the process more efficient, the functionality required will be ported to the client side of the web browser. This makes sure that the time delay is averted and the phishing attack can be thwarted with fewer computational resources.

To solve the above mentioned problem, a web browser add-on that works as a background script on the client side is to be implemented. All the background scripts required will be made cross platform compatible to make the development easier and more efficient.

The main objective of this system named Off-the-hook-plus is to find if the web page that the user visits is a phish or not. The work is explained as follows.

The page redirect logs alongs with the page details are used to find if the site is a phish or not. And then if the site happens to be a phish, the similar looking site which this web page tries to impersonate is given. To make the repeated accesses faster, a whitelist is maintained, where all the safe sites are logged.

ABSTRACT

ஃபிஷ்ஷிங் என்பது நம்பத்தகுந்த ஆதாரமாக நடித்து மக்களிடமிருந்து கடன் அட்டை விவரங்களையும் தனிப்பட்ட அடையாள எண்ணங்களையும் பெற்றுக்கொண்டு முறைகேடான முறைகளில் பயன்படுத்தும் சட்டவிரோதமான செயலாகும். ஃபிஷ்ஷிங் தாக்குதலுக்கு பயன்படுத்தப்படும் வலைத்தளங்களை தொகுத்து இவற்றில் ஏதேனும் ஒன்றை இணைய உலாவியின்மூலம் அணுகாமல் தடுக்க பயன்படுத்தப்படுகிறது. இவற்றை தொகுப்பதற்கு பல நாட்களும் அதிக கணினி வளங்களும் தேவைப்படுகின்றன. எனவே ஒரு வலைத்தளம் ஃபிஷ்ஷிங் செய்வதற்கு பயன்படுத்துவங்கினதற்கும் ஃபிஷ்ஷிங் தொகுப்பில் சேர்க்கப்படுவதற்கும் இடையில் மக்கள் பாதிக்கப்படக்கூடிய வாய்ப்பு அதிகம். இக்காலதாமதத்தை குறைக்க ஒரு அமைப்பு வடிவமைக்கப்பட வேண்டியுள்ளது.

இத்திட்டத்தின் குறிக்கோள் யாதெனில், இறுதிப்பயனர் அணுகும் இணையதளம் ஃபிஷ்ஷிங் தாக்குதல்களுக்கு பயன்படுத்தப்படுகின்றதா இல்லையா என்பதே. அதன் வடிவமைப்பு பின்வருமாறு விளக்கப்படுகிறது.

இணையத்தளத்திலுள்ள உள்ளடக்கமும் மற்றும் சில பதிவுகளும் ஃபிஷ்ஷிங் செய்யப்படுகிறதா என்பதை கண்டுபிடிக்க பயன்படுத்தப்படுகிறது. மேலும் அத்தகைய நகல்களின் அசல்களும் கண்டறியப்படும். இப்பணியை மேலும் விரைவாக்கிட அனுமதிப்பட்டியலும் பராமரிக்கப்படும்.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

WWW	World Wide Web
DSL	Digital Subscriber Line
CERN	Conseil Européen pour la Recherche Nucléaire
URL	Uniform Resource Locator
IoT	Internet of Things
HTML	Hypertext Markup Language
JS	JavaScript
CSS	Cascading Style Sheets
SHA	Secure Hash Algorithm
TLD	Top Level Domain
IP	Internet Protocol
DDOS	Distributed Denial of Service
RFS	Rough Fuzzy Sets
PHP	Hypertext Preprocessor
HTTPS	Hypertext Transfer Protocol Secure
UI	User Interface
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
MB	Mega Byte

CHAPTER 1

INTRODUCTION

In this chapter we will be discussing how the issue of phishing began and how it was tried to be curbed both in policy writing and that of its implementation.

1.1 THE INTERNET

The Internet had its humble beginnings as a research project to share and access resources in other computers. And in fact these "other" computers were the main frames of the time that were located in the research facilities and college laboratories of the United States of America. This Internet is not the traditional Internet that we know as the World Wide Web. It was just a network of computers that could be operated from other nodes in the network. One main roadblock that was encountered was that the main frames were expensive and far exceeded the computational requirements of the then public. As a result of this there were only a handful of expensive mainframes with a few government and private institutions in a single country. Never would they have imagined the used for the internet unless the next era of personal computers boomed. As this phase had computers be considered almost as national assets they were sometimes compromised by enemy nations performing acts of sabotage[8].

1.2 THE WORLD WIDE WEB

In the stage of the personal computers, people bought computers to perform basic operations like spreadsheets and graphics related tasks. Once the internet had been introduced to those machines, they were still restricted by the slow DSL connections that the telephone companies provided. But once this threshold was broken, we would think that the Internet as we know today would have started to thrive. But it did not. The way in which the information is accessed was not intuitive till the World Wide Web made its appearance in 1989 developed by Tim Berners-Lee at CERN [4]. This era with the internet has its own set of unethical activities that were performed using computers or against the computers. Some elaborate sabotage attempts even involved using the internet connections of those computers.

1.3 THE INTERNET OF CONTENT

The World Wide Web started the phase known as the “Internet of Content” were the websites of different organisations could be accessed by any person on the internet to get to know the organisation and other details like the availability time, recruitment offers and so on. The impact of the World Wide Web was accelerated once the search engines were developed to index and display the billions of web pages that were loaded into servers connected to the Internet. This made it possible for people to access almost anything hosted on the Internet without knowing upfront the URL related to the resource. This opened a plethora of problems related to the act of performing unethical activities using or against the World Wide Web. The most common internet based crimes were phishing and site defamation. We will look into them a bit later, once after we have explained the next era of the internet. And this is where the roots of performing malicious activities for the gain of nations or

individuals can be traced to.

1.4 THE INTERNET OF E-COMMERCE

For the next era to come upon, there were a few changes that were required on the internet. They were the ability to host dynamic content based on the users and the ability for the user to interact with such content. These abilities were provided by the advent of web based programming languages with Javascript leading the way. This made it possible that people could perform online cash based transactions for the services that the internet made possible in the first place. This kind of opened Pandora's box for the cyber related crimes of this day. The notoriety of phishing greatly increased because of the scope that you could have the banking credentials of several thousands of users.

1.5 THE INTERNET OF PEOPLE

And before deep diving into the domain of phishing let us have a short look into the other eras of the internet that have come along. We've had the "Internet of People" which was brought by the advent of social media platforms like FaceBook, Twitter and LinkedIn. People now share much more data online about themselves over the internet to the public. This has led to even more problems like social media addiction, anxiety and attention deficit among the users. But let us not just paint a dark picture of this era and move on to what the future has in store.

1.6 THE INTERNET OF MACHINES

We are currently in this era of the "Internet of Machines" were more and more IoT devices with the capability to connect to the Internet and use it to communicate with other IoT devices and some centralised computers. This is probably exciting times as even the standards of the Inter-

net of Machines has yet to be decided and wonder if we would be having another World Wide Web like platform available with the machines in mind. Even in this age, the unethical activities can be performed as was done in the previous eras of the Internet.

1.7 PHISHING

Now that we have an understanding of what the Internet is and why it is so important and how it is being used, let us dive into the topic of phishing. If a definition were needed, phishing is any social engineering made to trick the people to access the malicious resources that may get critical information such as passwords, bank account numbers etc. from the victims. Phishing is done not only for the monetary incentives it provides but also for the impersonation of people in social media or to compromise the networks of organisations or countries[1]. They are targeted upon the users who have access to such details like an e-commerce customer or a company manager.

1.8 SOCIAL ENGINEERING

The most common ways to phish are to send emails to those who are related. What such emails contain are the malicious links and other content to convince the users that it is indeed legitimate. The same strategy is applied to the hosted pages that are pointed to by those links. As a result of this many unsuspecting people are tricked. And in order to prevent these instances, many methods have been devised. But the most important thing is to be constantly vigilant that phishing is possible and that you might be a target and never clicking on such links. Some such phishing sites have also been indexed and are even placed above the real site in search engines. So, care must be taken even when the links are provided by the search engine.

1.9 PHISHING PREVENTION

Let us look into the other methods to prevent phishing. Since search engines have to index all pages to be displayed for the user's query, it seems logical to use some mechanism to find such links while indexing and use the same in browsers to notify users that they are accessing a page which is probably used for phishing. This works for most cases, but fails for those dynamically created pages which are not indexed by the search engines and newly created sites which have yet to be crawled because it takes a few hours for search engines to index new pages.

1.10 PROBLEM STATEMENT

Phishing attacks that occur due to the time delay in searching the phishing websites and adding them to the phish list takes time and resources. And this has no role to be done by the client side. To prevent such attacks by using the client side resources is what must be achieve by Off-the-hook-plus.

1.11 OBJECTIVES

The objectives of Off-the-hook-plus are as follows. It must be cross platform compatible that most users can benefit from it. It must be accurate enough to be trusted with the phish detection. It must have a low memory usage so that even machines with low main memory resources can use them. The phish detection must also maintain the accuracy overtime making it temporally resilient. Also the target phish website detector will enable the users to find which website the phish is trying to impersonate.

1.12 ORGANIZATION OF THESIS

In the following chapters, we will look into the details of the system architecture, implementation and evaluation of Off-the-hook-plus.

Chapter 2 gives the architectural models that have been used to detect phish websites along with the parameters that were incorporated. Then, the methods to optimise the models have been discussed. Finally we get to the part where we review the previous works with respect to Off-the-hook-plus to see how it is different from them. Chapter 3 gives the system architecture of Off-the-hook-plus along with its high level block diagram. Chapter 4 details down into the module design which gives the pseudocode of how each module works. Chapter 5 gives the implementation details for Off-the-hook-plus along with the expected inputs and outputs. Chapter 6 evaluates Off-the-hook-plus on the lines of its objectives to make sure that they meet those with performance metrics. Finally, we conclude what Off-the-hook-plus is and also list the future work that can be done on it. And then, we have the Appendix A which gives a detailed look into the possible states that the Output UI module can be in because of the phish detector and the target identifier.

CHAPTER 2

RELATED WORK

This chapter would be a deep dive into the methods that have been implemented or at least served in the process of creating one to detect phishing locally in the client side machines.

2.1 AUTOMATIC PHISHING CLASSIFICATION

The work [2] by Colin Whittaker, Brian Ryner and Marria Nazif for Google provided the base benchmark for most of the future works and so would be better if we have a better understanding of what they did. For creating the base dataset required to train a machine learning model, they used the links from the Gmail spam filters [6] and also those that were submitted by other users. The features used are the URL, the contents of the HTML page and also where the page is hosted. These features are then used by the model which is a logistic regression classifier to find if the site is used for phishing or not.

The training for the model is done offline using the blacklist for the last three months. This has to be done to account for the temporal resilience required from such models. This method of using a published blacklist introduces another risk, which is the risk of feedback loops, that pass down the same error to the classifier. This is because the list might have some false recognitions for the web pages that are submitted manually by other users. This means that the whole dataset has to be manually checked and such wrong classifications be removed. Though the percentage of such instances are very low the fact that this list has to

be verified manually really is a black mark for this method. Though they achieve an impressive false positive rate of under 0.1%. Even though we consider that the black list is perfect, the fact that the system uses a black list to work means that there will be an inevitable time lag between the time the phishing site is up and that the page is detected to be used for phishing.

This time lag has to be reduced by decreasing the time taken to develop the model and thereby help the users to find even the most recent phishing pages. In spite of the shortcomings that this work had, the features for the model are

1. The URL of the page
2. The HTML page contents
3. The host server details

2.2 CANTINA

This work was further taken up by Guang Xiang, Jason Hong, Carolyn P. Rose and Lorrie Cranor in their CANTINA+ [5] which provides a feature-rich machine learning framework for detecting phishing web sites. It is split into two phases. In the first phase, the task is to find the feature values for all the records in the database. Once this is done, the second phase is to find if the site is used for phishing or not. The above feature is limited to 15 features which are used in both the phases.

This method for performance optimisation and to reduce false positives uses a hash based page removal model in which the similar looking components of the website are removed, making it easier to find the differences. Once, the similar components are removed, the presence of a login form in the HTML content is searched for. If the content matched that of the login page, then the pre-trained model is used.

Since the creation of the model requires an updated list of phish-

ing sites, the list provided by PhishTank's verified blacklist [9] is being used. And as far as the time required to find the similar looking sites is concerned, it is greatly reduced by using the SHA 1 hashing algorithm. It is noted that though this hashing algorithm can be easily broken, it is being used for the efficiency and the high accuracy with which it finds out the phishing sites.

Though this model provides a faster way of finding those sites, it definitely comes with its own set of drawbacks. The first one being that SHA 1 algorithm that defeats the whole purpose of the malicious actor never being able to circumvent the system.

Though this system has it's disadvantages, the features are a few things that can be taken from them. They include the embedded domain, IP address, number of dots in URL, suspicious URL, number of sensitive words in URL, out of position top level domains (TLD), bad forms, bad action fields, non-matching URLs, out of position brand names, the age of domain, page in top search results, page rank and page results while searching for copyright company name and host name. Many models were used and it was found out that the Bayesian models and the Random Forests performed remarkably well.

2.3 AUTO UPDATED WHITELIST

Ankit Kumar Jain and B. B. Gupta provided another approach [7] to protect against phishing attacks at the client side using auto-updated white-list. The accurate and fast detection of phishing sites in a real time environment is paramount. The time constraints of the above methods are because they use a visual similarity based approach. This can be reduced by using a heuristic based approach which depends mainly on the feature set, the classifier and the training data.

The hypothesis is based on the fact that though the phishing pages

look similar to the corresponding real website, they do differ steeply in the functionality they offer. But almost all but the critical phishing functionality redirects to the corresponding real website.

To provide such functionality, a whitelist is used in this method. The whitelist contains the domain name and IP address as the parameters. The whitelist provides for the faster running time and to reduce the false positive rate. The working of the whitelist is as follows.

First when the user has never visited any site, the whitelist has no records. But when the user does so and visits a site, the whitelist is checked if it has the domain along with the IP address. If the record is present, the page is said to be a safe site. Else, the second component which is almost the same as that of the previous models kicks in to find if the requested site is a phishing site or not.

Thus, because of the whitelist the model has the advantage of being language independent and is capable of finding the embedded components in the phishing website that can cause a DDOS attack.

The model was developed further into a usable application down the lane. And this paper provided the following findings. The model provided the base for using auto-updated whitelists to speed up the process of finding out if the page is used for phishing or not. This is highly advantageous because most of the sites that a person accesses will not be a phishing page, significantly reducing the average time taken to process the site, as most of the websites visited by the end user will be safe sites and would not be ones used for phishing and become a threat.

2.4 OFF-THE-HOOK

The work [10] by Samuel Marchal, Giovanni Armano, Tommi Grondahl, Kalle Saari, Nidhi Singh, and N. Asokan implemented the client-side phishing prevention application named Off-the-hook.

The main improvements that they provided were better privacy, realtime protection, resilience to dynamic phishing and effective warnings. It also has an emphasis on making a software application that can be easily used by the general public to protect themselves against phishing.

A brand independent, evolution resistant model to detect the phishing pages has been developed. This means that sites of all domains will be identified correctly and the temporal resistance maintained. This is that even when the malicious actor knows how the system works, and tries to figure out a work around, the system learns and adapts itself to the changing conditions.

The main upside for this project is the cross platform capabilities that it intends to provide and the user base that it can have based on the ease of use that the application provides.

The downside is as follows. All the functionality required cannot be implemented inside the browser using Javascript alone. As a result of this, machine dependence creeps into the equation. This in the long run will be a major block to the ease of use that the application says will provide.

The other main disadvantage is that the model does not take into account the static IP addresses on which the page might be hosted. This model relies on the assumption that such IP addresses that host the phishing pages will be blacklisted and removed by the host provider.

Thus based on all the above mentioned related works, we will be redeveloping the application Off-the-hook to follow the basic networking and socket connections so that the applications both running inside the browser and within the operating system of the user will remain compatible.

2.5 FUZZY ROUGH SET FEATURE SELECTION

The conference paper [12] by Mahdieh Zabihimayvan and Derek Doran on Fuzzy Rough Set Feature Selection to Enhance Phishing Attack Detection gives us the ways of choosing the best possible features for training the model over. It uses the Fuzzy Rough Set (RFS) theory to select the most effective features and finds out that the Random Forest method gave the maximum possible F-score.

2.6 ADDITIONAL WORK

We will be implementing Off-the-hook-plus using programming languages that are designed with support for cross platforms. The Google index and web traffic will be used as additional features to increase the temporal resiliency. And to reduce the memory footprint, the required features will be selected using RFS theory. The target website will also be identified by using the SHA based similarity scores. The addon will also be made to be as lightweight as possible.

2.7 COMPARISON

Table 2.1 gives a comprehensive list of all the papers with their publications and problems solved along with the limitations that these solutions have.

Table 2.1 Comparison Table

Paper	Publication	Solved	Limitations
Large-Scale Automatic Classification of Phishing Pages	Proc. Netw. Distrib. Syst. Security Symp., 2010	Machine learning models can be used with reliable accuracy.	Needs blacklist for updating.
CANTINA: A feature-rich machine learning framework for detecting phishing Web sites	ACM Trans. Inf. Syst. Secur., 2011	SHA1 based similarity check for similar looking sites.	SHA1 could be manipulated.
A novel approach to protect against phishing attacks at client side using auto-updated white-list	EURASIP J. Inf. Secur., vol. 2016, no. 1, Dec. 2016	Auto-updated whitelist for faster detection of sites on average.	Not temporally resilient.
Fuzzy Rough Set Feature Selection to Enhance Phishing Attack Detection	IEEE International Conference on Fuzzy Systems, June 2019	Feature selection.	Not a user oriented application.

CHAPTER 3

SYSTEM DESIGN

This chapter provides the system architecture for Off-the-hook-plus along with the block diagram.

3.1 SYSTEM ARCHITECTURE

In this project we propose Off-the-hook-plus which will find if the website is a phish or not and also provide the link of the website that it is trying to impersonate. The system architecture is given by the Figure 3.1 which shows the different modules that work in tandem to get the functionality required by Off-the-hoo-plus. The web browser is used by the user to browse websites from which the Add-on gets the required data and shares it with the Background Process which finds if the website is a phish or not and also the target website. The Add-on has two components, Background Script which collects the page redirects and the Content Script which gets the page content like the URL and the HTML. This is sent to the dispatcher which checks if the website is a phish with the phish detector which uses a Random Forest Classifier and then calls the Target Identifier to get the original website using the SHA based page simliarity. The result is then displayed in the web browser to notify the user about the website. This will enable the user to use Off-the-hook-plus to find the phish website and the target websites that they impersonate.

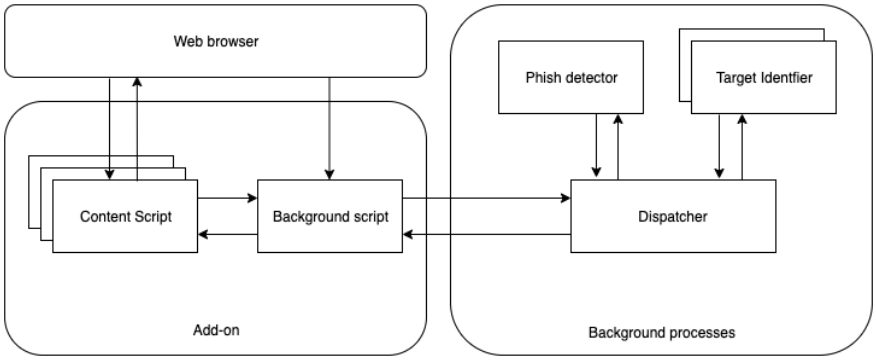


Figure 3.1 System Architecture

3.2 HIGH LEVEL BLOCK DIAGRAM

The high level block diagram of Off-the-hook-plus is shown in the Figure 3.2. This shows a more detailed representation of the modules present in Off-the-hook-plus. Each module has publishers and subscribers to listen to the data transmission events that reduce the memory overhead required. The Dispatcher in the Background Process has the Whitelist manager that takes care of the autoupdated whitelist that is essential for the faster execution times. Finally the OutputUI is the component that is used to display the different types of error messages that must be shown to the user through the web browser. These will be explained in detail in the next chapter.

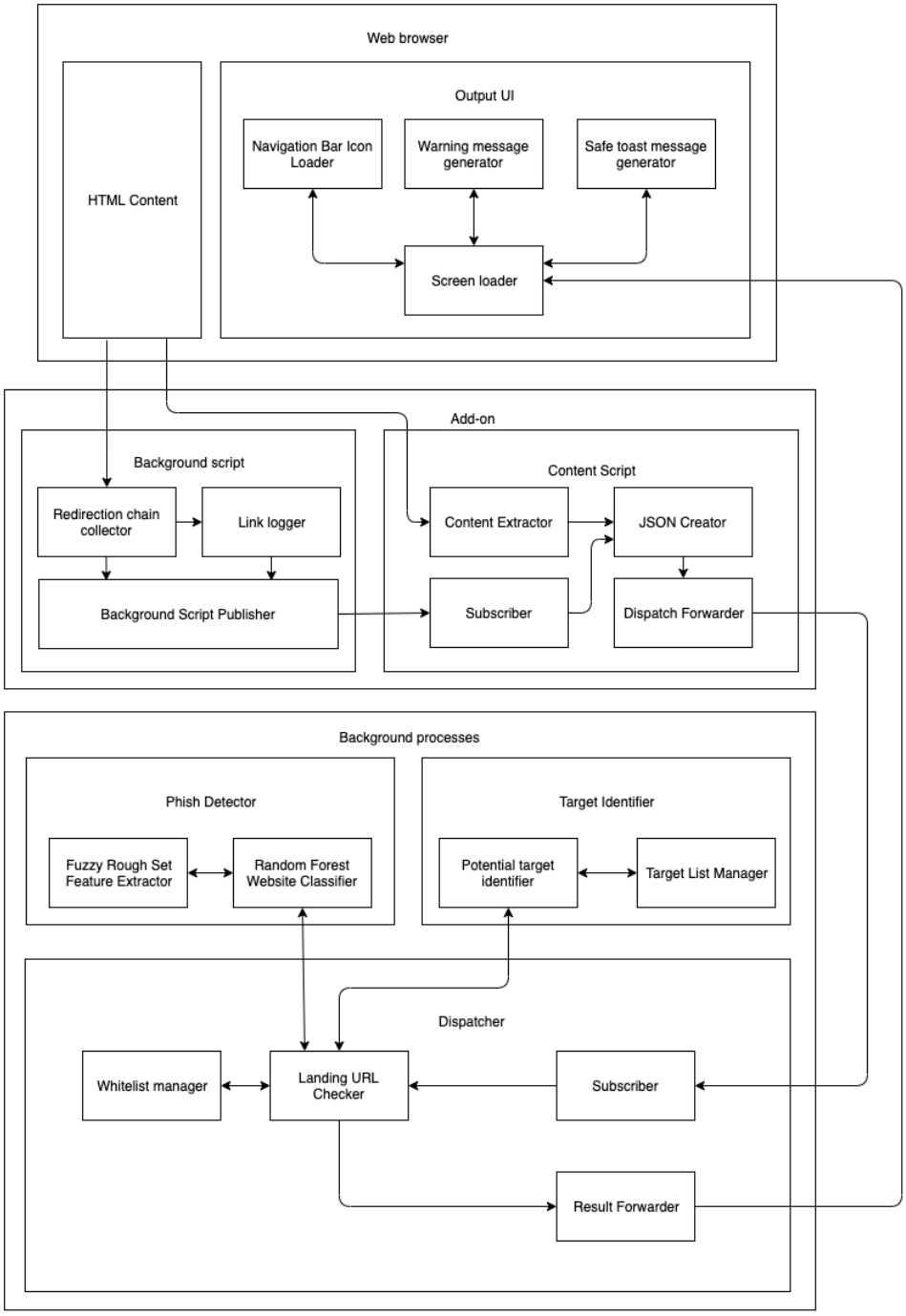


Figure 3.2 High Level Block Diagram

CHAPTER 4

DETAILED MODULE DESIGN

This chapter will explain in detail the modules and submodules that complete the functionality of Off-the-hook-plus.

4.1 MODULES

Off-the-hook-plus has three modules that complete its functionality. They are as follows.

1. Add-on
    a) Background script
    b) Content script
2. Background process
    a) Dispatcher
    b) Phish Detector
    c) Target Identifier
3. Web Browser
    a) HTML content
    b) Output UI

4.2 ADD-ON

An add-on is required in the form of an extension because of the reasons that the frameworks provided by Javascript for tasks related to machine learning are still in its infant stages[11]. And so the add-on performs the two important tasks of getting the page contents from the browsers and also making changes to the elements in the web browser

to convey the threat level for phishing.

4.2.1 BACKGROUND SCRIPT

The task to be done by this component is to get the list of redirection URL chains from the user's web browser, and publish it to the content script while also logging the links for reference purposes.

Begin
For each page load redirect
Add listener to that event
Get the list of redirects from listener
If page is fully loaded
Send the list of redirects to content script
Done
End

The URL items that have to be sent by the background script are as follows.

url: pathItem.url,
status: pathItem.statusLine,
redirectType: pathItem.redirectType,
redirectUrl: pathItem.redirectUrl,
metaTimer: pathItem.metaTimer

4.2.2 CONTENT SCRIPT

The task to be done by this component is to get the landing URL of the page and the contents of the page from the browser and also the URL redirection list from the background script and then combine those and send it to the background process which will have to do further

computation.

Begin
For each page load redirect
If page is fully loaded
Get the URL from the tab
Get the HTML content from innerHTML tag
Get redirection list from background script
Send them to the background process
Done
End

The main functionality of getting the URL and also loading the HTML content is done using the following snippets.

```
//Retrieve URL JS
tablink = tab.url;
//Retrieve Page content PHP
site = _POST['url'];
html = file_get_contents(site);
```

4.3 BACKGROUND PROCESS

The background process gets the contents from the content script and identifies if the site is used for phishing or not. It has the following subcomponents which are discussed in detail.

1. Dispatcher
2. Phish Detector
3. Target Identifier

4.3.1 DISPATCHER

The dispatcher is used for the performance enhancements it provides by using the whitelisted addresses that can be used without even having to run the model. It has direct control over the phish detector and target identifier.

Begin
If page address is in whitelist
Send the GREEN signal
Else
Send content to phish detector
Get results from phish detector
If phish is FALSE
Send the GREEN signal
Else
Send the RED signal
Send content to target identifier
If target is found
Publish target
Else
No target matched
End

4.3.2 PHISH DETECTOR

The phish detector gets the content from the dispatcher and gives the result which is either the site is a phish or not. It is done by using a machine learning model.

Begin
For each page URL

Get the fuzzy set feature values for the URL

Load the saved random forest model

Publish the result

Done

End

The features used are as follows,

1. Have IP address
2. URL length
3. Shortening service
4. Having @ symbol
5. Double slash redirecting
6. Prefix suffix
7. Having sub domain
8. Domain registration length
9. Favicon
10. HTTPS token
11. Request URL
12. URL of anchor
13. Links in tags
14. Server form handler
15. Submitting to email
16. Abnormal URL
17. iFrame redirection
18. Age of domain
19. Web traffic
20. Google index
21. Statistical Reports

The following is used to extract the features using the Fuzzy

Rough Set Theory.

Begin

Compute indiscernibility matrix M(A)

Reduce M using absorption laws

d - number of non-empty fields

Initialise all fields

For all fields

Compute fields using formulas R=SUT

Done

End

The model is a Random Forest Classifier which has the pseudocode as follows.

Begin

For each record in dataset

Get the fuzzy set feature values

Create an arff file to save results

Done

Train the dataset with at least 7 splits as random forest

Save the model as pkl file

End

4.3.3 TARGET IDENTIFIER

Once the dispatcher gets the signal that a site is phishing, it can be useful to find which site is being used as a template so that the unsuspecting user is fooled. This is done by using the similarity of hashes between the phishing and target website. The SHA algorithm is as follows.

Begin

Input is an array 8 items long where each item is 32 bits.

Calculate all the function boxes and store those values.

Store input, right shifted by 32 bits, into output.

Store the function boxes.

Store (Input H + Ch + ((Wt+Kt) AND $2\hat{3}1$)) AND $2\hat{3}1$ As mod1

Store (sum1 + mod1) AND $2\hat{3}1$ as mod2

Store (d + mod2) AND $2\hat{3}1$ into output E

Store (MA + mod2) AND $2\hat{3}1$ as mod3

Store (sum0 + mod3) AND $2\hat{3}1$ into output A

Output is an array 8 items long where each item is 32 bits.

End

4.4 WEB BROWSER

Though the above described components play major roles in Off-the-hook-plus, the one that the user will be able to view is this component. And so care has to be taken to make it look as professional as possible.

4.4.1 HTML CONTENT

The add on must be published as extensions for the browsers and so the UI of the components must be taken care of. And the tasks of the background scripts must run as helper tasks and not interfere with the main script, otherwise the UI of the extension will appear to be jittery.

4.4.2 OUTPUT UI

The two possible results for Off-the-hook-plus are that either the site is a phish or not. And if a site is not a phish no changes have to be made to notify the user. But if the site is found out to be a phish the

changes made to the UI must meaningfully convey to the user that the site is a phish.

```
Begin
    If site is phish
        Change icon to red
        Display warning message
        If site has target
            Display target link
        Else
            Display no target
        If site has target
    Else
        Change icon to green
        Change icon to green
    Display safe to proceed message
End
```

CHAPTER 5

IMPLEMENTATION

This chapter will discuss in detail how Off-the-hook-plus is implemented along with the key results and snapshots for better understanding.

5.1 ADD-ON

The add-on has the following components that were implemented using javascript and php as a Google Chrome extension.

5.1.1 CONTENT SCRIPT

This component takes the input which is the URL of the page and it's contents and sends it to the background process along with the redirection URL from the background script. A design of this component is required as the Chrome Extension cannot get the tab HTML content in the background and so a script in PHP is used to get the content of the URL from the Javascript code segment. Figure 5.1 shows the content script.

tablink = tab.url;

$html = file_get_contents($site);

5.1.2 BACKGROUND SCRIPT

This script is based on the open sourced code on GitHub[3] that demos how to get the background URL redirects of the current tab. It

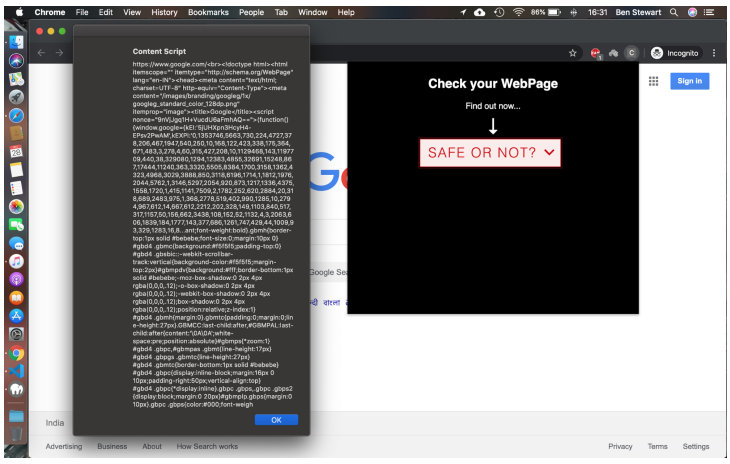


Figure 5.1 Content Script

handles multiple types of redirects and also their security levels based on the URL redirects. This component finally returns the list of path components that the page had traversed through. Figure 5.2 shows the background script contents. It has the path items, which are,

1. URL
2. Status
3. Redirect type
4. Meta timer

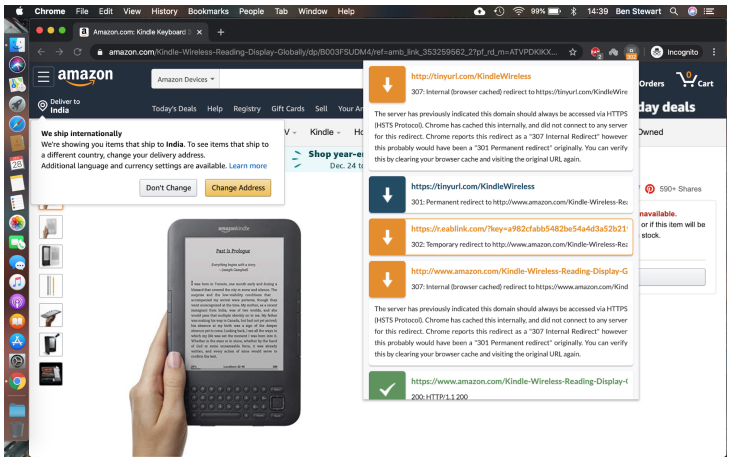


Figure 5.2 Background Script

5.2 BACKGROUND PROCESS

This component receives the data from the add-on and orchestrates the decision making using the dispatcher.

5.2.1 PHISH DETECTOR

The dataset used for this project was scraped from PhishTank[9] and has the records for 23827 such phishing sites with 30 features for each one of them. As a model using all features will take more time to execute, the most important features have to be extracted.

The phish detector is a random forest model trained using the features derived from the Fuzzy Rough Set Theory for feature selection. A few roadblocks faced while developing this model were that the feature selection kit required the data set to be as integers.

The base component was using the scikit roughsets package available in Python. The code is as follows and the output features were used to train the random forest model and provides the output as shown in Figure 5.3.

selector = RoughSetsSelector()

X_selected = selector.fit(X, y).transform(X)

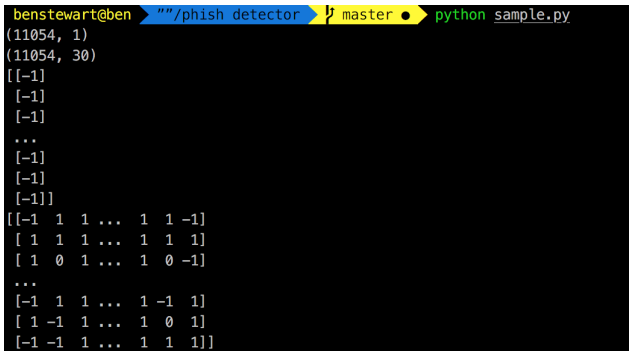


Figure 5.3 Feature Selection

The model was generated using the following code and stored as a pickle in the file.

//create model

clf4=RandomForestClassifier(min_samples_split=7)

clf4.fit(features_train, labels_train)

//save the model

joblib.dump(clf4, 'classifier/random_forest.pkl', compress=9)

Figure 5.4 gives the confusion matrix and the feature weightage score using the code as follows.

//feature weightage

*importances = clf4.feature_importances_*

//confusion matrix

print metrics.confusion_matrix(labels_test, pred4)

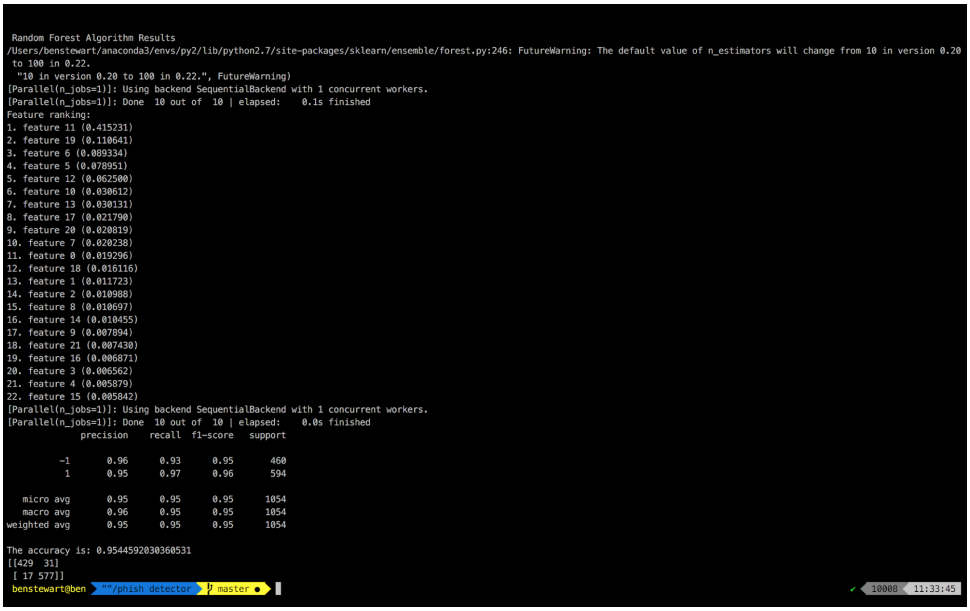


Figure 5.4 Random Forest Model

5.2.2 TARGET IDENTIFIER

The target identifier uses the following code to find the similarity between two web pages using the following code.

```
tags1 = get_tags(lxml.html.parse(path1))
tags2 = get_tags(lxml.html.parse(path2))
diff = difflib.SequenceMatcher()
diff.set_seq1(tags1)
 diff.set_seq2(tags2)
```

The dispatcher once it confirms that the site is a phish from the phish detector and then writes the url into the single-sites.json file. The url is taken from there and scraped along with the html content and is then compared using the above method.

```
params['url'] = url
response = requests.get(url, headers=headers, params=params)
```

5.2.3 AUTO UPDATED WHITELIST

The whitelist is automatically updated by the dispatcher if the url is detected to be safe using the following code.

```
//insert into whitelist
white_list_file=open('whitelist.txt', "a+")
white_list_file.write(url)
```

The whitelist is searched for before using the ML model to detect if the site is phishing or not, to save execution time.

```
white_list_file = open('whitelist.txt').read()
white_list = white_list_file.split('
n')
```

if url in white_list:

#url is safe

Figure 5.5 gives the shortening of execution time as the url is stored in the whitelist.

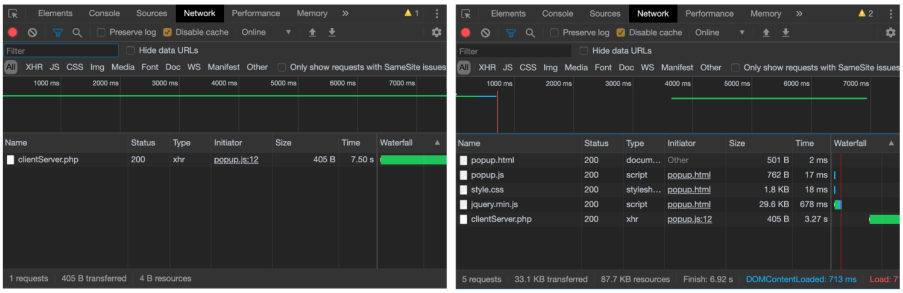


Figure 5.5 Execution Time with Auto Updated Whitelist

5.2.4 DISPATCHER

The dispatcher is called using the php script of the content script using the following statement.

$decision=exec("python test.py $site 2>&1 ");

echo $decision;

The command is run in the default shell of the machine and the output is passed to the Output UI of the add-on that reflects the same in the web browser.

5.3 WEB BROWSER

This component is what takes care of how and what is displayed to the end user when the application is being used.

5.3.1 OUTPUT UI

The content from the dispatcher is loaded into the Google Chrome Extension using the Output UI which has the content loaded into the div that has the HTML, JS and CSS designed for it upfront as shown in the Figure 5.6.

$("#div1").text(xhr.responseText);

Figure 5.6 Output UI

5.3.2 HTML CONTENT

The web browser must also take care of the whole extension so that the components do not get hidden and also are not visible to the user. The following css snippet makes sure the above holds good as shown in the Figure 5.7.

```
body{
    width:500px;
    height:100px;
    display:inline-block;
    align-items: center;
}
```

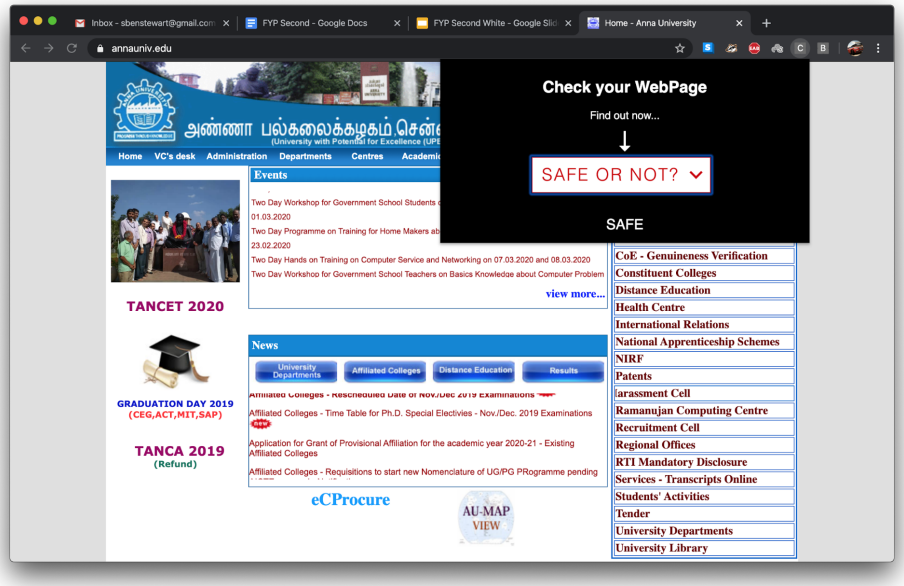


Figure 5.7 HTML Content

CHAPTER 6

EVALUATION METRICS

Off-the-hook-plus must be evaluated for the following performance metrics, which become important as this is meant for users who will never need to know about computer programming. The metrics used here are as follows.

1. Phish detection accuracy
2. Target detection ratio
3. Memory usage profiling
4. Add-on rendering time
5. Temporal resilience accuracy

6.1 PHISH DETECTION ACCURACY

Phish detection accuracy is really important because the main task for this application is to notify the people if the site is a phish or not. It is defined as the ratio of the total number of correct classifications to the total number of classifications.

$Accuracy = (TP+TN)/(TP+TN+FP+FN)$

The Figure 6.1 gives the phish detection accuracy along with the precision, recall and f1-score. It also has the confusion matrix that gives a visual representation of how the model tags the web pages.

Figure 6.1 Phish Detection Confusion Matrix and Metrics

6.2 TARGET DETECTION RATIO

Target detection ratio is to measure the ease of use for the user by providing them with the original site which is being mimicked by the phish. This will be the ratio of phish sites whose target has been found to that of the total number of phish sites.

Detection Ratio = (TP)/(TP+TN+FP+FN)

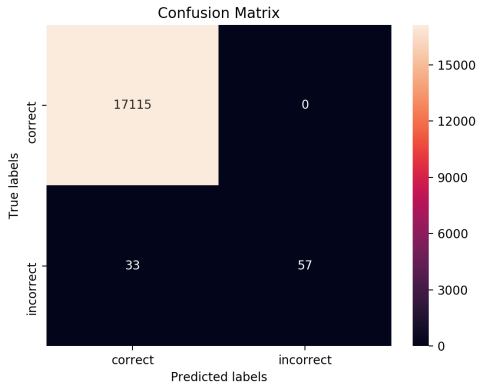


Figure 6.2 Target Detection Confusion Matrix

The Figure 6.2 gives the confusion matrix for the target identifier, from which the target detection ratio turns out be 0.994, which is pretty good a number, considering how difficult the page matching is.

6.3 MEMORY USAGE PROFILING

Since memory usage profiling is an application to be used by many people who might have different configurations of machines, we must take into consideration that the application must use as little memory as possible.

Current total memory usage = Total Memory - (Free + Buffers + Cached)

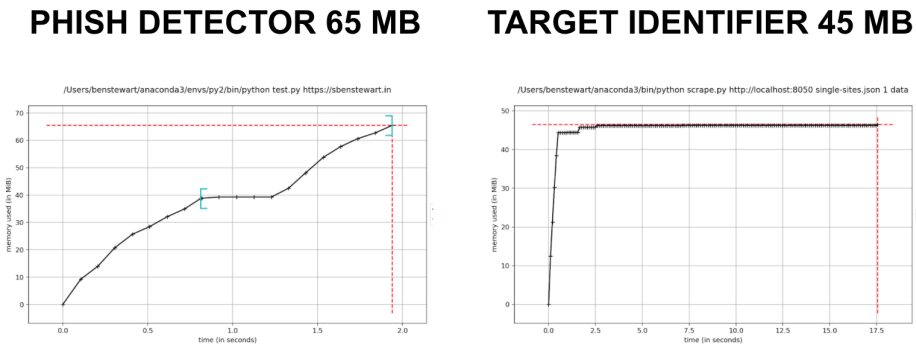


Figure 6.3 Background Process Memory Usage

The Figure 6.3 gives the average memory usage for the dispatcher along with the phish detector and the target identifier which in total accounts to 110MB which is the average memory used by an application on 64 bit systems.

The Figure 6.4 gives the average memory usage for the add on which includes both the content script and the background script, which is 1.9MB. This is a very reasonable memory requirement for the add-on. Thus in total, the application requires 111.9MB on average.

6.4 ADD-ON RENDERING TIME

Since the add-on has a background component which has to collect the data from multiple tabs that could be running simultaneously, the

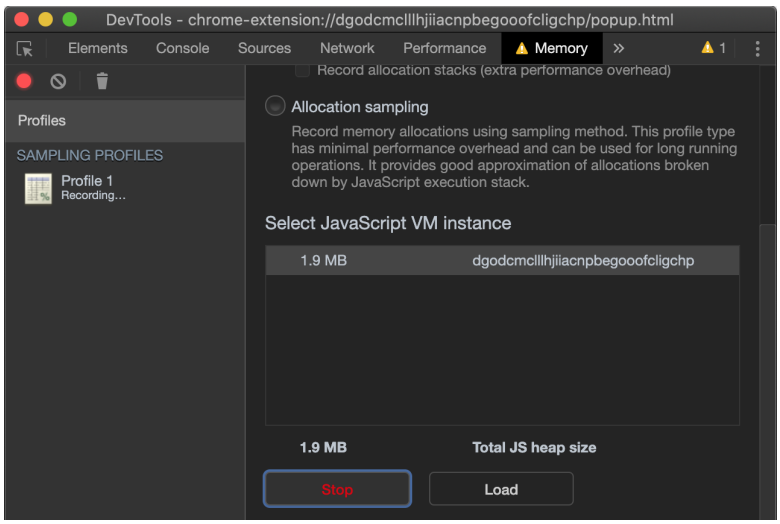


Figure 6.4 Addon Memory Usage

add-on must be tested to check if it is stable and does not take time to render and thereby blink.

Rendering Time = End time - Start time

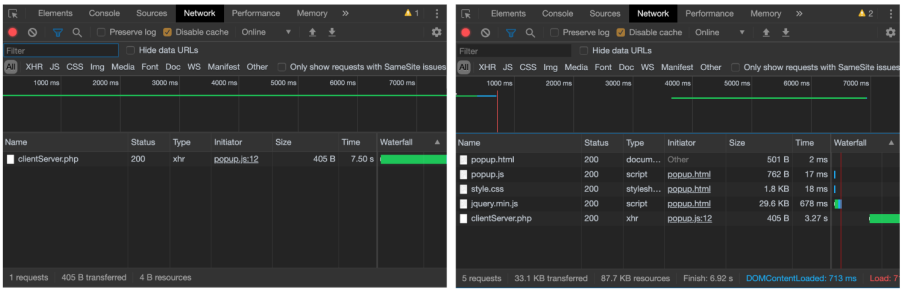


Figure 6.5 Addon Rendering Time

The Figure 6.5 gives the times that it takes for the first time which is 7.5 seconds and for the subsequent times, it drops down to 3.27 seconds which is a bit more than a 50 percent reduction in the execution time. The main reason for this reduction is the auto updated whitelist.

6.5 TEMPORAL RESILIENCY ACCURACY

Temporal resilience accuracy is a metric which says that the application must be resilient to adaptations by the malicious actor, over time. Thus this can be measured by checking if the accuracy does not reduce with the passage of time.

Accuracy = (TP+TN)/(TP+TN+FP+FN)

Figure 6.6 Phish Detection Confusion Matrix and Metrics

The Figure 6.6 gives the phish detection accuracy along with the precision, recall and f1-score for the model run after two months of the first instance being run whose results are in Figure 6.1. The accuracy drop from the first run is 0.0035. This makes it almost temporally resilient.

6.6 COMPARISONS

In Table 6.1 we will look into how our system compares with the systems implemented in the previous papers. The metrics that will be compared are the phish detection accuracy, memory usage, execution time and temporal resiliency. The metric of target detection has been omitted because those papers did not implement such a subsystem.

Table 6.1 Metrics Comparison Table

Metric	Previous Benchmark	Off-the-hook-plus
Phish Detection Accuracy (Percent)	95.54	96.11
Memory Usage (MB)	295	111.9
Execution Time (seconds)	8.74	7.5
Temporal Resiliency (Percent Decrease)	6.44	0.35

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

Off-the-hook-plus is a system that can be used on all platforms for detecting phishing sites has been developed. This works totally on the client side without the need for heavy computational resources that use servers. It is also very easy install and use on all machines even on ones that run short on main memory as it requires just around 120MB at peak times. The main benefit of this system is that it will in most cases find the target site which has been modified as a phish by the malicious actor. Thus the user will know the correct website to access as well.

7.2 FUTURE WORK

Will work on the improvement of the data transfer mechanisms between the modules, which will greatly reduce the resource constraints as all modules will not be required to run all the times. We will also work on improving the accuracy of the phish detector, while making the target identifier more robust and able to identify a lot more websites.

APPENDIX A

OUTPUT UI

The Output UI is the most important component that provides the interface between this system and the user. And so, all the possible outputs of this module needs explaining.

A.1 SAFE WEBSITE

As most of the sites that will be visited by a user will be safe, unless the search engine that the user uses has been compromised. The above holds good statistically. And, in the Figure A.1, the Output UI for the official university website is shown.

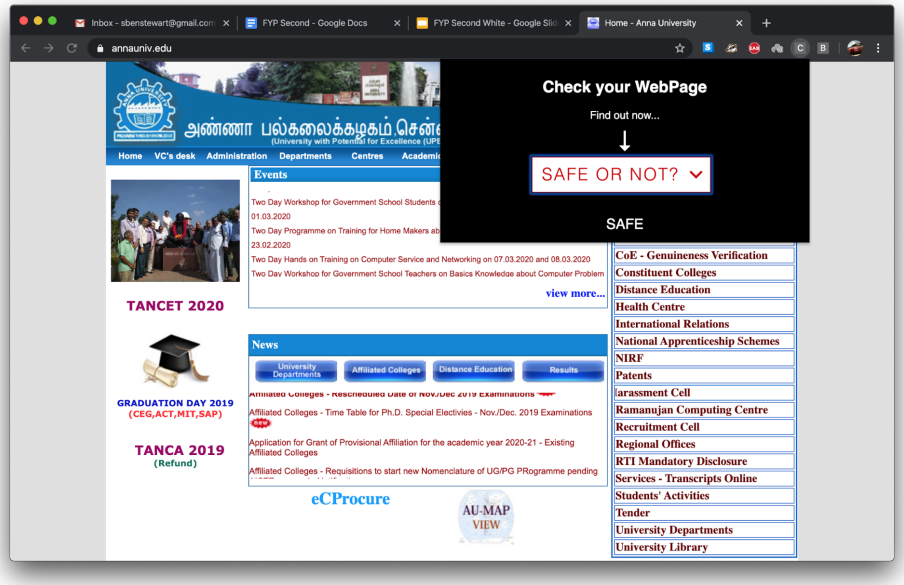


Figure A.1 SAFE Website

A.2 PHISHING WEBSITE

This type of websites make the smallest part of websites that the user visits. There are multiple different outputs possible for this scenario, which will be discussed in detail.

A.2.1 CORRECT PHISH DETECTION AND CORRECT TARGET IDENTIFICATION

The phish detector finds the phish correctly, and the target detector which displays the most closely related site is the same website that the phish is based on. This is shown in the Figure A.2, where the phish website impersonates Twitter, which both the phish detector and target identifier find correctly.

Figure A.2 Correct Phish Detection and Correct Target Identification

A.2.2 CORRECT PHISH DETECTION AND INCORRECT TARGET IDENTIFICATION

The phish detector finds the phish correctly, but the target detector which displays the most closely related site is not the same website that the phish is based on. This is shown in the Figure A.3, where the phish website impersonates FaceBook, but the target identifier could not

classify it.

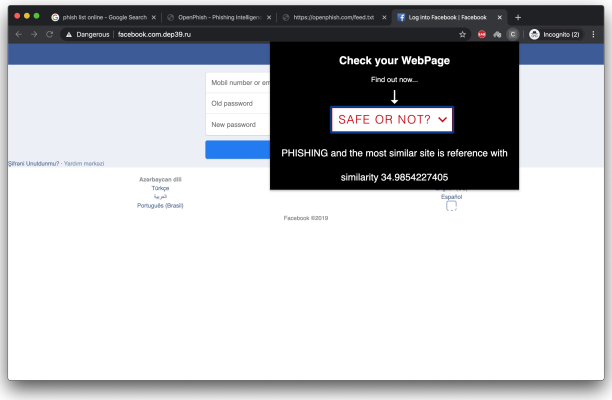


Figure A.3 Correct Phish Detection and Incorrect Target Identification

A.2.3 INCORRECT PHISH DETECTION

The phish detector finds the phish incorrectly. This is a problem, and so the model has been skewed so that the incorrect classifications will be only where the safe websites are incorrectly classified as phish. Figure A.4 shows the instance where the phish detector incorrectly classifies the personal website of S.Ben Stewart who worked on this system as a phish. The future work involves rooting out cases like this.

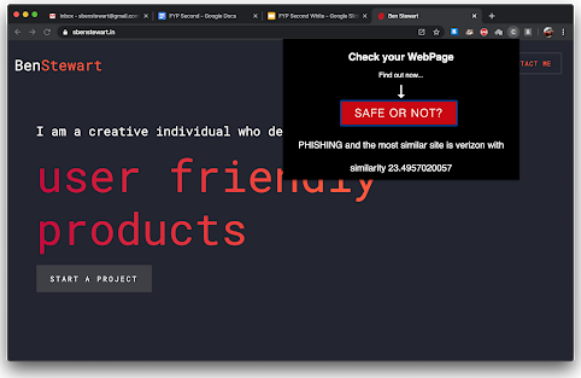


Figure A.4 Correct Phish Detection and Incorrect Target Identification

REFERENCES

[1] George Akerlof and Robert J Shiller, *Phishing for Phools: The Economics of Manipulation and Deception*, Princeton University Press, 2015.

[2] Whittaker C, Ryner B, and Nazif M, "Large-scale automatic classification of phishing pages", *Proc. Netw. Distrib. Syst. Security Symp.*, vol. 1, pp. 1–14, 2010.

[3] ccampbell, "Chrome extension for server side console logging", https://github.com/ccampbell/chromelogger, 2019.

[4] Web Foundation, "History of the web – world wide web foundation", https://webfoundation.org/about/vision/history-of-the-web/, 2018.

[5] Xiang G, Hong J, Rosé C P, and Cranor L, "CANTINA: A feature-rich machine learning framework for detecting phishing web sites", *ACM Trans. Inf. Syst. Secur.*, vol. 14, num. 2, 2011.

[6] Google, "Implementation for the usage of google safe browsing APIs (v4)", https://github.com/google/safebrowsing, 2019.

[7] Jain A K and Gupta B B, "A novel approach to protect against phishing attacks at client side using auto-updated white-list", *EURASIP J. Inf. Secur.*, vol. 2016, num. 1, 2016.

[8] Steven Levy, *Hackers: Heroes of the Computer Revolution*, Doubleday, 1984.

[9] OpenDNS, “Phishtank — join the fight against phishing”, https://www.phishtank.com/, 2020.

[10] Marchal S, Armano G, Gröndahl T, Saari K, Singh N, and Asokan N, “Off-the-hook: An efficient and usable client-side phishing prevention application”, *IEEE Trans. Comput.*, vol. 66, num. 10, pp. 1717–1733, 2017.

[11] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, Stan Bileschi, Michael Terry, Charles Nicholson, Sandeep N Gupta, Sarah Sirajuddin, Rajat Sculley D, Monga, Greg Corrado, Fernanda B Viégas, and Martin Wattenberg, “Tensorflow.js: Machine learning for the web and beyond”, 2019.

[12] Mahdieh Zabihimayvan and Derek Doran, “Fuzzy Rough Set Feature Selection to Enhance Phishing Attack Detection”, *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, vol. 1, 2019.