Study on Website Phishing and their Countermeasures

Thesis

Submitted in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

Routhu Srinivasa Rao

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGALORE - 575 025

June, 2020

DECLARATION

by the Ph.D. Research Scholar

I hereby declare that the Research Thesis entitled **Study on Website Phishing and their Countermeasures** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfilment of the requirements for the award of the Degree of **Doctor of Philosophy** in Department of Computer Science and Engineering is a bonafide report of the research work carried out by me. The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.

Routhu Srinivasa Rao, 158009CS15FV13
Department of Computer Science and Engineering

Place: NITK, Surathkal.

Date: June 3, 2020

CERTIFICATE

This is to certify that the Research Thesis entitled **Study on Website Phishing and their Countermeasures** submitted by **Routhu Srinivasa Rao** (Register Number: 158009CS15FV13) as the record of the research work carried out by him, is accepted as the Research Thesis submission in partial fulfillment of the requirements for the award of degree of **Doctor of Philosophy**.

Dr. Alwyn R Pais
Research Guide

(Signature with Date and Seal)

Chairman - DRPC
(Signature with Date and Seal)

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my Supervisor Dr. Alwyn R Pais, Associate Professor in Department of Computer Science & Engineering for his continuous encouragement, patience, motivation, enthusiasm, and immense knowledge. His guidance and insightful comments helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for this thesis.

My sincere thanks go to research progress committee members Dr. Mohit P Tahiliani, Assistant Professor in Department of Computer Science & Engineering, and Dr. Nagendrappa H, Assistant Professor in Electrical & Electronics Engineering, for giving their valuable suggestions, inspiration and moral support while evaluating our work time to time. I am also grateful to all office staff members of Computer Science & Engineering Department for their generous support throughout this work.

My greatest appreciation goes to my close friends Vaishnavi, Nikhil, Alok, Apurva, Ajnas, Somesh, Siva and Zubair for questioning me about my ideas, helping me and even for hearing my problem. I will always remain grateful to them for being a great reliable person to whom I could always talk about my problems and excitements. Most importantly, I would like to thank my family, especially my beloved parents for their continuous love and unconditional support not only for this work but also throughout my life.

Routhu Srinivasa Rao

ABSTRACT

Phishing is one of the manipulation technique which targets naive online users tricking into revealing sensitive information such as username, password, social security number or credit card number etc. Attackers fool the Internet users by masking webpage as a trustworthy or legitimate page to retrieve personal information. There are many anti-phishing solutions such as blacklist or whitelist, heuristic and visual similarity based methods proposed till date to prevent the phishing attacks. But online users are still getting trapped into revealing sensitive information in phishing websites. In this research work, we focus on designing new heuristic techniques with comprehensive feature set and different machine learning algorithms for the classification of phishing sites.

There exists many machine learning (ML) based techniques to detect the phishing sites but they do not achieve better detection accuracy. To overcome the disadvantages of existing schemes, we have presented an efficient feature-based machine learning framework for the detection of phishing sites. The feature set is collected from different resources such as URL, source code and third party services and fed to the machine learning classifier. The model achieved a significant accuracy of 99.55% using orthogonal Random Forest classifier with a True Positive Rate (TPR) of 99.45% and True Negative Rate (TNR) of 99.42%.

Although ML-based technique achieved a significant accuracy but due to the use of third-party services such as search engine or page ranking services the technique might fail when phishing sites hosted on compromised servers (PSHCS) are encountered. To counter these PSHCS, we presented two techniques with and without third party services. Firstly, we present a novel heuristic technique using twin support vector machine (TWSVM) to detect malicious registered phishing sites and also sites which are hosted on compromised servers. This technique achieved an accuracy of 98.4% in detecting phishing sites with TPR of 98.72% and TNR of 98.08%. This technique relies on the home page of the suspicious site for calculating the similarity score between the home page and suspicious site. This mechanism might fail when the correct home page of the

suspicious site is not retrieved. Hence, we presented an improved search engine based technique to identify the matched page for the suspicious site with a dynamic search query to calculate the similarity score. This technique not only detects PSHCS but also detects the newly registered legitimate site. The technique achieved an accuracy of 98.61% with TPR of 97.77% and TNR of 99.36%.

The above presented techniques rely on the source code of the website and third party services which needs loading the page for detecting the status of the website. Due to this, the response time of the detection process might get delayed at the client-side. Moreover, due to guaranteed visit of webpage, there might be a more chance of accidental download of malware from the webpage (drive-by-downloads). Hence, we proposed two lightweight techniques based on the inspection of URLs. These techniques are designed to use as first-level filtering of phishing websites without even visiting the suspicious site. The first technique is deployed as a web application which uses hand-crafted and Term-Frequency Inverse Document Frequency features for the detection. The technique achieved an accuracy of 94.26% with TPR of 93.31% and TNR of 96.65%. The second technique is designed for the mobile device where a multi-model ensemble of Long Short Term Memory and Support Vector Machine is presented for the phishing detection. This technique achieved an accuracy of 97.30% with TPR of 97.31% and TNR of 97.28%.

The earlier presented techniques either used content or URLs for the phishing detection but they lack the information of target website of the designed phishing site. To offer the same, we presented a lightweight visual similarity-based approach which maintains fingerprints of blacklisted phishing sites along with their target legitimate domains. Also, the technique includes heuristic features for the detection of phishing sites targeting non-whitelisted legitimate sites. This technique achieves a significant accuracy of 98.72% with TPR of 98.51% and TNR of 98.87%.

CONTENTS

List of Figures viii

List of Tables x

List of Abbreviations xi

1 Introduction 1

1.1 What is a phishing attack 1

1.1.1 What is zero-day phishing attack 4

1.1.2 A Phishing example 4

1.2 Why users fall for Phishing 5

1.3 Types of Phishing 6

1.4 Motivation 9

1.5 Problem Description 10

1.6 Objectives 10

1.7 Thesis Contributions 11

1.8 Thesis Organization 12

2 Literature Review 13

2.1 Types of anti-phishing techniques 13

2.1.1 List based techniques 13

2.1.2 Heuristic and Machine Learning based techniques 16

2.1.3 Search engine based techniques 21

2.1.4 Visual Similarity based techniques 23

2.1.5 Mobile based phishing detection 25

2.2 Evaluation Metrics 28

2.3 Research Gaps 32

2.4 Summary 34

3 A comprehensive feature based machine learning framework for the detection of phishing sites 35

3.1 Introduction . . . 35

3.2 Methodology . . . 38

3.2.1 Feature Extractor . . . 38

3.2.2 Machine Learning algorithms . . . 49

3.3 Experimentation and Results . . . 49

3.3.1 Tools used . . . 50

3.3.2 Dataset used . . . 50

3.3.3 Experimental evaluation . . . 50

3.4 Discussion . . . 55

3.5 Summary . . . 58

4 Detection of phishing sites hosted on compromised servers 59

4.1 Introduction . . . 60

4.2 Curb-Phish : A heuristic technique to detect PSHCS . . . 62

4.2.1 Methodology . . . 63

4.2.2 Experimentation and Results . . . 72

4.2.3 Discussion . . . 80

4.3 Jail-Phish : An improved search engine based technique to detect PSHCS 83

4.3.1 Methodology . . . 87

4.3.2 Experimentation and Results . . . 94

4.3.3 Discussion . . . 105

4.4 Summary . . . 109

5 URL based Phishing detection 111

5.1 Introduction . . . 112

5.2 CatchPhish : Detection of phishing websites by inspecting URLs . . . 114

5.2.1 Methodology . . . 114

5.2.2 Experimentation and Results . . . 122

5.2.3 Discussion . . . 127

5.3 PhishDump : A multi-model ensemble based technique for the detection of phishing sites in mobile devices . . . 130

5.3.1 Methodology 132
5.3.2 Experimentation and Results 138
5.3.3 Discussion 143
5.4 Summary 147
6 A lightweight visual similarity based approach for the detection of phishing sites 149
6.1 Introduction 150
6.2 Methodology 152
6.2.1 Blacklist filtering 152
6.2.2 Heuristic based filtering 159
6.3 Experimentation and Results 164
6.4 Discussion 171
6.4.1 BlackPhish: Chrome Extension 172
6.4.2 Effectiveness of BlackPhish 173
6.5 Summary 173
7 Conclusions and Future Scope 177
Bibliography 182
Publications 199

LIST OF FIGURES

1.1 Comparison of Phishing attacks during 2015 vs 2016 3
1.2 Phishing attacks from 2010 to 2018 3
1.3 Phishing Website targeting legitimate EBAY 5
1.4 Legitimate EBAY Website 5
1.5 Phishing website of PayPal with HTTPS protocol 7
1.6 Screenshot of Compromised domain and a Phishing page 8
1.7 Lifespan of phishing site in PhishTank 8

2.1 Classification of website phishing defense mechanisms 14

3.1 Architecture of the system 38
3.2 Phishing site with null links, footer and copyright section 46
3.3 (a) Background image used for phishing, (b) Phishing site without background image, (c) Fully developed phishing site along with its source code 48
3.4 Performance of individual features 56

4.1 Distribution of Phishing Kit 61
4.2 Comparison of Phishing attacks during 2009 vs 2014 61
4.3 Architecture of Curb-Phish 64
4.4 Performance of our system with different feature sets 76
4.5 Classification accuracy of individual features 78
4.6 Comparison of our work with existing works EW1 and EW2 79
4.7 Output of our extension 81
4.8 Phishing site hosted on compromised server 84
4.8 Phishing site hosted on compromised server 85

4.9 High Alexa ranked legitimate site and its absence in search results when queried with `domain + title` 86
4.10 Architecture of Jail-Phish 88
4.11 Comparison of TNR at different Threshold 92
4.12 Phishing site hosted on compromised server and indexed by search engine 100
4.13 Gain in TPR and TNR with existing works M1 and M2 104
4.14 Output of Jail-Phish extension 107

5.1 Architecture of CatchPhish 116
5.2 User interface of CatchPhish 128
5.3 Performance of CatchPhish with the inclusion and the exclusion of proposed features 130
5.4 Architecture of LSTM Cell 133
5.5 Architecture of the multi-model ensemble 137
5.6 Calculation of Threshold for top LSTM models 142
5.7 ROC of PhishDump with existing works 144
5.8 UI of PhishDump 145

6.1 Flowchart of the BlackPhish 153
6.2 Flowchart of the Blacklist Filtering 154
6.3 Flowchart of the Heuristic based Filtering 155
6.4 UI of BlackPhish 175

LIST OF TABLES

2.1 Phishing factor indicators 26
2.2 Comparison of existing anti-phishing techniques 26
2.3 Comparison of existing mobile based anti-phishing techniques 29
2.4 Summary of anti-phishing techniques 30

3.1 Dataset used in our model 51
3.2 Evaluation of heuristics features including third-party service features . 52
3.3 Evaluation of heuristics excluding third-party service features 52
3.4 Evaluation of our model with only third-party service features 53
3.5 Comparison of our work with other techniques 54
3.6 Comparison of RF with oRF 55

4.1 Working of our model 65
4.2 Tuning parameters for the experiments 73
4.3 Source of websites 73
4.4 Results of Experiment-1 75
4.5 Evaluation of similarity based features 77
4.6 Collected dataset information 95
4.7 Evaluation of Jail-Phish with LD1, LD2 and LD3 and comparison with M1 and M2 98
4.8 Evaluation of Jail-Phish with PD1, PD2 and comparison with M1 and M2 101
4.9 Evaluation of Jail-Phish with and without similarity computation module 102
4.10 All metrics comparison of our with existing works 105
4.11 Average time elapsed for the detection process 106

5.1 Components of URL with an example 115

5.2 List of features . . . 118
5.3 Phish-Hinted words . . . 120
5.4 Dataset used in our experimentation . . . 124
5.5 Evaluation of FS3 on D3 with various classifiers . . . 124
5.6 Evaluation of handcrafted features using RF classifier . . . 124
5.7 Evaluation of TF-IDF features using RF classifier . . . 125
5.8 Evaluation of hand-crafted and TF-IDF features using RF classifier . . . 126
5.9 Comparison of our work with existing technique . . . 126
5.10 Results of CatchPhish model on D4 and D5 . . . 127
5.11 Average response time for different input size . . . 129
5.12 Accuracy of individual features . . . 129
5.13 Dataset used for experimentation . . . 139
5.14 Evaluation of LSTM word-level and character-level models . . . 140
5.15 Evaluation of ML classifiers with LSTM features . . . 141
5.16 Evaluation of multi-model ensemble on D1-D5 . . . 142
5.17 Comparison of our work with existing techniques . . . 143
5.18 Comparison of our work with existing techniques . . . 146

6.1 Hand-crafted features . . . 160
6.2 Dataset used for experimentation . . . 164
6.3 Calculation of Threshold for simhash_noisy generation . . . 166
6.4 Calculation of Threshold for simhash_pt blacklist generation . . . 166
6.5 Calculation of Threshold for phash_img blacklist generation . . . 166
6.6 Effectiveness of Blacklist filtering . . . 167
6.7 Evaluation of URL features . . . 168
6.8 Evaluation of Source code-based features . . . 168
6.9 Evaluation of URL and Source code-based features . . . 169
6.10 BlackPhish with heuristic and blacklist filtering . . . 169
6.11 Comparison of our work with existing works . . . 171

LIST OF ABBREVIATIONS

Abbreviations	Expansion
APWG	Anti-Phishing Work Group
CSS	Cascading Style Sheets
DL	Deep Learning
DNS	Domain Name System
DOM	Document Object Model
FNR	False Negative Rate
FPR	False Positive Rate
GCSE	Google Custom Search Engine
GWS	Google Web Search
HTML	Hyper Text Mark-up Language
HTTPS	Hyper Text Transfer Protocol Secured
IP	Internet Protocol
LSTM	Long Short Term Memory
MCC	Matthews Correlation Coefficient
ML	Machine Learning
NER	Named Entity Recognition
PSHCS	Phishing Sites Hosted on Compromised Servers
RF	Random Forest
SERP	Search Engine Results Page
SSL	Security Socket Layer
SVM	Support Vector Machine
TF-IDF	Term Frequency - Inverse Document Frequency
TLD	Top Level Domain

Abbreviations	**Expansion**
TNR	True Negative Rate
TPR	True Positive Rate
TWSVM	TWin Support Vector Machine
URL	Uniform Resource Locator
WEKA	Waikato Environment for Knowledge Analysis
XGBoost	eXtreme Gradient Boosting

CHAPTER 1

INTRODUCTION

1.1 WHAT IS A PHISHING ATTACK

In this cyber world, most of the people communicate with each other either through a computer or any digital device connected over the Internet. The number of people using e-banking, online shopping and other online services have been increasing due to the availability of convenience, comfort and assistance. An attacker takes this situation as an opportunity to gain money or fame and steals sensitive information needed to access online service websites. Attackers believe that compromising human is easier than compromising hardware/software of a system and Phishing is one of the ways to deceive the users. It is carried out with a mimicked page of a legitimate site directing online user into providing sensitive information. The term phishing is derived from the concept of 'fishing' for victims sensitive information (Ollmann 2004). The attacker sends a bait as mimicked webpage and waits for the outcome of sensitive information. The replacement of 'f' with 'ph' phoneme is influenced from phone phreaking, a common technique to unlawfully explore telephone systems. The attacker is successful when he makes a victim to trust the fake page and gains his/her credentials related to that mimicked legitimate site.

Anti-Phishing Working Group (APWG) is a non-profit organization which examines phishing attacks reported by its member companies such as iThreat Cyber Group, Internet Identity (IID), MarkMonitor, Panda Security and Forcepoint. It analyzes the attacks and releases the reports periodically (quarterly & half-yearly). It also provides

statistical information of malicious domains and phishing attacks taking place in the world.

Phishing is on the rise in recent years and is very hard to defend against as it explores the weakness of human mind. According to the APWG (2016b) fourth-quarter report, 1,220,523 number of phishing attacks were recorded in 2016 and has been confirmed to be the highest than in any year since it began monitoring in 2004. It also records that there is an increase of 65% of total phishing attacks compared to 2015. The comparison of number of phishing attacks per month in 2015 and 2016 is given in Figure 1.1. It also revealed that last quarter of 2004 had 1609 phishing attacks per month whereas 92,564 phishing attacks per month is seen in last quarter of 2016 which shows an increase of 5,753% of attacks in over 12 years. According to various APWG threat reports[1], the number of phishing attacks encountered each year from 2010 to 2018 are shown in Figure 1.2. It is observed that phishing attacks have increased from 2010 to 2012 and then we see a slight decrease in the number of attacks in 2013 and 2014. It is also important to note that though the number of phishing attacks in 2017 and 2018 are lesser than that of 2016, the count is notable since 2016 saw the highest number of phishing attacks over the period of 12 years. RSA (2013) online fraud report estimates a loss of over USD $5.9 billion with 450000 phishing attacks. Another report from Kaspersky Lab revealed that their anti-phishing system was triggered 154,957,897 times on the systems of Kaspersky lab users in 2016. Interestingly, Kaspersky reported that in 2015 there were 148,395,446 instances triggered the anti-phishing system. Note that over this period of one year there was an increase of 6,562,451 more instances than in 2015. These figures reveal that phishing attacks increased year after year from 2010 to 2018 accounting to billions of dollars in loss.

Attackers design phishing sites in various ways and we classified them into different categories such as

- C1: Phishing sites with manipulation of content, including removal of brand names in the title, copyright, metadata etc. and replacement of legitimate hy-

[1] `https://www.antiphishing.org/trendsreports/`

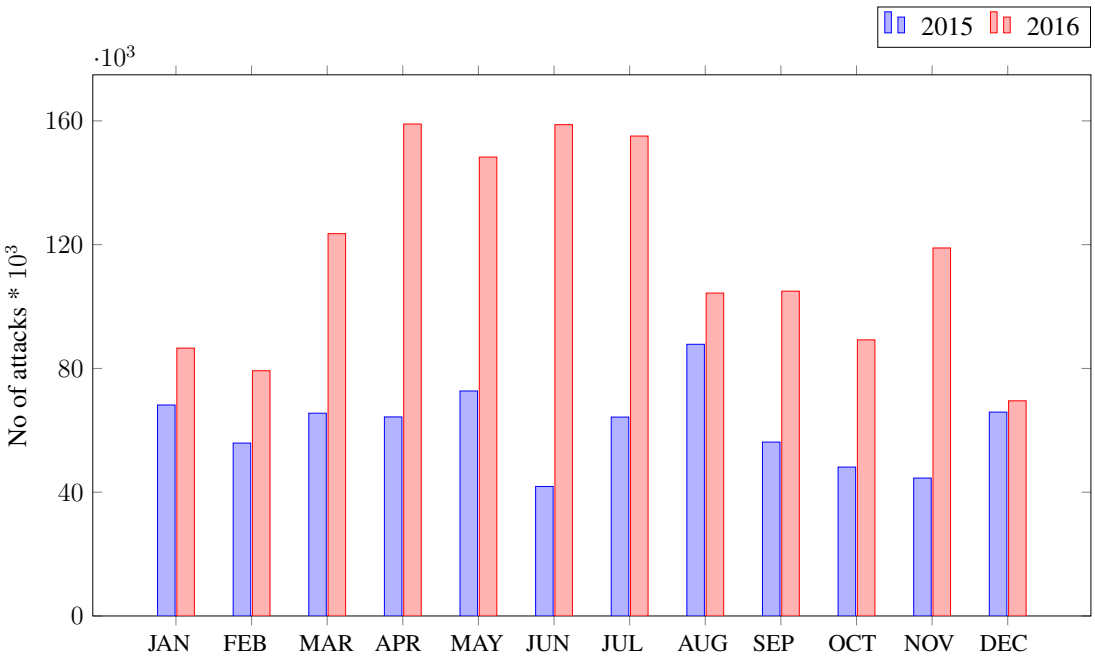


Figure 1.1: Comparison of Phishing attacks during 2015 vs 2016

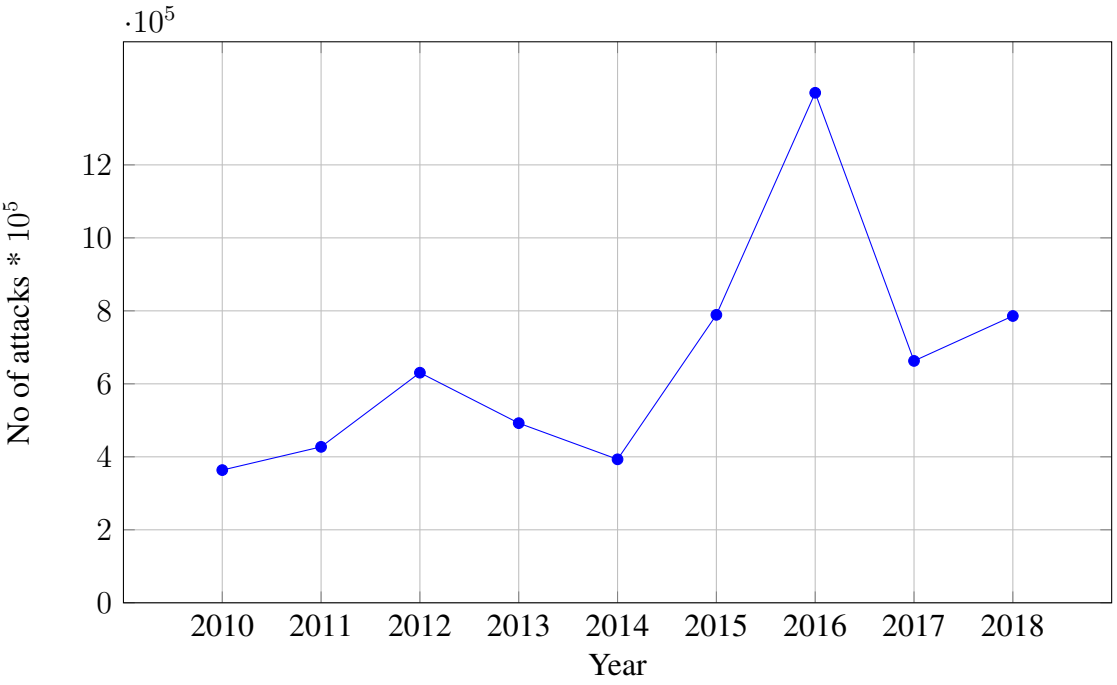


Figure 1.2: Phishing attacks from 2010 to 2018

perlinks with NULL or his own local link.

- C2: Phishing sites replacing the entire legitimate text with a single image.
- C3: Phishing sites using URL obfuscation techniques (misspellings in brand-

names or saving the brandnames in either subdomain or pathdomain using an excess number of dots or lengthy URL).

- C4: Phishing sites with HTTPS connection to imitate high legitimate behavior.
- C5: Phishing sites hosted on free or compromised web servers.
- C6: Phishing sites with legitimate content embedded in iframes or flash.

Note that the current phishing sites may belong to any one of the above categories, or may belong to more than one category. For example, there can be a phishing site which might be of C2 category with HTTPS protocol (C4) and hosted on the compromised web server (C5).

1.1.1 What is zero-day phishing attack

Zero-day Phishing attack is an attack which is not blacklisted and not trained on the old sample data (Almomani et al. 2012). These attacks are also termed as zero-hour or day zero attack. Phishing filters in most of the browsers use a blacklist database consisting of known phishing sites. As zero-day phishing sites are unknown and not listed in the blacklist database, they cannot be identified by the browser phishing filters or toolbars.

An easy method to check whether the toolbar or browser phishing filter can identify new phishing site is to visit `www.phishtank.com` and click on latest phishing sites links. We find most of the phishing sites are displayed in the browser (no blocking by the browser filters), thus toolbars and filters fail to identify new unknown zero-day phishing attacks.

1.1.2 A Phishing example

This example gives some clarity to the concept of phishing. Figure 1.3 claims to be like a legitimate site “ebay”. The design and content is mimicked exactly like ebay website. The user cannot identify the website as fake unless he observes the URL of the website in this case. Figure 1.4 is a snapshot of legitimate “ebay” site. By observing both the website’s snapshot one cannot identify which is phishing and legitimate unless the user is well trained. There are some circumstances even a well-trained individual may not

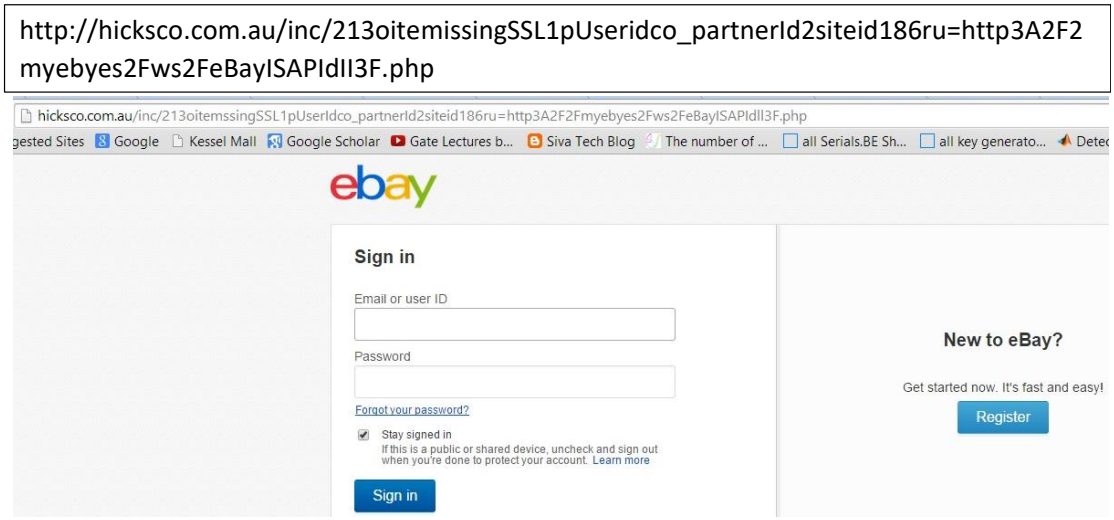


Figure 1.3: Phishing Website targeting legitimate EBAY

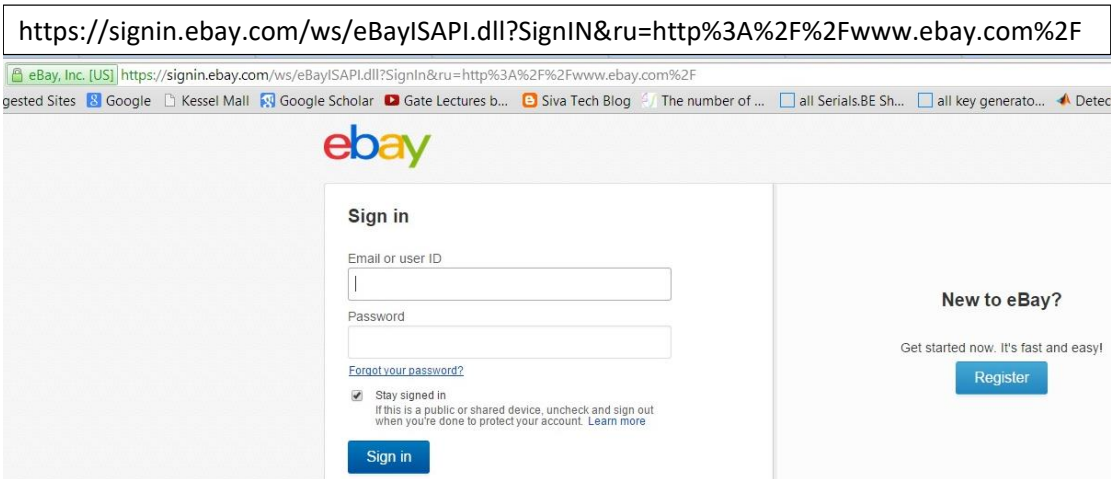


Figure 1.4: Legitimate EBAY Website

identify phishing sites.

1.2 WHY USERS FALL FOR PHISHING

According to Dhamija et al. (2006), online users fall for phishing due to various factors such as,

(1) Inadequate knowledge of computer systems.

(2) Inadequate knowledge on security and security indicators. (In the current scenario, even the indicators are being spoofed by the phishers.)

(3) Inadequate attention to warnings and proceeds further, undermining the strength of

existing tools. (abnormal behavior of toolbars)

(4) Inadequate attention to visual deceptive text in URL and Website content. (For example, paypal text is replaced as paypa1 in URL of address bar, copyright or title of phishing website.)

1.3 TYPES OF PHISHING

Phishing attacks take place through various forms such as email, websites and malware. To perform *email phishing*, attackers design fake emails which claims to be arriving from a trusted source. They send fake emails to millions of online users assuming that at least thousands of legitimate users would fall for it. The content in the email creates an urgency situation to sign into the website provided as a link in the mail. For example, Phishers send emails consisting of eye catching statements like "grand lottery" or some kind of urgency statements such as "Your Account will be suspended if not updated immediately". Once the user clicks on the link, browser is redirected to fake page which exactly looks like a mirror image of legitimate website. The efficiency of fakeness of a website depends on the care taken by the attacker. Some of the attackers are able to manage phishing websites along with security indicators such as green padlock, HTTPS connection etc. We can see the work of an attacker in Figure 1.5.

In *website phishing*, attacker builds a website which looks like a replica of legitimate site and draws the online user to the website either through advertisements in other websites or social networks such as Facebook, Twitter and some blogs etc. In *Malware phishing*, attacker inserts a malicious software such as Trojan horse into a compromised legitimate site without the knowledge of a victim. The malware can be attached to a link, music file or an application in a website and on clicking the links, malicious software is installed into the user computer, keeps track of sensitive information and sends to the attacker. According to APWG (2016a) report, 20 million new malware samples are captured in first quarter of 2016. The vast majority of the late malwares are multifunctional i.e. they steal the information, make the victim's system as a part of botnet or download and install different malicious software without client's notification (KasperskyLab 2014).

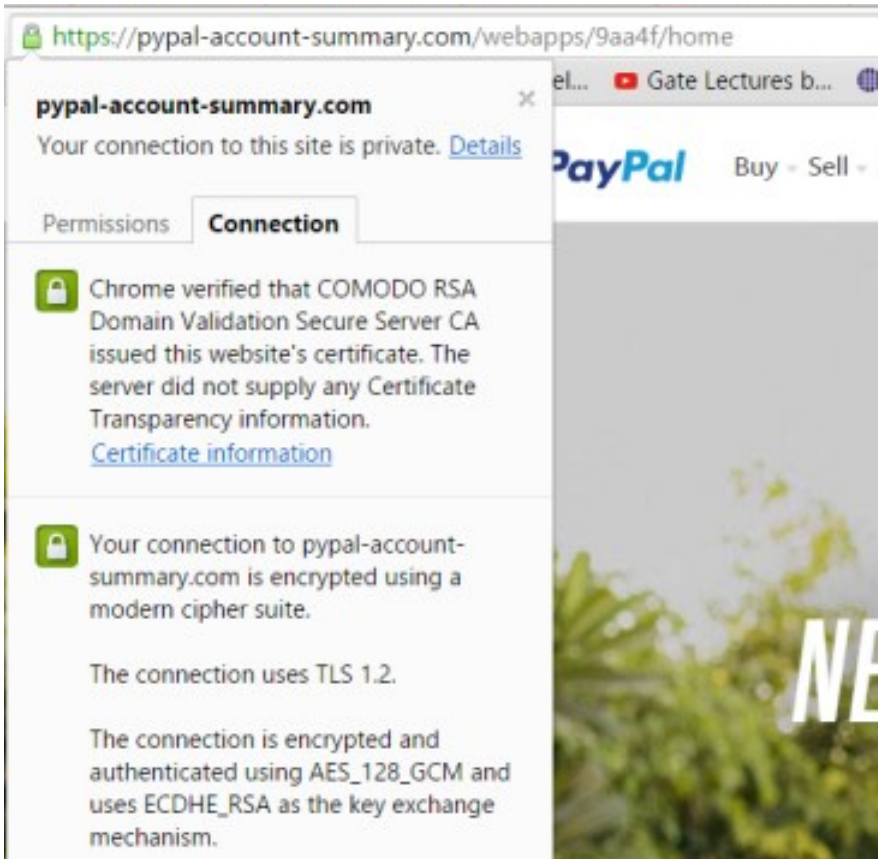


Figure 1.5: Phishing website of PayPal with HTTPS protocol

Sometimes attackers may not target entire society but limit the attack to a specific group of people or community belonging to an organization or a company. They send emails which pretend to be sent by a colleague, manager or a higher official of the company requesting sensitive data related to the company. This kind of phishing is called *Spear Phishing* (Hong 2012). The main intention of general phishing is financial fraud whereas spear phishing is collection of sensitive information. *Whaling* (Hong 2012) is a type of spear phishing where attackers target bigger fish like executive officers or high profile targets of private business, government agencies or other organizations.

Some of the existing techniques (Huh and Kim 2011, Zhang et al. 2007, Tan et al. 2016, Xiang et al. 2011) use third party based features such as Search Engine, WHOIS or Page Ranking features to detect phishing attacks. These techniques fail, when attackers use compromised domains for hosting their phishing pages. Figure 1.6 shows original website of a compromised domain and a phishing page which is hosted in the

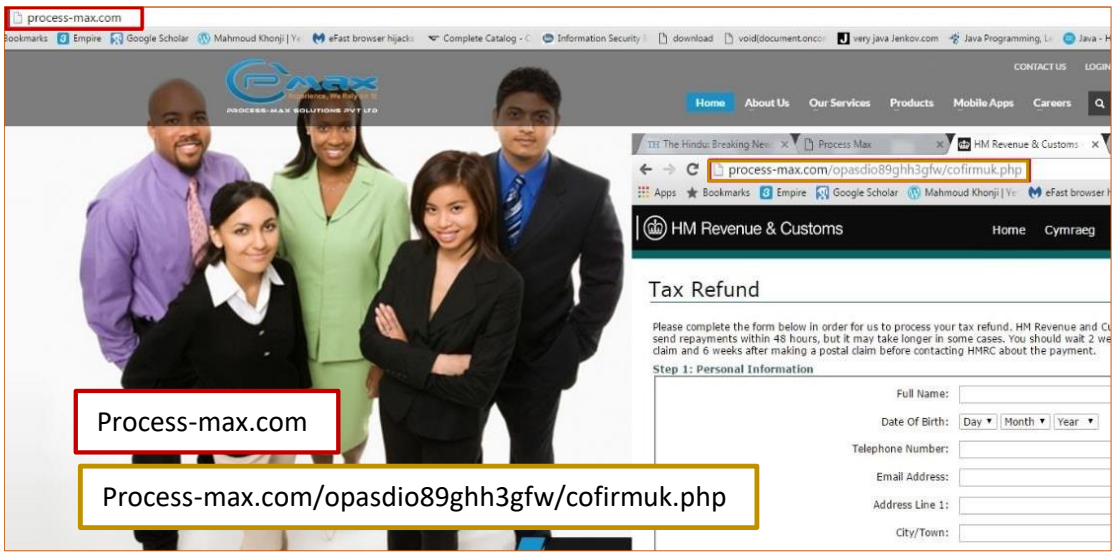


Figure 1.6: Screenshot of Compromised domain and a Phishing page

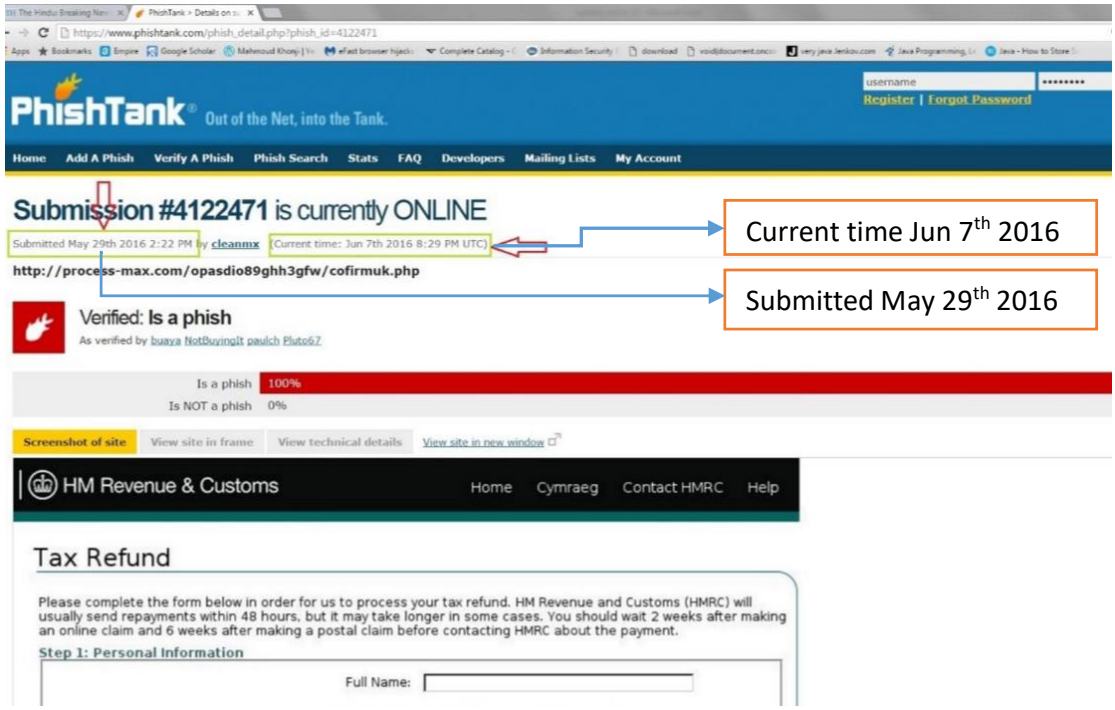


Figure 1.7: Lifespan of phishing site in PhishTank

compromised domain targeting HM Revenue and Customs website. As per the statistics of APWG report 2H2014 (APWG 2014), the life span of phishing sites have a median of less than 10 hours. This report specifies that half of the phishing sites went offline in less than a day, but most of the phishing pages using compromised domains might

be alive for more than a day in the Internet. From the Figure 1.7, it is clear that the above mentioned phishing site which is hosted on compromised domain is still alive after 9 days.The existing techniques are not able to prevent the phishing attacks as evident from Figure 1.2. It may be observed that the number of attacks has increased from 393K to 1,397K during the period 2014 to 2016. Hence, there is a need for designing better mechanisms for detecting phishing websites hosted on compromised domains along with maliciously registered phishing sites.

1.4 MOTIVATION

Attackers choose phishing because it is lucrative and is easy to perform due to the existence of phishing tool kits. According to RSA (2013) report, there has been a loss of $5.9 billion in 2013 itself. Phishing is a growing threat and is hard to defend against it. It has become highly prevalent problem because distributing millions of fake emails is a trivial task and even a less success rate is significantly profitable to the attackers. According to Jagatic et al. (2007), phishing is more effective with social websites such as Twitter, Facebook, Linkedin and Google+. Even though lack of awareness is main reason for growing phishing attacks but expecting users to understand phishing attacks and browse accordingly is unrealistic. Because, online user needs continuous education as phishers try to identify vulnerabilities in existing techniques and design new phishing mechanisms such that phishing warnings are avoided. There exist many techniques to detect phishing sites discussed in Chapter 2. Most of the approaches rely on third party services such as Google search engine, WHOIS, Page ranks and other api services. The third-party based techniques fail when the phishing sites hosted on compromised servers or newly registered legitimate sites are encountered. The list based techniques fails to detect phishing sites which are out of list but has content similar to that of blacklisted site. The visual similarity based techniques compare visual elements (screenshots, styles, logos, favicons) of suspicious site with whitelisted database of resources for the phishing detection. But these techniques are not well adaptable at the client-side due to their computation and space complexity. Thus there is a need for lightweight visual similarity-based technique detecting phishing sites targeting non-whitelisted legitimate resources. The literature also did not address about image based phishing, phishing

sites hosted on compromised servers, phishing sites containing embedded objects such as HTML, Forms, flash and javascript functions.

In this thesis, we aim to address the above limitations with an efficient model resulting in high accuracy and low latency at end user.

1.5 PROBLEM DESCRIPTION

Phishing is a manipulation attack which involves acquiring sensitive information from online users by masking oneself as a legitimate entity. It is a growing threat because it exploits human vulnerability but not system vulnerability. To counter the human vulnerability, online users need to be educated and trained to distinguish between legitimate and phishing website. But, expecting users to analyze the website and browse accordingly is so unrealistic because the accuracy depends on level of understanding and knowledge of the user. Moreover, attackers always come up with new ways of phishing by finding vulnerabilities in existing system which makes the user training and anti-phishing solutions ineffective. The training and educating user is the main solution to decrease the impact but alone itself is not sufficient. Hence we need a better mechanism which is efficient and highly accurate in detecting the phishing website.

1.6 OBJECTIVES

- Propose an ensemble machine learning framework to detect phishing websites (including zero-day phishing attacks).
- Propose a novel method which is independent of third party services for detection of phishing sites.
- Propose a technique which detects phishing sites hosted on compromised servers (PSHCS).
- Propose a visual similarity method to identify phishing websites with less storage cost and high speed.
- Propose a lightweight technique to detect phishing sites in mobile devices.

1.7 THESIS CONTRIBUTIONS

To alleviate and improve the performance of the existing techniques, we present a set of techniques in this research work targeting various kinds of phishing sites.

1. To detect the phishing sites more accurately and at the same time for the reduction of misclassification rate of legitimate sites, we presented an efficient feature-based machine learning framework (Rao and Pais 2019a) with features extracted from three sources such as source code of the website, URL and third-party services.

2. To detect the phishing sites hosted on compromised server, we presented two techniques that discovers inconsistency between the visited page and home page or matched results page. The first technique uses similarity scores between home-page and login page for the detection of phishing sites hosted on compromised server (PSHCS) and heuristic features extracted from URL and hyperlinks for the detection of malicious registered phishing sites. The second technique (Rao and Pais 2019b) uses search engine service with similarity computation module for detecting PSHCS and dynamic search query for detecting non-popular legitimate domains.

3. For the early detection of phishing URLs, we have presented two URL based techniques such that phishing sites are detected by only inspecting the URL without even visiting the URL. Out of which, the first technique (Rao et al. 2020) is deployed as a web application with Random Forest as a classifier including hand-crafted and Term Frequency - Inverse Document Frequency (TF-IDF) features as feature set. For the second technique (Rao et al. 2019), we used multi-model ensemble of Long Short Term Memory (LSTM) and Support Vector Machine (SVM) for the classification of phishing sites. This technique is deployed as an android application for the detection of phishing sites in mobile devices.

4. Finally, to reduce the computation and space complexity in the visual similarity based techniques, we presented a lightweight technique Rao and Pais (2019c)

which uses blacklist approach in the first level filtering and heuristic approach in the second level filtering. The enhanced blacklist mechanism is used to identify the phishing sites which are either replicas or near duplicate sites. The heuristic filtering including extraction of URL and source code based features are used for the detection of non-blacklisted phishing sites. This technique also provides target legitimate site of the blacklisted phishing site.

1.8 THESIS ORGANIZATION

The rest of the thesis is organized as follows: The survey on various existing techniques is discussed in Chapter 2. Chapter 3 discusses the machine learning framework with rich and comprehensive set of features for the detection of phishing sites. Chapter 4 presents the techniques for the detection of phishing sites hosted on compromised domains. Chapter 5 discusses lightweight techniques for the detection of phishing sites by only inspecting the URLs. Chapter 6 discusses lightweight visual similarity approach for the phishing detection. Finally, the summary of all the presented techniques and the future research directions are given in Chapter 7.

CHAPTER 2

LITERATURE REVIEW

There exist many works in detecting phishing emails and websites with different approaches. In this chapter, we survey different anti-phishing techniques designed for website phishing for both mobile and desktop devices. The chapter also provides various phishing indicators for each of the phishing category along with their limitations.

2.1 TYPES OF ANTI-PHISHING TECHNIQUES

Currently, most of the browsers, antivirus softwares and existing anti-phishing techniques are trying to protect the online users from phishing attacks through various ways such as use of whitelists, blacklists, source code, URLs, images, logos, document object models, search engine results, page ranking, and WHOIS data. We divide the anti-phishing techniques into various categories based on the mechanisms used by these techniques and discuss each category with their limitations. The classification of website phishing defense mechanisms is given in Figure 2.1.

2.1.1 List based techniques

Most of the modern browsers such as Chrome, Firefox and Internet Explorer etc. follow list based technique to block phishing sites. There are two types of list based techniques such as whitelist and blacklist. The whitelist contains a list of legitimate URLs which can be accessed by the browsers. This technique allows, website to be downloaded only if it is present in the whitelist. Some of the techniques using whitelist are explained below:

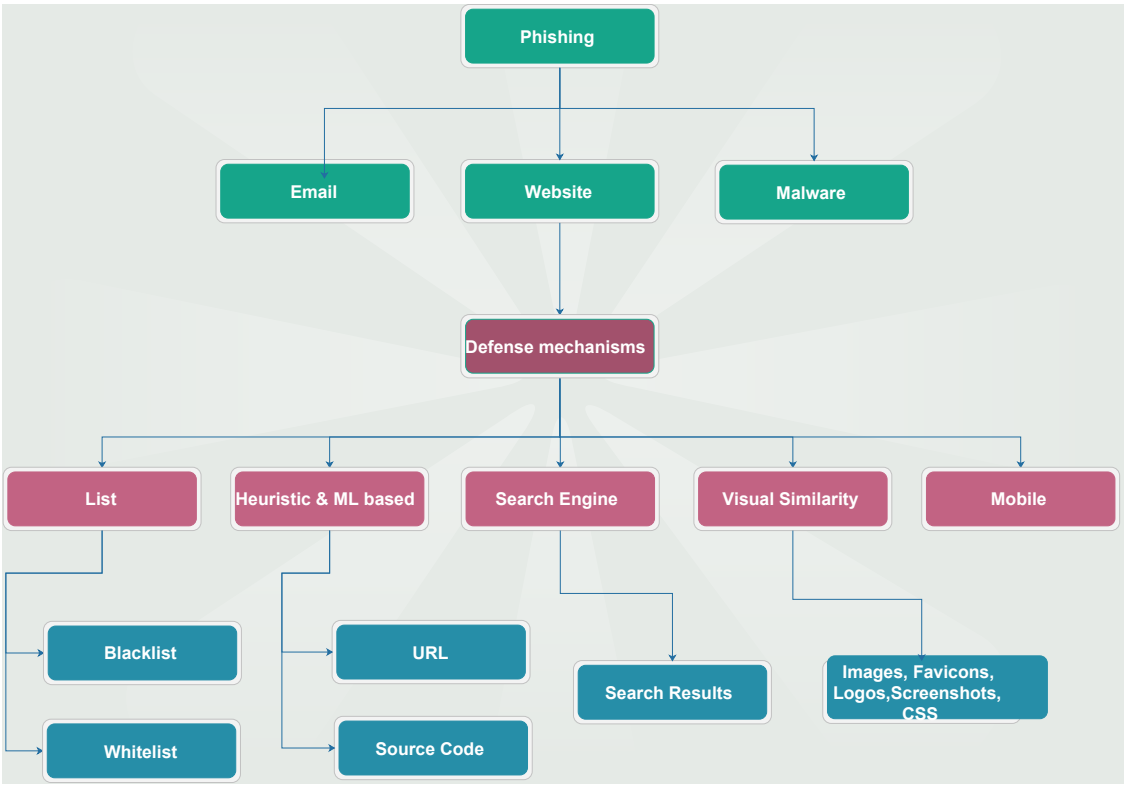


Figure 2.1: Classification of website phishing defense mechanisms

Cao et al. (2008) proposed a technique which maintains all familiar login user interfaces (LUI) of online user which is further used for detection of phishing websites. The Login user interface information includes the set of IPs mapped to the URL, InputArea, CertHash, ValueHash and URL. Authors named their proposed work as Automated Individual White-List (AIWL) which alerts the user when LUI of visited website is not in the whitelisted LUIs.

Dong et al. (2010) proposed a technique to detect phishing sites by analysing the user behavior. The technique maintains a user profile consisting of user data relationship and a user-specific whitelist. The users data corresponding to the website is collected and is used by the detection engine to update the user profile and detect the phishing sites.

Sengar and Kumar (2010) proposed a whitelist-based technique named as PageSafe to prevent the online users from accessing the phishing sites. The whitelist consists of a list of URLs with their IP address mapping. The non-whitelisted URLs undergo

pharming detection to check page anamolies.

Similarly there exists another list based technique called Blacklist which contains phishing or malicious URLs that are blocked by browsers in downloading webpage. Some of the blacklist based techniques are given below.

Phishnet (Prakash et al. 2010) is blacklist technique which predicts the variants of phishing URLs by applying heuristics of combinations of known phishing URLs. The heuristics include interchange of top level domain (TLD), directory structure, IP address and query etc. A unique Phishing URL with randomized string in path fails in detection. With this technique, original legitimate pages in compromised domain are also classified as phishing pages.

Ma et al. (2009) proposed an anti-phishing technique which uses blacklist-based features in addition to other features such as presence of IP address, path tokens, WHOIS etc. The technique checks the presence of URLs in six blacklists (SORBS, URIBL, SURBL, Spamhaus) and each of these are used as features in the detection process. These features are further fed to the machine learning classifiers such as Naive Bayes (NB), Support Vector Machine (SVM) and Logistic Regression (LR) for predicting the legitimacy of the URLs.

Zhang et al. (2008) proposed a highly predictive blacklisting (HPB) system which uses link analysis algorithm to estimate the likelihood of a URL to be malicious. The authors balances the globally compiled blacklists and locally compiled lists with the relevance score and severity rankings for the individual contributors of the blacklist. Further information on list based techniques can be obtained from Aleroud and Zhou (2017); Jain and Gupta (2017); Khonji et al. (2013); Mohammad et al. (2015); Varshney et al. (2016c)

Limitations- The limitation of whitelist techniques is that they block the legitimate websites which are not whitelisted resulting in high false positive rate (legitimate classified as phishing). Similarly, due to the behavior of blocking only blacklisted URLs, the non-blacklisted phishing sites are downloaded by the browser resulting in high false negative rate (phishing classified as legitimate). These non-blacklisted phishing sites

are also called as *Zero-day phishing sites* (Almomani et al. 2012; Khonji et al. 2013). Another limitation is that, a small change in the blacklist URL is sufficient to bypass the list based technique. Finally, frequent update of these lists are mandatory to counter the new phishing sites in addition to the extra cost of remote querying and matching the visited URL.

2.1.2 Heuristic and Machine Learning based techniques

These techniques extract the features from different sources like URL, digital certificates, anchorlinks, copyright, textual content, website traffic and WHOIS etc. In this thesis, we define heuristic features as the most common characteristics that are found to exist in phishing sites in reality and the techniques that use such features are termed as heuristic techniques. Some heuristic techniques also use machine learning algorithms to analyze hidden patterns from the features for the classification of phishing sites. We divide these heuristic techniques into two categories

A. URL based techniques : These techniques extract features from URL to detect the phishing URLs. Such techniques use count-based features, binary features, blacklisted words to check if a suspicious URL is legitimate or phishing.

Techniques such as Choi et al. (2011); Gastellier-Prevost et al. (2011); Gowtham and Krishnamurthi (2014); He et al. (2011); Huang et al. (2012); Marchal et al. (2016); Moghimi and Varjani (2016); Mohammad et al. (2012); Sahingoz et al. (2019); Su et al. (2013); Thomas et al. (2011); Wang and Shirley (2015); Weiss and Khoshgoftaar (2017); Zhang et al. (2017) use count-based features such as number of characters, hyphens, redirections, numbers and digits. Similarly some of the techniques (Canali et al. 2011; Chu et al. 2013; Gastellier-Prevost et al. 2011; Moghimi and Varjani 2016; Mohammad et al. 2012; Ranganayakulu and Chellappan 2013; Shirazi et al. 2018; Xu et al. 2013) use binary features like presence of IP address, special characters (*, -, /, ?), blacklisted words, brand names, and https protocol. Also, there exists techniques (Canali et al. 2011; Verma and Dyer 2015) which use frequency distribution of characters in the URL as a parameter to detect the legitimacy of the URL.

B. Source Code based techniques : These techniques extract features from source

code of the suspicious URLs to detect phishing sites. Features such as hyperlinks-based features, text-based features, and network based features are used by various approaches. Some of the techniques (Chou et al. 2004; Joshi et al. 2008) also detect phishing sites by evaluating the post data. These techniques considers response of the submitted page as a parameter for the classification. The hyperlink-based features (Chou et al. 2004; Cook et al. 2008; Marchal et al. 2017; Moghimi and Varjani 2016; Rao and Pais 2019a; Shirazi et al. 2018; Srinivasa Rao and Pais 2017) include ratio of foreign links, broken links, common URLs and null links. Text-based feature (Marchal et al. 2016; Xiang et al. 2011) comprise extraction of prominent keywords from the web page, copyright, title, description and TF-IDF is one such method to extract prominent keywords. These features are fed to the machine learning algorithms for the phishing classification.

Now a days, most of the researchers are concentrating on use of machine learning algorithms (ML) and deep learning algorithms (DL) applied on the features extracted from the websites to detect phishing attacks. As ML or DL based techniques are based on heuristic features so they are able to identify the zero-day phishing attacks which makes them advantageous when compared to list based techniques. The machine learning algorithms used in the literature (He et al. 2011; Pan and Ding 2006; Zhang et al. 2014) for the classification include Sequential Minimum Optimization (SMO), J48 tree, Random Forest (RF), Logistic Regression (LR), Multilayer Perceptron (MLP), Bayesian Network (BN), Support Vector Machine (SVM) and AdaBoostM1 etc. Similarly, the deep learning algorithms such as Deep Neural Network (DNN), Recurrent Neural Network (RNN), Long Short Term Memory (LSTM), Deep Belief Network (DBN) and Convolution Neural Network (CNN) are used for the phishing classification (Bahnsen et al. 2017; Li et al. 2019; Sahingoz et al. 2019; Yi et al. 2018; Zhao et al. 2018). These techniques work efficiently on the large sets of data. According to Whittaker et al. (2010) it is possible to achieve more than 99% true positive rate and less than 1% false positive rate. According to Khonji et al. (2013), ML based classifiers achieved an accuracy of above 99% . This made us to concentrate on using machine learning algorithms as classifiers. The other side of ML based techniques is that its performance

depends on the size of training data, feature set and type of the classifiers. Hence, in this section, we discuss some of the heuristic techniques that extract features from URL and source code for the classification of phishing sites using machine learning algorithms.

Pan and Ding (2006) proposed an anti-phishing approach based on the extraction of website identity from DOM objects such as title, keyword, request URL, anchor URL and main body etc. When a phishing site fakes a legitimate site, it claims false identity demonstrating abnormal behavior compared to the legitimate site. They experimented with ten features using supporting vector machine classifier for the classification of data. This approach fails when the attacker hosts a phishing page on the compromised domain.

CANTINA (Zhang et al. 2007) technique operates on the textual content present in the website. The Term Frequency - Inverse Document Frequency (TF-IDF) algorithm is applied on the textual website content to extract high TF-IDF score words and fed to the search engine to identify a phishing site. It also includes additional heuristic features and simple forward linear model for classification of websites. As the performance of this approach depends on the textual content of the suspicious site, it fails when the textual content is replaced with images. It also has a performance problem with respect to time since it depends on the third-party service like search engine.

Miyamoto et al. (2008) evaluated the performance of machine learning algorithms in classifying the websites as legitimate or phishing website. Authors incorporated all the features of CANTINA model to detect the phishing websites. They employed nine machine learning algorithms for the classification of websites. Among all of them, the AdaBoostM1 algorithm performed the best with a lowest error rate of 14.15%. This technique has the same limitation as of CANTINA approach i.e. it fails when the textual content is replaced with an image.

Xiang et al. (2011) proposed a comprehensive feature based approach which uses machine learning algorithms such as Bayesian networks for classification of phishing websites. This technique is called CANTINA+ which is an extension to CANTINA including eight novel features such as HTML DOM features, third-party services and

search engine based features. It has divided entire datasets into two parts: First, unique testing phish dataset resulted in 92% detection accuracy on feeding to machine learning algorithm. Second, near-duplicate testing phish dataset resulted in 99% detection accuracy on feeding to machine learning algorithm. The authors have used six learning algorithms to train the dataset and calculate the performance of the model. The Bayesian Network algorithm performed the best in comparison with other algorithms. This technique fails to detect the phishing sites when the website textual content is replaced by an image.

He et al. (2011) proposed a heuristic based technique which uses the combination of CANTINA, (Pan and Ding 2006), and PILFER (Fette et al. 2007) methods to detect phishing sites. The authors extracted twelve features from the suspicious website and fed them to the support vector machine for classification of phishing and legitimate websites. As this technique adopted the features either from CANTINA or Pan and Ding (2006), it has the same limitation as of CANTINA technique. It also suffers from high time complexity in loading webpage due to the use of third-party services like search engine, page ranking etc.

Zhang et al. (2014) proposed a model consisting of URL and website content to detect Chinese phishing e-business websites. Authors incorporated fifteen domain specific features for detecting phishing attacks in Chinese e-business websites. They evaluated the model with four different machine learning algorithms for classification of phishing sites. Sequential Minimal Optimization (SMO) algorithm performed best in detecting phishing site with an accuracy of 95.83%. The limitation of this approach is that it targets only one domain of phishing sites i.e. it may not work efficiently with non-Chinese websites.

Gowtham and Krishnamurthi (2014) proposed a heuristic based technique combined with machine learning classifier to detect phishing site. They have used private whitelist and login form identifier filters to reduce the false positive rate and computation cost in the system. SVM algorithm is used as a classifier for classification of phishing websites. The Authors have extracted 15 features from URLs characteristics, URL identity, keyword identity set and domain name analysis to form a feature vector. This feature

vector is fed to SVM to calculate the performance of a model. The proposed model has achieved true positive rate of 99.65% and false positive rate of 0.42%. As this technique depends on the textual content of a website, it might not work effectively when the entire website content is replaced with a single image.

Mohammad et al. (2014b) presented an intelligent model for detecting phishing websites based on self-structuring neural networks. The authors extracted seventeen features from URL and website content which are used as an input to artificial neural networks for classification of websites. This technique has an advantage of adapting its neural network according to the change of phishing characteristics. For better performance, this model has to be retrained frequently with a fresh and up to date training dataset.

Chiew et al. (2015) presented a visual similarity based technique where the logo of a suspected website is extracted using machine learning technique and fed to Google image search to retrieve target identity. The suspected site is termed as phishing when actual domain is not equal to domain resulted from Google image search result. This technique depends on success ratio of logo extraction through machine learning. This technique fails when the phishing site with absence of logo is encountered.

Moghimi and Varjani (2016) proposed a rule based method and developed it as a browser extension to detect phishing sites. This method is based on the features extracted from website content and used SVM to classify the phishing websites. As this approach is solely dependent on website content, it fails when attacker redesigns the website with modified visual content such that brand name is misspelled or trimmed, changing all links to its local domain. This technique only targets phishing sites imitating legitimate bank websites.

Whittaker et al. (2010) proposed a blacklist technique which extracts common features of known phishing websites, and uses these features for the detection of phishing sites. They used RF classifier for the detection of phishing sites. The authors claim that RF performs the best with noisy live phishing data and they are able to achieve a false positive rate lower than 0.1%. As this technique depends on the blacklist of phish-

ing data, it fails to detect phishing sites which are not blacklisted. Further information on heuristic and machine learning based techniques can be obtained from Aleroud and Zhou (2017); Jain and Gupta (2017); Khonji et al. (2013); Mohammad et al. (2015)

Limitations- Firstly, some of the phishing sites may not have the common features resulting in poor detection rate compared to list based mechanism. Secondly, an attacker can bypass the heuristic features once he finds out the algorithm or the features used in the detection process, thereby reaches his goal of stealing sensitive information.

2.1.3 Search engine based techniques

These techniques use search engine results as a base to detect the legitimacy of the websites. These techniques extract the keywords, title, copyright, domain from the given website's source code. These elements are used to generate a lexical signature, which is fed to search engine. The presence of Domain URL in the search engine results page (SERP) determines the legitimacy of the website. Some of the latest and popular search engine based techniques are given below.

CANTINA Zhang et al. 2007 uses famous Term Frequency - Inverse Document Frequency (TF-IDF) algorithm to extract the unique keywords describing the website. These words are assembled to generate a lexical signature, fed to the search engine to output search results. If the domain of the given URL is present in the output search results, then it is classified as legitimate else as phishing.

Xiang and Hong (2009) used title, copyright and TF-IDF terms as a search query to detect the legitimacy of the given URL. The authors used Named Entity Recognition module to identify the brand name embedded in the copyright, title and TF-IDF keywords. The brand names are given as search query to the search engine to return the search results. The presence of the suspicious domain in the search results is considered as legitimate else phishing.

Chiew et al. (2015) used logo as the search image and is fed to the Google image search interface to identify the legitimacy. The authors used machine learning algorithm to extract the logo from the given URL. The suspicious domain is checked for the presence in the returned search results, if found, it is classified as legitimate else

phishing.

Varshney et al. (2016b) proposed a lightweight search engine technique which uses domain and title extracted from the given URL for the detection of phishing sites. The authors designed a Chrome extension and claimed that it is lightweight as it requires a single search engine request.

Jain and Gupta (2018) proposed a two-level search engine based technique to detect phishing sites. The authors used search engine mechanism similar to Varshney et al. (2016b). They applied this mechanism at level one to detect phishing sites and hyperlink based features at the next level to detect newly registered and unpopular legitimate domains. The hyperlink features include the ratio of local (L) and null (#) hyperlinks (N) to the total number of hyperlinks (T) in the website. If the ratio of L and T is high and the ratio of N and T is low then the technique classifies the suspicious website as legitimate site. But, the technique fails when the designed phishing site contains all the hyperlinks assigned with a single common local page, leading to L/T as 1. These type of hyperlinks are also termed as common hyperlinks (Rao and Pais 2019a).

Ramesh et al. (2014) proposed a method based on the direct and indirect associations with the domain of the suspicious URL. This method uses TF-IDF for the indirect associations and anchor links for the direct associations. The intersection of both these sets determines the status of the website.

Dunlop et al. (2010) proposed a technique which feeds the snapshot of the given website to OCR software for converting the image to text. This technique uses the text as a search query to the search engine and then checks whether the returned results include the domain of the suspicious URL. If the domain is present then the URL is classified as legitimate else it is classified as phishing.

Tan et al. (2016) used weighted URL tokens system and N-gram model for the extraction of identity keywords from plaintext and URL. These keywords are fed to search engine and the returned results are checked for the presence of domain of the suspicious URL. If the domain is present then suspicious URL is classified as legitimate else it is classified as phishing.

Chang et al. (2013) proposed a method which segments a logo from given suspicious website and is fed to the Google image search to determine the identity of the website. The returned keyword from the image search is fed to Google text search to get the search results. If the domain of query-website is matched with any domain of the search results then it is classified as legitimate else it is classified as phishing. The authors extracted the logo by segmenting `1 x 2, 2 x 2 or 3 x 3` size segments from top left part of the webpage. But, this assumption might not be true for all the cases and moreover it does not result in best fit logo extraction due to which the search engine might return unwanted results. Further information on search engine based techniques can be obtained from Aleroud and Zhou (2017); Jain and Gupta (2017); Khonji et al. (2013); Mohammad et al. (2015)

Limitations- The limitation of search engine based techniques is that they fail to detect phishing sites which are hosted on compromised servers. In this case, the phishing pages are termed as legitimate sites leading to high false negative rate. On the other side, the newly registered legitimate sites are classified as phishing sites due to its absence in search results leading to high false positive rate. Finally, the techniques which use search query as identity keywords extracted from copyright, plaintext, header texts fail in extracting relevant terms when the text in phishing site is replaced with an image.

2.1.4 Visual Similarity based techniques

Since the phishing sites have visual appearance similar to that of target legitimate site, it is difficult for the users to distinguish between fake and real website with naked eye. Hence, the visual similarity based techniques are used to differentiate between suspicious site and its target legitimate site. The main objective of the phisher is to deceive the user by making exact image of legitimate site such that the user doesn't get any doubt. Hence, anti-phishing techniques compare suspicious website image with legitimate image database to get similarity ratio which is used for classification of website. The website is classified as phishing when the similarity score is greater than a certain threshold else it is treated as legitimate. Some of the visual similarity based techniques are given as follows.

Rosiello et al. (2007) proposed Document Object Model (DOM) similarity-based approach named as *DOMAntiphish* for the detection of phishing sites. Authors maintained whitelist of DOMs of target website which are used for similarity calculation between the DOM of visited website and stored DOMs. This technique fails when DOM obfuscation techniques like insertion of empty tags or deletion of some non-important tags are applied to the designed phishing site.

Hara et al. (2009) proposed a visual similarity based technique which uses whitelist of screenshots of targeted websites for the detection of phishing sites. The screenshot of visited website is compared with whitelisted screenshots for calculating the similarity score between them. If the similarity score is above a threshold and both the URLs are unique then the visited site is classified as phishing else legitimate. The limitation of this approach is that it has very high false positive rate of 17.5%.

PhishZoo (Afroz and Greenstadt 2011) is an anti-phishing technique which uses trusted profiles for the detection of phishing websites. It detects the phishing sites which look like legitimate sites by comparing the visited website content against stored trusted profiles. It fails when the phishing site is significantly different from the target website. The trusted profiles include URL, SSL, images and scripts.

Mao et al. (2017) proposed a technique called Phishing Alarm which uses whitelisted Cascaded Style Sheets (CSS) between the visited page and target pages for determining the phishing status of the website. Authors claimed that CSS features are very hard to be evaded by the attackers because of their property that defines the visual structure to the HTML elements in the webpage. The limitation of this technique is it fails when the entire HTML content is replaced with an image in the website. Also, phishing sites may go undetected when the target website is not whitelisted.

Zhang et al. (2011) proposed a Bayesian approach which detects phishing sites by comparing the visual and textual contents between the target website and visited website. The textual contents include plain text extracted from DOM after removing HTML tags and stemming process. The visual contents include extraction of images from HTML and forwarded to normalization for the generation of visual signatures with

color and coordinate features. Earth Movers Distance is used to calculate the similarity score between the protected website and the visited website. If the score is above a threshold then it is classified as phishing else it is classified as legitimate.

Auntietuna (Ardi and Heidemann 2016) is a technique which stores cryptographic hashes of small chunks of data as whitelisted data for every target website. The small chunks of data is chosen based on the text between $< p >$ and $< div >$ tags. For any visiting website, the number of matchings of hashes between the visited and target website is calculated and if it is greater than a threshold then it is classified as phishing site. As the technique is relying on cryptographic hash, a small change in the chunk of data is sufficient to bypass the technique. Further information on visual similarity based techniques can be obtained from Aleroud and Zhou (2017); Jain and Gupta (2017); Khonji et al. (2013); Mohammad et al. (2015)

Limitations- The limitations of visual similarity techniques are as follows: 1) Image comparison of suspicious website with entire legitimate database store takes more time. 2) More space to store legitimate image database. 3) Web page with animated website compared with phishing website leads to low percentage of similarity that leads to high false negative rate. This technique fails, when the background of web page is slightly changed without deviating from visual appearance of legitimate site.

To summarize the different types of techniques, we provide some of the phishing factor indicators for all phishing categories used in the literature as shown in Table 2.1. Also, the summary of these categories with respect to seven features is given in Table 2.2. These include **CSD** : Compromised Sites Detection; **CSA** : Client-side Adaptability; **ZDD** : Zero-day Detection; **DBD**: Drive-by -Downloads Detection; **LI** : Language Independent; **TPI** : Third-party Independent; **TI** : Target Independent

2.1.5 Mobile based phishing detection

Mobile webpages differ from their desktop versions in terms of content, layout, and functionality. Due to the small screen size of mobile devices, the visibility of URL, page layout, content in the browser is limited whereas the users can have a relatively larger view in desktop browsers. The attackers create a phishing site which looks sim-

Table 2.1: Phishing factor indicators

Category	**Phishing factor indicator**
List	Blacklist and Whitelist of URLs, digital certificates
URL	# of dots, slash, presence of special characters $(@, *, \$, :, \|, -, ',)$, digits, HTTPS, IP address, phish-hinted words either in the base URL or path of the entire URL string, pagerank and age of the domain.
Source Code	null anchorlinks ratio, foreign anchorlinks ratio, frequency of domain in CSS, image, scripts, broken links ratio, identity keywords extracted from copyright, plaintext, header texts, title, and other metadata.
Search engine	matched domain in the search results page
Visual Similarity	images, screenshots, DOMs, logos, favicons, and CSS etc are compared with stored trusted resources.

Table 2.2: Comparison of existing anti-phishing techniques

Category	CSD	CSA	ZDD	DBD	LI	TPI	TI
List	✓	✓		✓	✓	✓	✓
URL		✓	✓	✓	✓	✓	✓
Source code		✓	✓			✓	✓
Search engine		✓	✓		✓		✓
Visual similarity	✓		✓		✓	✓	

CSD : Compromised Sites Detection **CSA** : Client-side Adaptability; **ZDD** : Zero-day Detection; **DBD**: Drive-by -Downloads Detection; **LI** : Language Independent; **TPI** : Third-party Independent; **TI** : Target Independent

ilar to that of legitimate site. Thus, it is difficult for any mobile user to distinguish between legitimate and fake website. For instance, websites may be blacklisted in desktop browsers but same websites may not be blacklisted in mobile web browsers. Most of the anti-phishing techniques were designed to detect malicious websites in desktop computers and they may not work for mobile webpages due to the hardware limitations, difference in screen size and layout.

As there exists very limited work in detecting mobile phishing in the literature, we skip the categorization of the techniques but we discuss some of the latest and popular mobile-based anti-phishing techniques in this section.

Han et al. (2007) proposed a technique to detect phishing sites based on the pre-stored Login User Interface (LUI) information in mobile devices. A plug-in at the browser side compares the suspicious site LUI information with the pre-stored LUI information to detect the phishing sites.

An automated anti-phishing scheme MobiFish has been proposed by Wu et al. (2016) for mobile devices. This scheme detects website phishing and malicious applications in mobile devices by comparing the actual identity with the claimed identity obtained using Optical Character Recognition (OCR). The extracted text from OCR is used to calculate the claimed identity whereas actual identity is extracted from the URL. If these identities found to be different, then a warning is alerted to the user.

URL based technique has been proposed by Orunsolu et al. (2017) to detect website phishing in mobile devices. In this technique, website status has been obtained using frequency analysis of phishing features and these URL-based features are fed to SVM for the detection of phishing attacks.

A defense system has been proposed by Hou and Yang (2012) which logs keystrokes and warns the user as he enters sensitive information in a malicious mobile application. In this technique, the authors used a whitelist of trusted applications and their corresponding UserIDs to detect the legitimacy of the application.

Amrutkar et al. (2017) designed a Firefox extension named as kAYO to detect phishing sites in mobile devices. The authors extracted various static features from the con-

tent of HTML, Javascript, URL and mobile-specific capabilities. They used Logistic Regression for the classification of phishing and legitimate sites.

Bottazzi et al. (2015) implemented an android application MP-Shield as a proxy service on top of TCP/IP stack to detect phishing sites. Authors used Virtual Private Network (VPN) service to analyze the IP packets and thereby packets are sent to Watchdog for checking the presence of URL in the blacklist. For the non-blacklisted URLs, a set of attributes such as URL length, digits count and the number of subdomains are extracted and fed to various machine learning algorithms for the classification.

Chorghe and Shekokar (2016) proposed a technique to detect phishing sites in android mobile devices. It includes extraction of URL from the browser followed by static analysis of URL, extraction of HTML, lexical and third-party based features. Finally, the extracted features are fed to SVM for the classification of URL.

Ndibwile et al. (2017) used deceptive login simulation for the detection of mobile phishing pages. They implemented the technique as an android application named as UnPhishMe which feeds fake credentials to the login form of the given suspicious website. The authentication of the login process with fake credentials is used to detect the legitimacy of the website. The authentication is achieved based on the matching of hash codes of suspicious URL before and after the login process.

The summary of existing mobile-based anti-phishing techniques with respect to attributes such as Language independent (LI), Third-party independent (TPI), Drive-by-download independent (DBDI), Target independent (TI) along with their limitations are given in Table 2.3 and overall summary of phishing defense mechanisms are given in Table 2.4.

2.2 EVALUATION METRICS

Since the subsequent chapters include various experimental results and comparison with existing works, we would like to introduce several traditional metrics to identify the performance of our system. We considered condition positive as phishing (P) and negative as legitimate (L). Correctly predicted phishing sites is termed as hit or true positive (TP), correctly predicted legitimate sites as true negative (TN), incorrectly predicted le-

Table 2.3: Comparison of existing mobile based anti-phishing techniques

Technique	Description	LI	TPI	DBDI	TI	Limitations
Han et al. (2007)	Proposed a technique to detect phishing sites in mobile devices based on pre-stored Login User Interface (LUI) information.	Yes	Yes	No	No	Detection fails when the phishing site falls out of LUI list.
Hou and Yang (2012)	A defense mechanism that logs the keystrokes and warns the user if sensitive information is entered in malicious mobile application.	Yes	Yes	No	No	The technique fails if a new application is installed by the user.
Bottazzi et al. (2015)	MP-Shield, an android application is developed which inspects the IP packets for URLs. These URLs are checked in blacklist and heuristic features are extracted from non-blacklisted URLs.	Yes	No	No	Yes	Fails for image-based phishing sites
Wu et al. (2016)	MobiFish detects website phishing and malicious applications by comparing actual identity with claimed identity obtained from Optical Character Recognition (OCR).	Yes	Yes	No	No	Detection rate rely on capability of OCR
Chorghe and Shekokar (2016)	A system to detect phishing in mobile devices by extraction of URL and source code based features	Yes	No	No	Yes	Dependent on third-party services
Orunsolu et al. (2017)	A technique to classify the URL based on frequency analysis of phishing features from URL.	No	Yes	Yes	Yes	Language dependent i.e, cannot handle non-English webpages
Amrutkar et al. (2017)	kAYO, a firefox extension to detect phishing sites in mobile devices based on static features extracted from the suspicious URL.	Yes	Yes	No	Yes	Less True Positive Rate (TPR) of 89%
Ndibwile et al. (2017)	Detects the legitimacy of the website based on the login form authentication with fake credentials	Yes	Yes	No	Yes	Fails to detect non-login phishing pages

LI : Language Independent; **TPI** : Third-party Independent; **DBDI** : Drive-by-Download Independent; **TI** : Target Independent

gitimate sites as false positive (FP), incorrectly predicted phishing sites as false negative (FN). For a better system, false positive should be minimum and true positive should be maximum.

- *True Positive Rate (TPR)* : % of correctly predicted phishing sites (TP) out of

Table 2.4: Summary of anti-phishing techniques

Type	Techniques	Description	Limitations
Blacklists (or Whitelists)	BList- Ma et al. (2009); Prakash et al. (2010) and WList-Cao et al. (2008); Dong et al. (2010)	Blacklists are frequently updated lists of previously detected phishing URLs, Internet Protocol (IP) addresses or keywords.Whitelists, on the other hand, are the opposite, they are the list of legitimate URLs. **Advantages:** High accuracy rate	Fails to detect zero day phishing sites
Heuristics and Machine Learning	Marchal et al. (2016); Moghimi and Varjani (2016); Sahingoz et al. (2019)	These depend on the most common characteristics that are found to exist in phishing attacks in reality. But, these characteritics are not guaranteed to always exist in all phishing site and may even present in legitimate sites too **Advantages:** Detects zero day phishing sites	Results in high false positive rate when phishing characteristics present in legitimate sites
Search Engine	Chiew et al. (2015); Varshney et al. (2016b); Xiang and Hong (2009)	These techniques use search engine results as a base to detect the legitimacy of the websites. They extract the keywords, title, copyright, domain from the source code for the generation of search query. **Advantages:** Low false positive rate	Fails to detect phishing sites hosted on compromised servers and also fails to detect newly or unpopular legitimate sites. sites
Visual Similarity	Hara et al. (2009); Mao et al. (2017); Rosiello et al. (2007)	This technique compares the visual resemblance features(pixels, images, screenshots, page layouts, and logos etc) with pre-stored database of legitimate content. **Advantages:** Can detect zero day phishing sites also	Very low accuracy rate and takes more time and space to store legitimate resources
Mobile	Amrutkar et al. (2017); Orunsolu et al. (2017); Wu et al. (2016)	These techniques detect phishing sites in mobile devices with limited resources such as RAM, low computational power,screen size, limited page content. **Advantages:** Fast detection	Techniques designed to detect malicious websites in desktop computers may not work for mobile webpages due to the hardware limitations.

total number of phishing sites. It is also termed as Sensitivity or Recall.

$$TPR = \frac{TP}{TP + FN} * 100 \tag{2.1}$$

- *True Negative Rate (TNR)* : % of correctly predicted legitimate sites (TN) out of total number of legitimate sites. It is also termed as Specificity.

$$TNR = \frac{TN}{TN + FP} * 100 \tag{2.2}$$

- *False Positive Rate (FPR)* : % of incorrectly predicted legitimate sites (FP) out of total number of legitimate sites.

$$FPR = \frac{FP}{FP + TN} * 100 = 1 - Specificity \tag{2.3}$$

- *False Negative Rate (FNR)* : % of incorrectly predicted phishing sites (FN) out of total number of phishing sites.

$$FNR = \frac{FN}{FN + TP} * 100 = 1 - Recall \tag{2.4}$$

- *Accuracy (Acc)* : % of correctly predicted legitimate and phishing sites out of total number of websites.

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} * 100 = \frac{TP + TN}{P + L} * 100 \tag{2.5}$$

- *Error rate (ERR):* % of phishing and legitimate sites that are incorrectly classified out of total number of websites.

$$ERR = \frac{FP + FN}{P + L} * 100 \tag{2.6}$$

- *Precision (Pre):* % of correctly predicted phishing sites out of total number of predicted phishing sites.

$$Pre = \frac{TP}{TP + FP} * 100 \tag{2.7}$$

- *F-Measure:* It is harmonic mean of precision and recall. The F-Measure will always be nearer to the smaller value of Precision or Recall. It is a combined measure that assesses the precision and recall tradeoff. It is also termed as F1 score or F score.

$$F = 2 * \frac{Pre * Recall}{Pre + Recall} \tag{2.8}$$

- *Area Under the ROC curve (AUC):* AUC measures the area underneath the entire Receiver operating characteristic (ROC curve). The closer the AUC of a model

towards 1, the better it is. The ROC curve is the plot between false positive rate and True Positive rate.

- *Matthews Correlation Coefficient (MCC):* It calculates the quality of binary classification by considering true, false positives and negatives. This measure is considered as a balanced measure which is used for the dataset with very different class sizes. The measure provides the correlation coefficient between predicted and observed outcomes in the binary classification.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.9}$$

2.3 RESEARCH GAPS

In the previous section, we have discussed various categories of anti-phishing techniques and also have given brief overview of the techniques. We also provided limitations of each phishing detection category which leads to the following research challenges.

1. **Search Engine based technique:** Majority of the techniques are search engine dependent to detect phishing sites. Because of this dependency on search engine, newly registered domains are classified as phishing sites which results in reduction of true negative rate. Phishers compromise less reputed and secured website for hosting their phishing website. We also observed that the same compromised legitimate site is used for hosting several phishing websites. It indicates that compromised website information is being sold in dark web. These kind of websites are very hard to bring down as most of the anti-phishing classify the website as legitimate. Hence, there is a need for a third party independent or an improved search engine based mechanism for the detection process.

2. **Phishing sites with Embedded objects:** Majority of the techniques are not able to detect Phishing websites which use embedded objects such as HTML, images, scripts, forms and flash. Therefore, detection of phishing site which use embedded object is still an open challenge.

3. **Phishing sites with dynamic content:** Phishing sites which use javascript functions to make the content of a website dynamic such that anti-phishing techniques that depend on static source code of websites are bypassed. Most of the existing techniques extract textual features from the static source code of the website. Therefore, exploring and identifying the live phishing site with such dynamic content are promising research direction.

4. **Ensemble classifier:** Machine Learning (ML) is a buzz word in the technology world right now. It is being used in almost all of the real world applications. Khonji et al. (2013), and Whittaker et al. (2010) have mentioned that ML based techniques achieve an accuracy of more than 99% for phishing detection. Most of the existing techniques use single and weak classifiers for the classification of data but several studies show that combining multiple weak classifiers into one aggregated classifier leads to better classification performance than that can be obtained from any of the weak individuals (Bauer and Kohavi 1999). Therefore, exploring and identifying the better ensemble classifier is one of the research challenge of this thesis.

5. **Visual Similarity based technique:** There exists many techniques which use logos, screenshot of entire website , Document Object Model (DOM) layout, CascadingStyle Sheet (CSS) of website, pixels, OCR, and histograms for comparison with trusted templates of websites. As these techniques are more complex than text matching techniques, an efficient visual similarity based technique is needed to detect phishing sites with low space and time complexity.

6. **Mobile Phishing:** Nowadays, use of mobile devices increased rapidly which made the attackers to target mobile users. Due to the hardware limitations in the mobile devices such as RAM, Screen size, and low computation power, detecting phishing sites in mobile environment is a challenge for the research.

7. **Feature Selection Ensemble:** Majority of the techniques use machine learning algorithms with large number of features. Ensemble learning is commonly used for classification but it can also be used for other disciplines like feature selec-

tion. Hence, exploring and identifying the better feature selection ensemble for selecting the optimal number of features is one of the research direction.

The main scope of this research work is to detect the phishing sites which are hosted on compromised servers and phishing sites which repeats over the time with new URL. The work also focuses on providing first line of fast filtering of phishing sites using only URLs. The overall purpose is to improve online user protection against phishing sites by designing feature rich machine learning framework for the phishing detection. The present thesis addresses these gaps except research gap 3 and 4 which will be addressed in the future work

2.4 SUMMARY

In this chapter, we have provided the survey of different anti-phishing techniques designed to prevent the stealing of sensitive information. We have also categorized these techniques based on their working mechanisms. The phishing indicators for each category are given along with their limitations. This discussion lead to the open issues and research challenges in phishing detection which needs attention from the researchers.

In the next chapters, we discuss our proposed techniques which falls under the category of list, heuristic and machine learning, search engine and visual similarity to detect phishing sites. The machine learning based technique includes comprehensive and rich set of features with ensemble classifier to detect phishing sites. The list and visual similarity based technique includes features extracted from noisy HTML to detect the manipulated blacklisted phishing sites. The heuristic based technique used a set of features extracted from both URL and source code to detect the phishing sites with a significant accuracy.

CHAPTER 3

A COMPREHENSIVE FEATURE BASED MACHINE LEARNING FRAMEWORK FOR THE DETECTION OF PHISHING SITES

There exist many techniques to counter the phishing sites but still the online users are tricked to reveal the sensitive information. As mentioned in Chapter 2, the techniques include feature extraction from various resources such as list, source code, search engine and visual elements but lacks a technique with comprehensive feature set that detects phishing sites with a significant accuracy. Hence, in this chapter, we present a novel classification model which includes comprehensive heuristic features that are extracted from URL, Source Code, and third-party services to overcome the disadvantages of existing anti-phishing techniques.

3.1 INTRODUCTION

Machine learning (ML) is one of the most frequently used words in the Internet world which is most widely used in various applications such as traffic predictions, health care, social media services, virtual personal assistants (Siri, Alexa or ok Google), search results refining, image recognition, video surveillance, email spam and malware filtering etc. Some of the recent techniques in the literature also applied ML on the features extracted from the websites to detect the phishing sites. These techniques use heuristic methods to extract the features from the URLs. As ML based techniques are based on heuristic features, they are able to identify the zero-day phishing attacks which make

them advantageous than list based techniques. As mentioned earlier, according to Whittaker et al. (2010), it is possible to achieve more than 99% TPR and less than 1% FPR. This influenced us to use machine learning algorithms for the classification of phishing sites.

Also, it should be noted that heuristic methods exploit the unique characteristics of phishing websites and consider them as a common parameter for detecting phishing websites. As heuristic techniques have the advantage of detecting zero-day attacks and have been used by major browsers and antivirus softwares in real time (Khonji et al. 2013), we have also used the approach of heuristic mechanism in our technique. Our technique has a risk of misclassification of legitimate sites into phishing sites (false positives) because of the disadvantage of heuristic features. To reduce the impact of false positive rate, we have also included third-party services for the generation of feature vector in the detection process. This overcomes the limitations of heuristic features.

In this chapter, we present new heuristic features (in addition to the existing features) with machine learning algorithms to reduce the false positives in detecting new phishing sites. We have also made an attempt to identify the best machine learning algorithm to detect phishing sites with high accuracy than the existing techniques. We have used eight machine learning algorithms (J48 Tree Decision, SMO, LR, MLP, BN, RF, SVM and AdaBoostM1) to classify the websites as legitimate and phishing. Based on the experimental observations, Random Forest outperformed the others. The choice of considering these machine learning algorithms is based on the classifiers used in the recent literature (He et al. 2011, Pan and Ding 2006, Xiang et al. 2011, Miyamoto et al. 2008, Zhang et al. 2014).

Attackers are using common anchor links in the entire website to overcome the anti-phishing techniques such as He et al. (2011), Pan and Ding (2006), Zhang et al. (2014), and Rao and Ali (2015b). Similarly, they also insert broken anchor links of local domain in the entire website to bypass the anti-phishing techniques (Mohammad et al. 2014b, Pan and Ding 2006, He et al. 2011, Zhang et al. 2014, Abdelhamid et al. 2014) that depend on foreign anchor features. Using our proposed features, we are able to detect phishing sites which are not being detected by existing techniques with a very

high accuracy and hence can be used for better classification of phishing websites.

From Dietterich (2000), it is noted that ensembles perform better than any single classifier. An ensemble of classifiers is a set of classifiers whose individual decisions are combined either by weighted or unweighted voting to classify the testing data. The accuracy of the ensemble method depends on the diversity of set of classifiers (i.e. they make different errors at new data points). Dietterich (2000) concluded that Random trees performs best when encountered with noisy data due to nature of diversity of training data.

From Fernández-Delgado et al. (2014), it is observed that Random Forest performs the best among all 179 classifiers on the real world classification problem. RF is also being used for detecting phishing emails (Akinyelu and Adewumi 2014, Fette et al. 2007), phishing tweets (Aggarwal et al. 2012) and malicious posts on Facebook (Dewan and Kumaraguru 2015). Whittaker et al. (2010) used Random Forest classifier for maintaining Google's phishing blacklist pages automatically. Based on the following techniques (Whittaker et al. 2010, Akinyelu and Adewumi 2014, Fette et al. 2007, Aggarwal et al. 2012), we chose the Random Forest classifier for classification of data using our novel proposed features.

Oblique Random Forests (oRF) performs better than orthogonal Random Forests (Menze et al. 2011). Hence, we compared RF with novel oRF models proposed in Zhang and Suganthan (2014), Menze et al. (2011), Zhang and Suganthan (2015) for identifying the best Random Forest classifier. From the experimental results, we observed that Principal Component Analysis Random Forest (PCA-RF) performed the best out of all oRFs. The analysis of these works in the literature influenced us to use Random Forest algorithm for training our model. To the best of our knowledge, our work is the first to use Oblique Random Forests (oRF) algorithm for classification of phishing websites.

Our work makes the following three research contributions. First, we propose five novel heuristic features (HF3, HF4, HF6, HF7, HF8) capturing key characteristics of phishing sites using the source code of a website. Second, we have experimentally

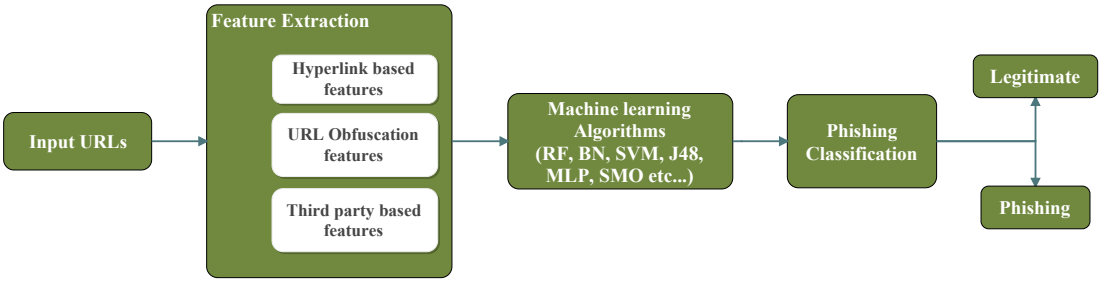


Figure 3.1: Architecture of the system

shown that third-party services like the search engine, WHOIS database and Alexa page ranking complements the heuristic techniques to detect phishing. The results of our technique illustrate that combining third-party based features with proposed features gives the classification accuracy of 99.3% with RF and 99.55% with PCA-RF as classifiers. Third, we propose a feature (HF7) which detects phishing pages when the entire website is replaced with a single image.

3.2 METHODOLOGY

The architecture of our method is given in Figure 3.1. A set of webpage URLs is fed as input to the Feature Extractor, which extracts the most significant features for the phishing detection. The extracted features are used to train the model with different machine learning algorithms for the classification of websites.

3.2.1 Feature Extractor

We have implemented a java program which uses Jsoup[1] library to download and parse the source code from a given URL. By crawling the source code of a website, we extract the features which are used in detection of phishing websites. We have classified the extracted features into three categories such as:

- URL Obfuscation features
- Third-Party based features
- Hyperlink based features

[1]http://jsoup.org/

A. URL Obfuscation features

These are the features which are found in URL of a phishing website, trying to mislead the user with various methods. Before going on to understand these features, we describe the anatomy of a typical URL.

URL is an acronym for "Uniform Resource Locator" which specifies the address of a resource such as HTML document, image, video clip, Cascading Style Sheet etc. on the Web. It consists of three parts: Protocol used to access the resource (HTTP), name of the machine hosting the resource (hostname) and path of the resource (pathname). The typical structure of an URL is as follows:

```
<Protocol>://<Subdomain>.<Primary domain>.<gTLD>.<ccTLD>/<
   Path domain>
```

where gTLD is generic Top Level Domain and ccTLD is country code Top Level Domain. For example, in URL `http://www.reg.signin.example.com.pk/secure/login/web/index.php`, we have, protocol as *http*, Subdomain as *reg.signin*, primary domain as *example*, gTLD as *com*, ccTLD as *pk*, Domain as *example.com.pk*, Hostname as *reg.signin.example.com.pk* and pathname as *secure/login/web/index.php*.

The existing features which identify visual deceptive text techniques in the URL of a website are described below. In particular, feature UF2, UF4 and UF5 are adopted from Zhang et al. (2007), Mohammad et al. (2012), Mohammad et al. (2014a); feature UF3 is adopted from Basnet et al. (2011), Mohammad et al. (2012), Mohammad et al. (2014a) and feature UF1 is a variant of one feature in CANTINA (Zhang et al. 2007).

a) *UF1-Dots in Hostname:* This feature counts the number of dots in the hostname of a URL to detect the status of a page as phishing or legitimate site. Usually, URL contains a primary domain, subdomain, gTLD or ccTLD or both. The attackers add the domain of a legitimate website as the subdomain of a phishing website. Thus, users fall for phishing by seeing the legitimate domain in the phishing URL. For example, let us consider a phishing URL targeting ebay website: `http://www.reg.signin.ebay.secure.web.login.user.phish.com/index.html`. Here, ebay is added in subdomain of phishing website (*phish.com*). The attacker includes

more number of dots to hide the actual domain of phishing site. Larger the number of dots in the hostname, higher the probability of website to be a phishing site.

$$UF1 = \text{ Number of dots in Hostname} \tag{3.1}$$

b) *UF2-URL with @ symbol:* This feature checks for the presence of special character @ in the URL. This feature is a binary feature which is set to 1 for the presence of @ in URL else set to 0. Phishers add special symbol @ after the target domain followed by actual phishing domain to the URL giving an impression as legitimate URL to the user. For example, consider a phishing URL: `http://www.signin.ebay.com/@www.phishing.com/abs/def/index.html`, user gets fooled by clicking on this link and is directed to *phishing.com* website. When the browser encounters @ symbol in URL, it ignores everything prior to @ symbol and downloads the real address which follows @ symbol.

$$UF2 = \begin{cases} 1 & \text{if @ present in URL} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

c) *UF3-Lengthy URL to hide Suspicious domain:* This feature counts the number of characters in a hostname to detect the status of a website. This feature is similar to feature UF1 except that phisher uses long URL by adding more number of characters to the URL to hide the actual phishing domain of the website. Larger the number of characters, higher the possibility of website to be a phishing site.

$$UF3 = \text{ Total length of hostname in URL} \tag{3.3}$$

d) *UF4-Using IP Address:* This feature is a binary feature which checks the presence of IP address in a given URL. If present, the feature is set to 1 else it is set to 0. Most of the legitimate sites do not use IP address as an URL to download a webpage. Hence, when the user encounters IP address, he can almost be sure that someone is trying to steal his/her sensitive information.

$$UF4 = \begin{cases} 1 & \text{if IP present} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

e) *UF5-Presence of HTTPS (HTTP plus SSL):* This feature checks for the presence of HTTPS in a given URL. If the URL contains HTTPS then the feature is set to

0 else it is set to 1. Most of the legitimate websites use HTTPS connection when there is a need of sensitive information to be forwarded. This feature is not enough for detecting phishing sites because the attackers are also able to get fake HTTPS connection to their phishing websites

$$UF5 = \begin{cases} 0 & \text{if HTTPS present} \\ 1 & \text{otherwise} \end{cases} \tag{3.5}$$

B. Third-Party service based features

In this section, we explain the third-party service based features used in our detection model. These features have been significant in detecting phishing websites. In particular, feature TF1 is adopted from Zhang et al. (2007), Mohammad et al. (2012), Mohammad et al. (2014a); feature TF2 is adopted from Mohammad et al. (2012), Mohammad et al. (2014a), Basnet et al. (2011), Garera et al. (2007); and feature TF3 is adopted from CANTINA+ (Xiang et al. 2011).

a) *TF1-Age of Domain:* This feature calculates the age of a website's domain which is extracted from WHOIS[2] database. The rationale behind this feature is that most of the phishing websites are hosted on newly registered domains resulting absence of entries in WHOIS database or age of domain as less than a year. But this is not always guaranteed in phishing sites because phishers might also host their phishing page on a compromised domain. The importance of this feature depends on the number of compromised domains used by the phishers.

$$TF1 = \begin{cases} 0 & \text{if DNS Record absent} \\ Age & \text{otherwise} \end{cases} \tag{3.6}$$

b) *TF2-Page Rank:* As phishing sites have a short lifespan and less traffic compared to legitimate sites; these sites are either not recognized or least popular rated by page ranking services like Alexa[3]. This feature calculates the page rank of a suspected website in Alexa Database. To obtain the PageRank score from the Alexa PageRank system, we just send a normal HTTP request to the URL `http://data.`

[2]https://www.internic.net/whois.html
[3]http://www.alexa.com/siteinfo/page-rank-calculator.com

`alexa.com/data?cli=10&url="+domain` and use XML parser to get the Alexa ranking.

$$TF2 = \begin{cases} 0 & \text{if Page Rank absent} \\ PageRank & \text{otherwise} \end{cases} \tag{3.7}$$

c) *TF3-Website in Search Engine Results:* Keywords which uniquely define a website are fed to a search engine to display the related URLs as results. The webpage is classified as legitimate if its base domain is matched with any of the domains of top N search results otherwise classified as a suspicious site. CANTINA used this feature for detection of a phishing site, by feeding high ranked Term Frequency - Inverse Document Frequency (TF-IDF) keywords to Google Search engine. TF-IDF algorithm is applied on the entire text of website to get unique identity terms of a suspicious website. However, CANTINA fails when TF-IDF extracts wrong keywords as website identity keywords. Hence, we skipped TF-IDF technique to calculate website identity terms and considered Title, Description and Copyright as website identifiers in our model. This feature queries the search engine with Title or Copyright or Description of a suspicious website and classifies the status of website based on the presence of its domain in top N search results. We considered N as 10 based on the experimental results of CANTINA+. Due to the deprecation of Google search API[4], we considered Bing search engine for the evaluation of this feature.

$$TF3_i = \begin{cases} 0 & \text{if local domain matched with top N} \\ & \text{search results} \\ 1 & \text{otherwise} \end{cases} \tag{3.8}$$

where *i*=1, 2, 3 for Title, Copyright and Description respectively

C. Hyperlink based features

In this section, we explain the features which are extracted from hyperlinks located in the source code of a website. A link or hyperlink is an element in an electronic document that connects from one web source to another. The web source can be an image, program, HTML document and an element within an HTML document.

[4]https://developers.google.com/web-search/

To imitate the behavior of trusted website, phishing websites would impersonate the links or hyperlinks, which point to the trusted website domain. This implies that phishing pages regularly contain hyperlinks that point to a foreign domain.

In this chapter, we consider links pointing to the local domain as local links and links pointing to the foreign domain as foreign links. For any legitimate site, the number of local links would be greater than the number of foreign links. Hence, we consider this as one of the feature to classify a suspected site as a legitimate or phishing site. Similarly, we extract various URL features such as script resource URLs, CSS URLs, Images source URLs to classify the phishing site from the legitimate site. In particular, HF1, HF2 and HF5 are adopted from Pan and Ding (2006), Mohammad et al. (2012); and features HF3, HF4, HF6, HF7, HF8 are the novel ones we proposed.

a) *HF1-Frequency of domain in anchor links:* This feature examines all the anchor links in source code of a website and compares the most frequent domain with the local domain of a website. If both the domains are similar then the feature is set to 0 i.e. website is classified as legitimate else the feature is set to 1. The rationale behind the assumption of this feature is that attackers design phishing sites sticking to similar links of legitimate domains to reduce the load of designing. For any legitimate site, the number of local links would be greater than foreign links of a website resulting most frequent domain coincide with the local domain. This makes the difference in behavior of legitimate and phishing sites and is considered as one of the features to detect phishing sites.

$$HF1 = \begin{cases} 0 & \text{if most frequent domain equals} \\ & \text{local domain} \\ 1 & \text{otherwise} \end{cases} \tag{3.9}$$

b) *HF2-Frequency of domain in CSS links, image links, and Script links:* This feature is similar to HF1, except that most frequent domain is calculated for the request URLs of CSS file, image and script file from $< src >$ and $< link >$ tags of the source code. We have calculated most frequent domain for links of CSS, image and script

files individually and also in combined files. The most frequent domain of request URLs is compared with the local domain to check for the legitimacy of a website. If the comparison is successful then the feature is set to 0 and the website is classified as legitimate else the feature is set to 1 and the website is classified as suspicious.

$$HF2 = \begin{cases} 0 & \text{if most frequent domain equals} \\ & \text{local domain} \\ 1 & \text{otherwise} \end{cases} \tag{3.10}$$

c) *HF3-Common page detection ratio in Website:* This feature calculates the ratio of frequency of the most common anchor link to the total number of links in a website. The rationale behind this feature is that phishers may redirect a few or all of the links in login page to a common page. They might skip imitating the links of a legitimate site and rather stick to a single URL with the local domain. We believe that attackers take less effort in designing phishing site by mimicking the login page. Hence, they do not very often use a different anchor link and redirect most of the anchor links to a common page. For example, sample code of three anchor links of a phishing site pointing to a common URL is shown below.

```
<html lang=``en">
<head>...</head>
<body>
...
<a href="http://tutorials.phishingsite.com/ion/index.
   html">forgot password</a>
<a href="http://tutorials.phishingsite.com/ion/index.
   html">Contact us</a>
<a href="http://tutorials.phishingsite.com/ion/index.
   html">Help</a>
...
</body>
</html>
```

This kind of scenario in phishing sites results in high common page detection ratio. But in legitimate sites, the anchor links mostly point to different filenames which results in less common page detection ratio. Note that, if there are no anchor links present in a website then the feature is set to 0 else it is set to Common page detection ratio value.

$$HF3 = \begin{cases} \frac{al_{c_w}}{al_{n_w}} & \text{if } al_{n_w} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

where al_{c_w} is frequency of most common link and $al_{n_w}$is total number of anchor links in a website. From the experimental results, it is observed that this feature is significant in classifying the phishing websites. The dataset upon training with this feature individually resulted in accuracy of 83.73% as shown in Figure 3.4

d) *HF4-Common page detection ratio in Footer:* This feature calculates the ratio of frequency of most common anchor link to the total number of links in footer section of the website. This feature is similar to HF3 feature, except that it is concentrated on footer section of the website.

$$HF4 = \begin{cases} \frac{al_{c_f}}{al_{n_f}} & \text{if } al_{n_f} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

where al_{c_f} is frequency of most common anchor link in footer and $al_{n_f}$is total number of anchor links in footer section of a website.

e) *HF5-Null Links ratio in website:* We consider anchor links with null value or anchor links starting with null value as Null links. This feature calculates the ratio of Null links to the total number of anchor links in a website. The main intention of an attacker is to make the online user stay on the same page until he submits his sensitive information. Hence, when a user clicks on any link present in the login page, he is redirected to the same login page. This is being done by adding following code as an anchor link.

```
<a href="#"> or <a href="#content"> or
<a href="JavaScript :: void(0)">
```

$$HF5 = \begin{cases} \frac{al_{null_w}}{al_{n_w}} & \text{if } al_{n_w} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

where al_{null_w} is number of null links and $al_{n_w}$is total number of anchor links in a website.

f) *HF6-Null Links ratio in Footer:* This feature extracts ratio of null links in footer section of website to the total number of footer links in a website. In this feature, we identify links with null value in footer section rather than on entire HTML, which

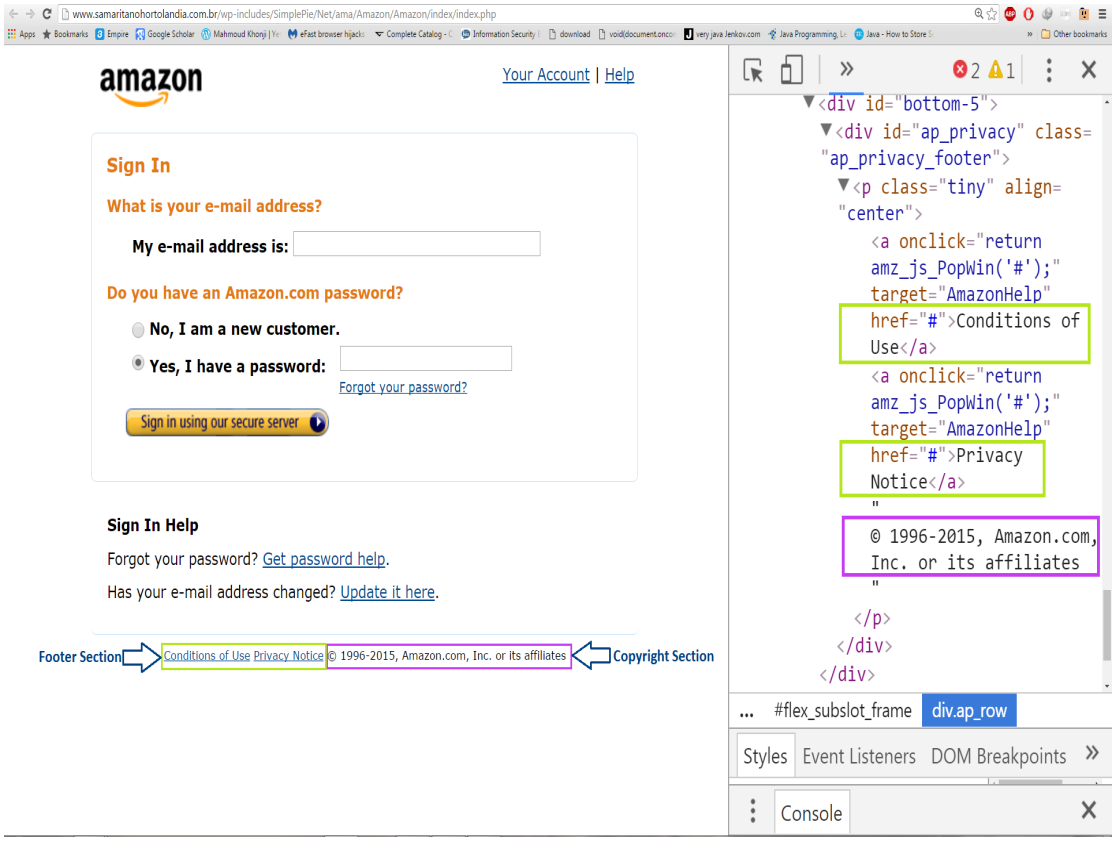


Figure 3.2: Phishing site with null links, footer and copyright section

reduces false positive ratio and time complexity compared to HF5 feature. We might deal with less number of links rather than all links in the website. Most of the legitimate websites do not often use null links in the footer section of a website but the phishing sites usually insert null links in the footer section to make the user stay on the same page until the sensitive information is submitted by the user. Figure 3.2 shows the Screenshot of phishing site along with null footer link.

$$HF6 = \begin{cases} \frac{al_{null_f}}{al_{n_f}} & \text{if } al_{n_f} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

where al_{null_f} is number of null links in footer and al_{n_f} is total number of anchor links in footer section of a website.

g) *HF7-Zero links presence:* This feature checks for the presence of anchor links in the body section of the source code of a website. If absent, then the feature is set to 1 else it is set to 0. The rationale behind this feature is that there exists at least one an-

chor link such as `signin`, `signup`, `forgot password` or others in legitimate site's login page whereas in phishing websites, attackers replace entire legitimate site content with a single background image. We termed these kinds of phishing sites as *Image based Phishing*. Attacker attains the above case by taking a screenshot of targeted legitimate site and embeds it as background page for their phishing site. On top of the screenshot they insert input fields such as username or email, password and Login Button following the same styles of targeted legitimate domain.

Figure 3.3 explains how the phishing is carried out by replacing textual content with a single background image. Figure 3.3(a) shows the background image used by the attacker. The location of background image can be seen in address bar as `https://gator3231.hostgator.com/~cheaks/bost/FTEROO09K/img/bg.PNG`. Figure 3.3(b) shows the form imitating the same style of legitimate site. This figure is taken by removing the background image from the source code of phishing site. Figure 3.3(c) shows the fully developed phishing site with background image and input fields embedded into the body of website. The background image location can be seen in the inspect code of the website. The input fields are properly aligned by adjusting the position of fields with z-index, absolute position, height, width, top and left dimensions. For better understanding, sample source code of phishing site is given below:

```
<html lang=``en">
<head>...</head>
<body style="background-image: url(img/bg.PNG);
   background-repeat: no-repeat;"> <!--Background image
   inserted here-->
<form method="post" action="LOGIN.php" style="z-index:
   1; width: 395px; height: 232px; position: absolute;
   top: 208px; left: 211px">
<div style="z-index: 1; width: 340px; height: 40px;
   position: absolute; top: 53px; left: 24px">
<input style="z-index: 1; position: absolute; top: -3px;
    left: 0px; width: 338px; height: 38px" type="email">
</div>
... <!--similarly password and submit button are also
   properly aligned to look like legitimate site-->
</form>
</body>
</html>
```

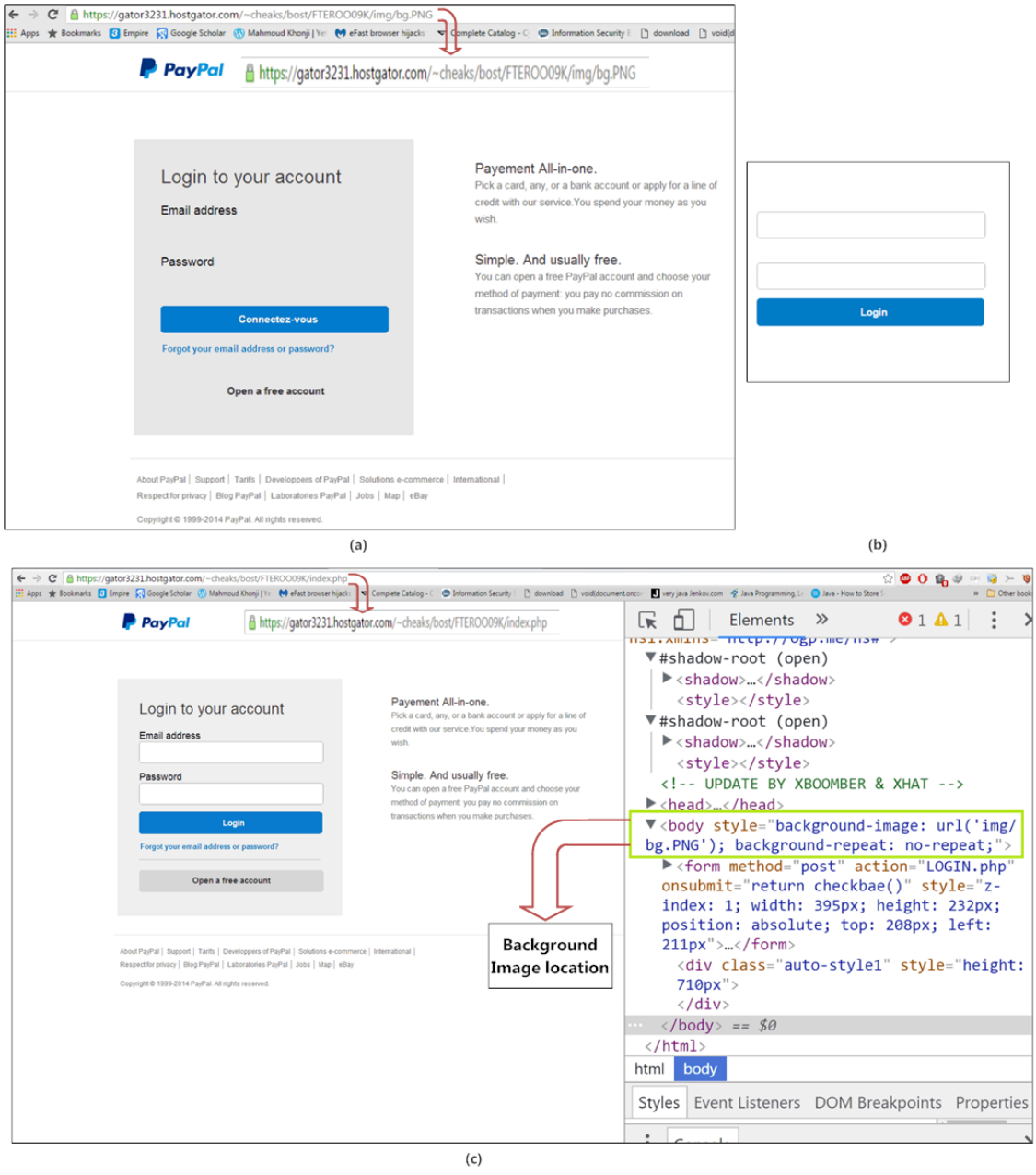


Figure 3.3: (a) Background image used for phishing, (b) Phishing site without background image, (c) Fully developed phishing site along with its source code

Hence, HF7 can be defined as given below.

$$HF7 = \begin{cases} 0 & \text{if } al_n = 0 \\ 1 & \text{otherwise} \end{cases} \tag{3.15}$$

where al_n is number of anchor links in the website.

h) *HF8-Broken links ratio:* This feature extracts ratio of *not found* links to the total number of links in a website. In legitimate sites, when all the links are connected either they return 200 Ok HTTP status code indicating server has accepted the request or sends 404 status code indicating page not found in that server. Legitimate sites rarely return status code as 404, whereas in phishing sites, attackers include links with random filenames leading to not found pages. Hence we have used this feature to classify the legitimate sites and phishing sites.

$$HF8 = \begin{cases} \frac{nf_l}{l_n} & \text{if } l_n > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

where nf_l is number of *not found* links and l_n is total number of links in a website.

3.2.2 Machine Learning algorithms

To evaluate the performance of our feature set, we used eight machine learning algorithms in training our model including J48 tree, RF, SMO, LR, MLP, BN, SVM and AdaBoostM1 (AM1). We performed comparison of all the classifiers with a primary goal of choosing best classification model among them. These algorithms have been used by previous researchers in Xiang et al. (2011), Zhang et al. (2014) for training their model. All the above machine learning algorithms are implemented in R using RWEKA and Kernlab package. From the extensive experiments, we found that Random Forest performed the best among the other algorithms consistently. We also performed comparison of RF with various Oblique Random Forest for identifying best RF classifier. The reason behind the high performance of Random Forest is due to the ensemble of weak learning models (tree predictors) with diversity of training data among them. Note that, combining multiple weak classifiers into one aggregated classifier leads to better classification performance than that of any individual classifier (Ho 1998). It is because the probability of making mistakes at the same place in all of the trees is very low. Hence, it achieves high accuracy than individual weak classifier.

3.3 EXPERIMENTATION AND RESULTS

Given a URL of suspected website, we have to decide whether it is a legitimate or phishing site. We used a java library called Jsoup to download the source code of static

pages and parsed them to extract significant features used in detecting status of a website. The reason behind choosing Jsoup is that it can handle non-well-formed HTML very effectively and also it has its own API providing functionality to select elements of DOM. The other parsers like tagSoup, Jtidy, NekoHTML and HTML Cleaner are lenient to certain degree with non-well-formed HTML.

3.3.1 Tools used

We have implemented a java program to extract all the features from the websites in java platform standard edition 8. We have collected the URLs of live phishing sites from PhishTank and legitimate sites from Alexa database. We have written a snippet of code to automate the process of extracting features and store in a file. This file of features is given as input to the classifier for calculating the effectiveness of our features in detecting the phishing sites. We have simulated all the machine learning algorithms in R language using RWeka, Kernlab and ObliqueRF packages. RWeka contains interface code which connects to Weka, a data mining tool which provides implementation of machine learning algorithms. Kernlab is an extensible package which consists of kernel-based machine learning methods. We used Kernlab for the implementation of SVM. We employed the methods presented by Menze et al. (2011) and Zhang & Ponnuthurai (Zhang and Suganthan 2014, Zhang and Suganthan 2015) for comparison of RF with various oRF in R and Matlab respectively.

3.3.2 Dataset used

To calculate the performance of our model, we collected 2119 phishing sites from PhishTank and 1407 legitimate sites from Alexa Database, as shown in Table 3.1. We divided the dataset into two sets: training set and test set. 75% of original dataset is taken as training set which is used to train the model and remaining 25% of the dataset is taken as test set which is used to evaluate the model.

3.3.3 Experimental evaluation

We conducted five experiments to assess the performance of our model trained with various machine learning classifiers. All the experiments were conducted with the same

Table 3.1: Dataset used in our model

Type	Source	Sites	URL
Legitimate	Alexa's Top websites	1407	http://www.alexa.com/topsites
Phishing	PhishTank database	2119	http://www.phishtank.com/developer

dataset of 3526 (1407+2119) websites. Each experiment was repeated ten times with a randomly selected training set. The average value of each metric was considered as final value such that biasing is avoided. In Experiment 1, we evaluated our model with all the features including third-party based features. In Experiment 2, we evaluated our model excluding third-party service based features. In Experiment 3, we evaluated only third-party service features used in predicting phishing websites. In Experiment 4, we compared our model containing all the features with CANTINA and CANTINA+. In Experiment 5, we compared orthogonal Random Forest with various Oblique Random Forest algorithms.

Experiment 1- Evaluation of heuristics features including third-party service features

In this experiment, we assessed heuristic features including third-party service features with eight different classifiers. We tested our system with a dataset of 3526 websites comprising of both legitimate and phishing websites. We performed comparison of all the classifiers with a primary goal of choosing the best classification model among them. Random Forest algorithm performed the best with an average prediction accuracy of 99.31% followed by J48 tree decision algorithm with an accuracy of 98.98% as shown in Table 3.2. All the machine learning algorithms have classified the websites with an acceptable true positive rate which indicates the goodness of our proposed features in classifying the websites.

Experiment 2- Evaluation of heuristics excluding third-party service features

In this experiment, we assessed heuristic features excluding third-party service features with eight different classifiers. We tested the scheme with the same dataset used in Experiment 1 to check the effectiveness of only heuristics in detecting phishing web-

Table 3.2: Evaluation of heuristics features including third-party service features

Metrics	RF	J48	LR	BN	MLP	SMO	AdaboostM1	SVM
Sensitivity	0.9944	0.9901	0.9597	0.9921	0.9581	0.9458	0.9807	0.9714
Specificity	0.9910	0.9895	0.9413	0.9825	0.9407	0.9220	0.9587	0.9430
Precision	0.9942	0.9930	0.9600	0.9883	0.9612	0.9476	0.9725	0.9609
ACC	0.9931	0.9898	0.9522	0.9882	0.9512	0.9363	0.9718	0.9594
ERR	0.0069	0.0102	0.0478	0.0118	0.0488	0.0637	0.0282	0.0406

Table 3.3: Evaluation of heuristics excluding third-party service features

Metrics	RF	J48	LR	BN	MLP	SMO	AdaboostM1	SVM
Sensitivity	0.9427	0.9403	0.9351	0.9036	0.9265	0.9149	0.9079	0.9438
Specificity	0.9159	0.8825	0.8877	0.9017	0.8992	0.8868	0.8588	0.8893
Precision	0.9438	0.9250	0.9249	0.9339	0.9323	0.9235	0.9099	0.9286
ACC	0.9319	0.9176	0.9160	0.9027	0.9156	0.9036	0.8888	0.9222
ERR	0.0681	0.0824	0.084	0.0973	0.0844	0.0964	0.1112	0.0778

sites. We performed the comparison of all the classifiers, among them Random Forest algorithm performed the best with an average prediction accuracy of 93.19%, followed by J48 Tree decision algorithm with an accuracy of 91.76% as shown in Table 3.3. This experiment shows a drop of phishing detection accuracy by 6.12% due to lack of third-party service features. The results of this experiment demonstrate that third-party service features have significant influence on the performance of our model.

Experiment 3- Evaluation of third-party service features

In this experiment, we considered only third-party service features for prediction of phishing websites. The results are tabulated in Table 3.4. From the results, it is evident that third-party service features are significant in detecting phishing websites. Amongst all classifiers, Random Forest performed the best with an average prediction accuracy of 98.55% and an error of 1.45%. Comparing the results of Experiment 1 with Experiment 3 using Random Forest algorithm, there is a drop of 0.76% in accuracy. However, the results illustrated that combining third-party services features with heuristic features made the classification more accurate.

Table 3.4: Evaluation of our model with only third-party service features

Metrics	RF	J48	LR	BN	MLP	SMO	AdaboostM1	SVM
Sensitivity	0.9885	0.9830	0.9125	0.9826	0.9179	0.8999	0.9929	0.9321
Specificity	0.9810	0.9752	0.8253	0.9723	0.8171	0.8250	0.9548	0.8262
Precision	0.9875	0.9834	0.8859	0.9816	0.8827	0.8839	0.9704	0.8905
ACC	0.9855	0.9799	0.8775	0.9785	0.8772	0.8698	0.9776	0.8898
ERR	0.0145	0.0201	0.1225	0.0215	0.1228	0.1302	0.0217	0.1102

Experiment 4- Comparison of our work with other techniques

In this experiment, we implemented existing works such as CANTINA and CANTINA+ and tested them with the same dataset used in Experiment 1. We compared the performance of our work with these techniques with an aim of evaluating the performance of our features against the baseline models. This experiment demonstrates that our model offers more accuracy than these methods in detecting phishing websites. Random Forest algorithm performed the best with an accuracy of 99.31% followed by J48 algorithm with an accuracy of 98.98% as shown in Table 3.5

Experiment 5- Comparison of RF with oRF

In this experiment, we evaluated the performance of various Oblique Random Forest classifiers (Zhang and Suganthan 2014, Menze et al. 2011, Zhang and Suganthan 2015) and compared with orthogonal RF to identify the best classifier for detection of phishing sites.

We employed the methods of various oRF such as `MPRaF-P`, `MPRaF-T`, `PCA-RF` and `LDA-RF` proposed by Zhang & Ponnuthurai (Zhang and Suganthan 2014, Zhang and Suganthan 2015). `MPRaF-P`, `MPRaF-T` means MPSVM based Oblique Random Forest with axis-parallel split, Tikhonov regularization respectively and `PCA-RF`, `LDA-RF` are PCA based RF and LDA based RF respectively.

We also employed the method proposed by Menze et al. (2011) for simulating various oRFs such as `oRF-svm`, `oRF-ridge`, `oRF-ridge_slow`, `oRF-pls`, `oRF-log` and `oRF-rnd`. The decision hyperplanes in these oRFs are based on support vector machine, ridge regression (fast and slower versions), partial least squares regression

Table 3.5: Comparison of our work with other techniques

Classifiers	Techniques	Sensitivity	Specificity	Precision	ACC	ERR
RF	**CANTINA**	0.9154	0.8559	0.9057	0.8918	0.1082
	CANTINA+	0.9889	0.9949	0.9967	0.9913	0.0087
	Our Work	0.9944	0.9910	0.9942	0.9931	0.0069
J48	**CANTINA**	0.9210	0.8379	0.8958	0.8880	0.112
	CANTINA+	0.9886	0.9896	0.9931	0.9890	0.011
	Our Work	0.9901	0.9895	0.9930	0.9898	0.0102
LR	**CANTINA**	0.8922	0.7973	0.87189	0.8550	0.145
	CANTINA+	0.9248	0.8784	0.9177	0.9060	0.094
	Our Work	0.9597	0.9413	0.9600	0.9522	0.0478
BN	**CANTINA**	0.9109	0.8064	0.8782	0.8696	0.1304
	CANTINA+	0.9914	0.98426	0.9881	0.9878	0.0122
	Our Work	0.9921	0.9825	0.9883	0.9882	0.0118
MLP	**CANTINA**	0.9143	0.8440	0.8981	0.8862	0.1138
	CANTINA+	0.9454	0.9009	0.9343	0.9273	0.0727
	Our Work	0.9581	0.9407	0.9612	0.9512	0.0488
SMO	**CANTINA**	0.8922	0.7936	0.8661	0.8528	0.1472
	CANTINA+	0.9273	0.8763	0.9197	0.9071	0.0929
	Our Work	0.9458	0.9220	0.9476	0.9363	0.0637
AdaboostM1	**CANTINA**	0.8925	0.7987	0.8692	0.8549	0.1451
	CANTINA+	0.9894	0.9526	0.9685	0.9745	0.0255
	Our Work	0.9807	0.9587	0.9725	0.9718	0.0282
SVM	**CANTINA**	0.9356	0.8128	0.8844	0.8871	0.1129
	CANTINA+	0.9527	0.9011	0.9352	0.9320	0.068
	Our Work	0.9714	0.9430	0.9609	0.9594	0.0406

and randomness respectively. All the variations of above mentioned Oblique Random Forests and orthogonal Random Forest were constructed with 100 decision trees (*ntree*=100) and tuned with *mtry*=4, where *mtry* is subset of attributes needed for split. *mtry* is calculated as square root of total number of attributes.

$$mtry = max(sqrt(ncol(x)), 2) \tag{3.17}$$

Table 3.6: Comparison of RF with oRF

Techniques	Sensitivity	Specificity	Precision	ACC	ERR
RF	0.9945	0.9923	0.9891	0.9932	0.0068
MPRaF-T	0.9672	0.9865	0.9806	0.9785	0.0215
MPRaF-P	0.9945	0.9942	0.9918	0.9943	0.0057
PCA-RF	0.9945	0.9961	0.9945	0.9955	0.0045
LDA-RF	0.9750	0.9809	0.9723	0.9785	0.0215
oRF-pls	0.9874	0.9364	0.9594	0.9668	0.0332
oRF-svm	0.9804	0.9622	0.9748	0.9731	0.0269
oRF-ridge	0.9891	0.9424	0.9626	0.9705	0.0295
oRF-ridge slow	0.9885	0.9369	0.9591	0.9678	0.0322
oRF-log	0.9925	0.9407	0.9623	0.9720	0.0280
oRF-rnd	0.9863	0.9741	0.9817	0.9831	0.0169

Note that, all the models were trained with 75% of dataset and tested with remaining dataset. The experiments are repeated ten times for each oblique Random Forest and then the average value is considered for the comparison. The results are shown in Table 3.6. From the results, it is observed that PCA-RF outperformed all of the various RF with an accuracy of 99.55% followed by MPRaF-P with an accuracy of 99.43%. It may be observed that oRF with trained pls, ridge, ridge_slow, svm and rnd achieved less accuracy than RF. It is due to the fact that, the oRF with above training models outperforms RF in case of heavily correlated and high dimensional data but their performance falls below RF in case of factorial or discrete data (Menze et al. 2011). Note that, our extracted phishing website features are not heavily correlated and had maximum discrete data. The usage and popularity of Random Forest is increasing due to its flexibility and simplicity on complex data. It overcomes the problem of overfitting and handles the sparse/missing data well.

3.4 DISCUSSION

We used Bing Search engine for comparing suspicious domain with top N search results, WHOIS database for calculating the age of domain, Alexa page ranking for identifying the rank of website as third-party service features. These features make our

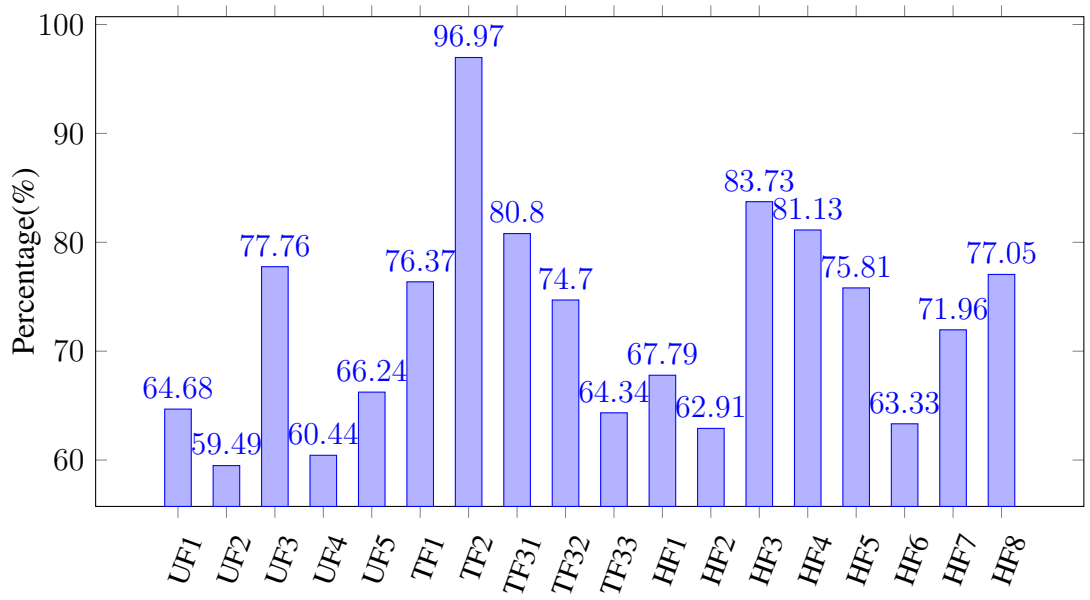


Figure 3.4: Performance of individual features

method suffer from network latency due to querying each service.

Alexa page rank feature fails when the attacker uses URL as a compromised domain. It also creates false positive in classifying the newborn legitimate sites as phishing sites. The failure is due to the short lifespan of phishing site whereas lifespan of reputed legitimate site would be at least greater than a year. WHOIS-based feature also suffers from the same problem of Alexa feature i.e. it fails, when the compromised domain is used for hosting a phishing site resulting in greater domain age of the phishing page.

To bypass null links feature in a login page, phishers might insert the current URL of phishing page as a link value instead of null (#) value. However, this case is countered by the Common page detection ratio filter which is based on the most frequent URL. They can also bypass the null links or Common page detection ratio filter by copying the login page in many folders and then links are pointed to these files.

The main intention of attackers is to design a phishing website with the least effort. Hence, the attacker uses the same links pointing to images of legitimate site instead of pointing to the local domain. This makes the phishing attack detection possible by the hyperlink based feature. But with an extra effort, the attacker can bypass this feature by using links pointing to its own domain.

Sometimes legitimate sites also behave like phishing sites consisting of a large number of foreign links pointing to their mapped domains. These legitimate sites, request images, scripts, style and other graphics files from their mapped domains. For example, let us consider a legitimate website: `https://www.paypal.com/in/webapps/mpp/home`,which requests the scripts, images, CSS files from paypalobjects.com domain. These type of websites produce high false positive rate by reducing the effect of HF1 and HF2 features. To address the limitation of these features, we collected a set of legitimate sites and their mapped domains associated with each other in a list. Whenever a feature encounters a foreign domain listed in a mapped domain, its associated domain is compared with the local domain of a website. The mapped domain is considered as a local domain if the comparison is successful otherwise treated as foreign domain.

The results of our experiments clearly demonstrate that, the third-party service features have a significant influence on the performance of our model. By incorporating these features, our model outperformed existing techniques. We also evaluated the contribution of individual features in detection of phishing websites with an aim of identifying top performing features. The performance of each individual feature is calculated with an accuracy metric using Random Forest algorithm as shown in Figure 3.4.

It is observed that the following features achieved an accuracy of above 70%. The features are length of hostname (UF3), age of domain (TF1), Page Ranking (TF2), search engine title (TF31), search engine copyright (TF32), Common page detection ratio in website (HF3), Common page detection ratio in footer (HF4), null links in website (HF5), Zero links presence (HF7) and broken links ratio feature(HF8). Out of five proposed novel features, four of them result as top performing features in detecting phishing sites. It is also observed that each feature used for training our model has an accuracy of over 60% which reveals the goodness of our features in detecting phishing websites. But it should be noted that the statistics are dependent on the quality and quantity of training data which is further used in training the model for the classification of websites.

3.5 SUMMARY

In this chapter, we have presented a novel model for the detection of phishing sites using heuristic features extracted from various sources. Individual performance of new features such as HF3, HF4, HF5, HF7 had achieved an accuracy of above 70% which shows the contribution of proposed features in improving the accuracy of the detection model. We compared our work with CANTINA and CANTINA+ to demonstrate the advantage of our model over these approaches in detecting phishing sites. The features used in our model to detect phishing do not depend on initial image database or prior knowledge such as web history.

Our model also detects phishing sites which imitate legitimate sites by replacing the website content with an image, which most of the anti-phishing techniques fail to detect. Our model detects zero-day phishing sites which the list based techniques fail to detect. With the help of these rich set of heuristic features, our model is able to achieve high detection rate of 99.55% and low false positive rate of 0.45% using Oblique Random Forest algorithm.

Although this technique achieved high detection accuracy, it has some drawbacks. The model uses third-party services for classification of websites resulting in dependency on the speed of third-party services. Also, due to the use of third party services there exists more chance of misclassifying phishing sites hosted on compromised domains. Hence, in the next chapter, we present a technique which detects phishing sites hosted on compromised domains. Also, we present an improved search engine mechanism to detect the PSHCS and newly registered legitimate sites.

CHAPTER 4

DETECTION OF PHISHING SITES HOSTED ON COMPROMISED SERVERS

Phishing attacks are on the rise and are hosted on compromised domains such that legitimate behavior is embedded into the designed phishing site to overcome the detection. The traditional heuristic techniques using HTTPS, search engine, Page Ranking, and WHOIS information may fail in detecting phishing sites hosted on the compromised domain. Moreover, list-based techniques fail to detect phishing sites when the target website is not whitelisted. In this chapter, we present two techniques for the detection of phishing sites hosted on compromised servers along with malicious registered phishing sites. The first technique (**Curb-Phish**) is a novel heuristic technique that detects the phishing websites hosted on compromised domains by comparing the login page and home page of the visiting website. The hyperlink based features are used to detect phishing sites which are maliciously registered. The advantage of the technique is that it is lightweight and it does not make use of third-party services.

To improve the detection accuracy, the third-party services such as search engine, page rank and WHOIS can be used. Hence, we have presented an improved search engine based approach (**Jail-Phish**) which detects phishing sites hosted on compromised servers along with newly registered legitimate sites. This technique compares the suspicious site and matched domain in the search results for calculating the similarity score between them. For both the techniques, similarity scores between the pages are used for the classification of phishing websites.

4.1 INTRODUCTION

The main intention of this chapter is to detect the phishing sites hosted on compromised server based on the similarity score between the pages. We argue that there exists a similarity of some degree between the pages of same website such as copyright, brand names, filenames, logos and images etc. in the legitimate sites whereas on the other side, the dissimilarity within the pages is high in phishing sites hosted on compromised servers.

Attackers distribute phishing sites with the use of phishing toolkits (Britt et al. 2012; Mohammad et al. 2015; Rao and Pais 2017) using the technique given in Figure 4.1. The phishing toolkits are zip files or tar files which consists of copied legitimate login page (HTML, PHP etc), and resource files (images, CSS, logos, Javascripts etc). The zip file is copied to a web server for hosting the phishing site. The attacker may choose different web servers such as compromised web server, free hosting providers or paid web server. The designer of the phishing toolkit takes good care of the phishing files such that traditional anti-phishing techniques are easily bypassed. One such bypassing trick is to skip or misspell the brand names in the copied legitimate site. Mostly the brand names are found in the title, URL, copyright, headers, description etc. The other tricks include removal of anchor links that point to target legitimate URLs with NULL anchor links (Mohammad et al. 2012; Rao and Ali 2015b; Rao and Pais 2019a; Srinivasa Rao and Pais 2017). Note that, the attacker can also design the phishing site with his own code starting from scratch but it is highly expensive with respect to time and design.

In the recent APWG 2017 first half report, APWG claimed that there is an increase in number of phishing sites hosted on compromised domains APWG (2017). According to Moore and Clayton (2007) study, they observed that 76% of designed phishing sites were hosted on compromised servers. If we consider the statistics of various APWG reports collected during the period from 2009 to 2014 [1], it is observed that at least 70% of the phishing sites were hosted on the compromised servers for each year. We confined the statistics to a graph which shows the number of phishing attacks conducted through malicious registrations and compromised domains as shown in Figure 4.2. From the

[1] https://www.antiphishing.org/resources/apwg-reports/

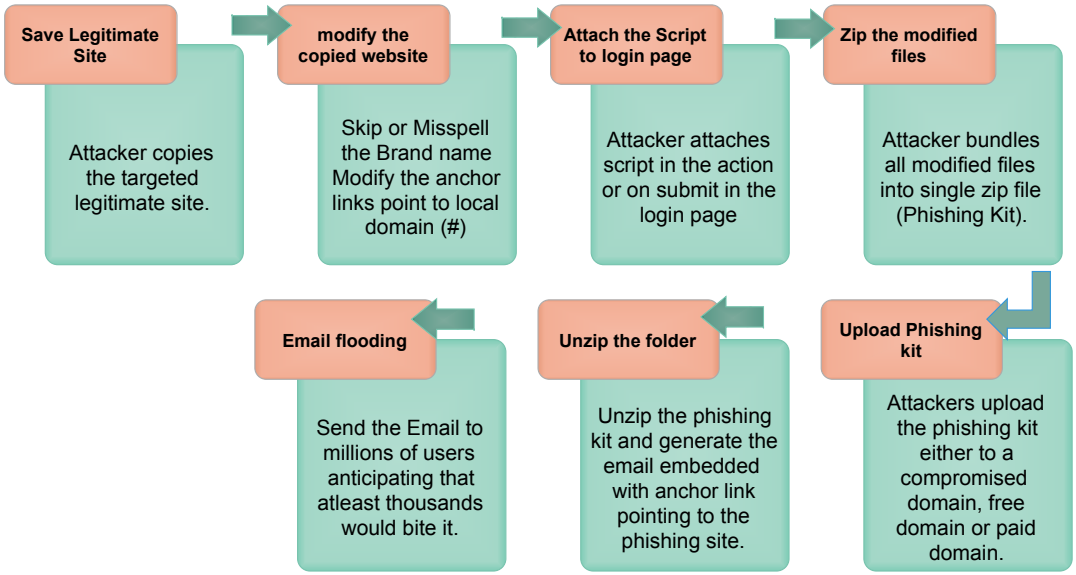


Figure 4.1: Distribution of Phishing Kit

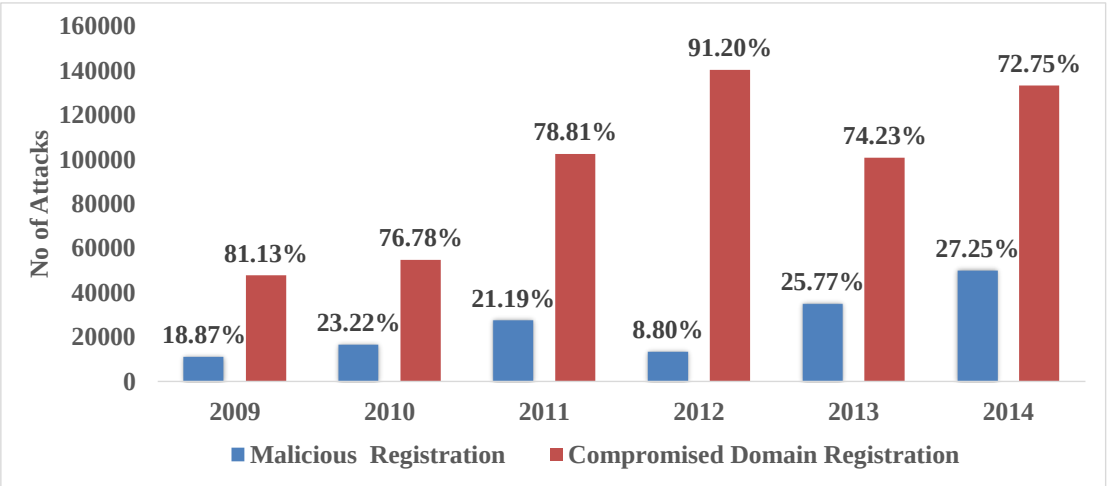


Figure 4.2: Comparison of Phishing attacks during 2009 vs 2014

graph, it is evident that along with the rise of phishing attacks over the years, the number of phishing sites using compromised servers for hosting has also increased. Note that, the statistics of compromised server attacks for 2015 to 2017 were not mentioned in the graph as they are not available in APWG 2015 to 2017(APWG 2017). From these statistics, we believed that there is a need for technique which detects the phishing sites that are hosted on compromised servers.

The advantage of hosting phishing site on the compromised server over the malicious registration is that, the designed phishing site also carries the same reputation as

that of the compromised domain. The reputation may include page ranking, visibility of domain in search engine results, and an age of domain with WHOIS database etc. In addition to the above, there is no requirement of buying the domain or resources for maintaining their phishing page. Note that, due to its drawn reputation score from the legitimate site, the phishing site will have more chance to be alive for a longer span of time than malicious registered phishing domain. While storing the designed phishing kit in the compromised server, attackers leave the compromised domain pages intact such that no suspicious behavior is observed by the owner of the legitimate site. The drawn reputation of legitimate site enables the phishing page to bypass the blacklist techniques (Drew and Moore 2014; Prakash et al. 2010; Rao and Pais 2017) and third-party service based detection techniques (Chiew et al. 2015; Moghimi and Varjani 2016; Rao and Pais 2019a; Xiang et al. 2011; Zhang et al. 2007).

4.2 CURB-PHISH : A HEURISTIC TECHNIQUE TO DETECT PSHCS

The contributions of our first technique are summarized as follows.

- Proposed efficient similarity-based features for detection of phishing sites hosted on compromised servers.
- Proposed TWSVM based phishing detection (this is the first attempt using TWSVM) for the websites hosted on compromised servers and other malicious registered sites.
- Proposed hyperlink-based features for detecting single / common page phishing sites and image based phishing sites
- Proposed technique is robust to different URL masquerade techniques and can also detect zero-day phishing attacks.

The main advantages of our work over the existing whitelist or blacklist based techniques are as follows. Firstly, it does not require any database of pre-approved images or DOMS or CSS for the comparison which results in reduction of space and computation complexity.

Secondly, for extracting most important identity keywords, we chose least Term Frequency and Inverse Document Frequency (TF-IDF) score calculated on different sections of the same webpage in spite of large corpus of all websites which again results in the reduction of computation and space complexity.

Third, our work is robust to masquerade techniques (misspell or omit brand names in URL) of new phishing sites with compromised domain because the phishing page is designed to be visually similar to that of the target website. Hence, there exists structural and textual dissimilarity between the phishing page hosted on the compromised server and its corresponding home page.

4.2.1 Methodology

Attackers copy the phishing page into a compromised server such that resources and purchase of domain are avoided. It also gives an advantage of carrying the same reputation of compromised domain and is able to bypass search engine and URL based techniques. Due to this behavior of storing phishing page in compromised server, the phishing page looks significantly different from the compromised domain pages.

Based on the above observation, the basic idea of our work is to find the degree of dissimilarity between the login page of phishing site and home page of the compromised domain. If the website is not compromised then there exists some similarity between any two pages of the website. We divide our work into two phases. Firstly, we extract the hybrid features consisting of basic URL, similarity and hyperlink based features for the generation of feature vectors. Secondly, we apply the machine learning algorithm on the generated feature vectors for performing the classification on the extracted data. The reason to choose the machine learning classifier is due to the greater accuracy achievement in the real-time applications.

The architecture and stepwise working of Curb-Phish is given in Figure 4.3 and Table 4.1 respectively. To ease our discussion we define the following terms.

- **S** - Suspicious site to be checked.
- **H** - Home page of given suspicious URL (S).

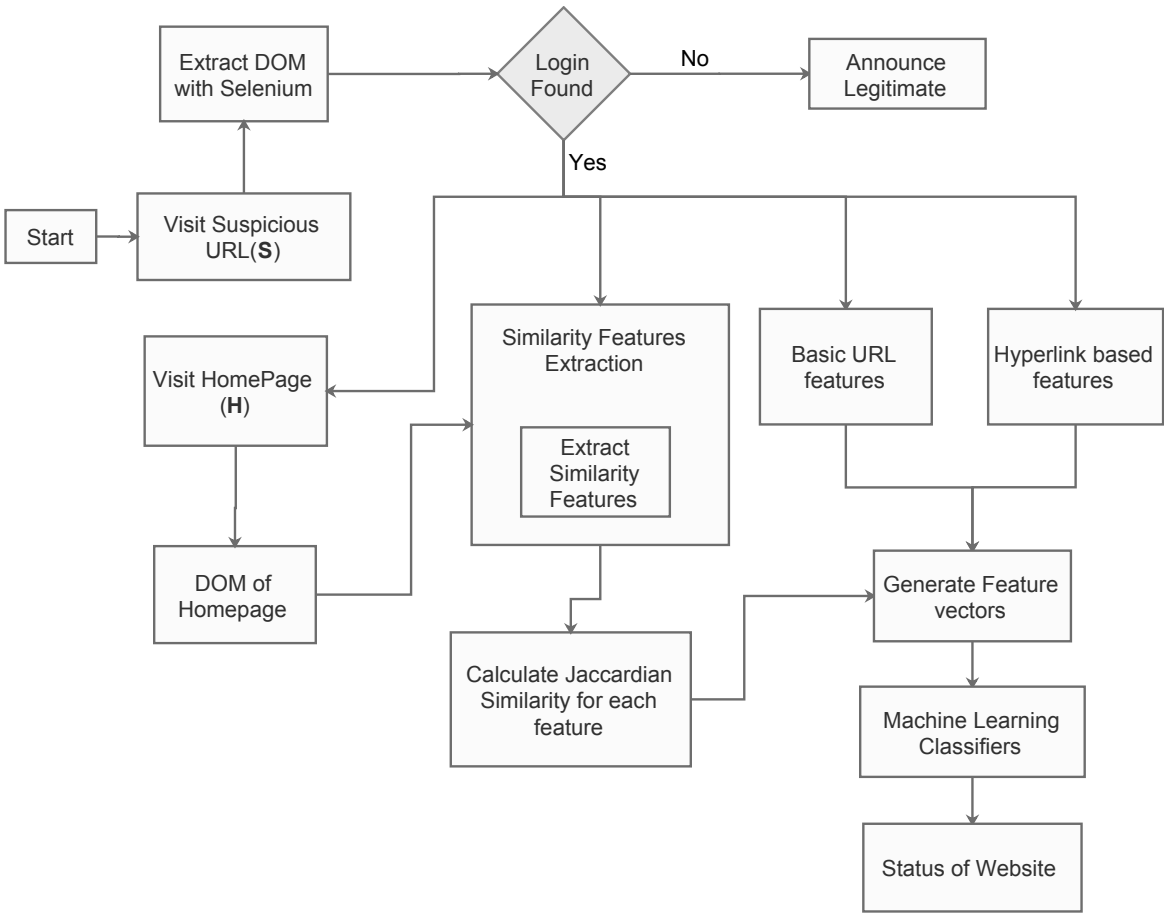


Figure 4.3: Architecture of Curb-Phish

• **Target Website** - It is the website which is imitated by the attacker.

• **Website Identity** - The real brand name of a website.

• **PSHCS** - phishing sites hosted on compromised server.

Our model can be applied to any URL for the calculation of similarity. In this case for the phishing URL detection, our model is applied on login pages of given URL since sensitive information is gathered from the login page. Normally, the login page is identified at three locations 1) Visiting page of URL 2) Modal window and 3) Iframes. Attackers use Iframes for loading the imitated target website's content such that the anti-phishing techniques which rely on source code of websites are easily bypassed. This is due to the fact that content in IFrame is hidden from the source code of the website. Hence, we used Selenium Webdriver to identify the login page at all the above locations. The page is categorized as a login page when it contains at least one input field

Table 4.1: Working of our model

Step 1 :Visit the given suspicious URL
Step 2: Extract the Document object Model (DOM) using Selenium WebDriver
Step 3: Feed the DOM to Jsoup API for parsing the source code.
Step 4: Extract the similarity based features (P1-P8) required for the detection of phishing website hosted on compromised server.
a) Save all the features in an ArrayList of features where each feature consists of set of words.
Step 5: With the same Webdriver, visit the homepage and repeat the steps 2,3, and 4
Step 6: Calculate the Jaccard similarity coefficient for home page *H* and suspicious page *S*.
Given two sets A and B where A, B $\in$ single feature in H and S respectively, then $$J(A,B) = \frac{\lvert A \cap B \rvert}{\lvert A \rvert + \lvert B \rvert - \lvert A \cap B \rvert} \quad (4.1)$$
where J(A,B)$\in$ [0,1] and J(A,B)=1 when both A and B are empty.
Step 7: Extract the features (R1-R5) and (P9, P10) from the given input suspicious URL (*S*).
Step 8: Generate the feature vectors based on R1-R5 and P1-P10.
Step 9: Feed these feature vectors of all legitimate and phishing sites to machine learning classifiers such as SVM, TWSVM and PSVM etc.
Step 10: Do the classification with 10-fold cross-validation and tuned parameters to get the status of website as phishing or legitmate

with `type=password` otherwise categorized as non-login page. All the login pages undergo Feature Extraction phase and for the non-login pages, the detection process is skipped as the user does not have any way to give his/her sensitive information.

Feature Extraction

In this module, we extract the features from the source code of login page and its respective home page. These features are structured into two categories.

- Basic URL features (R1-R5)
- Proposed features
  - Similarity based features (P1-P8)

- Hyperlink based features (P9-P10)

A. Basic URL features

These features are proposed in the existing literature (He et al. 2011; Mohammad et al. 2012; Xiang et al. 2011) which are extracted from the URL of the login page. They are as follows.

a) *R1: Number of dots in hostname :* This feature counts the number of dots in the hostname of the given URL. The rationale behind this feature is that attackers append the brand name of the target website to his phishing URL by the use of dots in hostname or pathname. For example, `http://www.iamphishurl.com/pathname/index.php` is a phishing URL targeting ebay then the attacker adds the ebay brand name in the hostname such that legitimate behavior is observed by the online user i.e. `http://www.signin.ebay.iamphishurl.com/pathname/index.php` The existing works such as Xiang et al. (2011) and He et al. (2011) used the count of dots in the entire URL but we observed that brand name importance is greater in the hostname rather than file path.

b) *R2: Presence of @ in URL :* This feature checks the presence of special symbol @ in the visited URL. If so then it is set to true (1) else set to false (0). Attacker inserts special symbol like @ in the phishing URL prefixed with legitimate address and postfixed with the actual phishing URL. The reason for using this is that when the browser encounters @ symbol in the URL it ignores the text prefixed before the @ and redirects to the URL specified after the @. Hence, attackers use this trick to deceive the user that he is visiting the legitimate URL. For example, when user clicks the URL `http://www.signin.ebay.com@www.iamphish.com/pathstructure/index.php` he is redirected to *iamphish* domain but not to the legitimate *ebay* website.

c) *R3: Use of HTTPS :* This feature checks the presence of HTTPS in the protocol of the visited URL. If HTTPS is present then it is set to true (1) else set to false (0). Use of HTTPS claims the legitimacy behavior in the websites whereas in phishing sites

use of HTTPS is rare unless the phishing site is hosted on compromised HTTPS domain.

d) *R4: Host length :* This feature calculates the number of characters in the hostname of the visited URL. It is observed that phishing URLs use lengthy URL such that actual brand or primary domain is hidden from the address bar.

e) *R5: Age of the domain :* This feature calculates the age of the domain of the given URL using the information extracted from WHOIS database. Usually, the life span of most phishing domains are less than 12 months as they are taken down at a faster rate (Mohammad et al. 2012).

f) *R6: Presence of hyphens and underscores in hostname :* This feature checks the presence of hyphens $(-)$ and underscores $(\_)$ in the hostname. If any of the symbol is present the feature is set to true (1) else false (0). Some attackers try to misguide the online user by substituting one or more letters of trusted website URL with the above symbols in creating phishing URL. These small changes may not be noticed by users, leading to visiting of phishing site. For example, an attacker might design a phishing site URL as `http://www.signin_ebay.com` for the legitimate site `http://www.signin.ebay.com`.

g) *R7: Base URL Length:* This feature is similar to *R4* feature but it calculates the number of characters in the Base URL of the visited URL. Base URL is calculated as protocol + hostname + pathname of given URL. It is observed that attackers either insert target website URL in the path or hostname of designed phishing URL such that online user would get deceived and directs to phishing URL. This behavior also attempts to hide the actual domain name of the URL from the address bar.

B. Proposed features

These features are further divided into similarity-based features and hyperlink-based features. The similarity-based features include P1 to P8 whereas hyperlink-based features include P9 and P10. The similarity-based features are extracted from the source code of both login page and home page of the given URL. Each similarity-based feature

is fed to Jaccard similarity coefficient module to calculate the similarity between login and home page. The hyperlink-based features are extracted from login page of given URL.

a) *P1: Filename Similarity :* This feature extracts the filenames from the login page and home page of the given URL. For the legitimate website, there exists at least some degree of reuse of filenames in both home page and login page. The filenames are extracted from various locations such as CSS, Javascript, Images, favicons and logos. For identifying the resource filenames, we extract all the links from `img[src]`, `a[href]`, `link[href]`, `script[src]` and then save the details. The same procedure is followed for the home page too.

b) *P2: Copyright Similarity :* This feature identifies similarity degree between the copyright text of respective login page and home page. The legitimate behavior is, maintenance of same copyright text across all the pages in the entire website unless some page is hosted by the attacker on the compromised domain.

c) *P3: Title Similarity :* We extract the titles from both login and home page anticipating that brand name exists in both the documents. But it is not mandatory to have the brand information in the titles of all the pages in the website.

d) *P4: Description similarity :* The websites use meta information to describe the brand information and the action involved in that respective page. Hence, we try to find out whether there is any similarity between home and login page with descriptions of each as sets of words. These sets of words are given as input to Jaccardian similarity module to identify the similarity score.

e) *P5: Maximum Frequency Domain similarity :* Most of the websites have greater number of local domains than foreign domains. The local domains are the pages which are containing same domain that of the hostname of the URL and the reverse is called foreign domain. The maximum anchor links pointing to a domain is considered as maximum frequency domain. Usually, for the legitimate sites, the maximum frequency domain is set as local domain. Phishers have anchor links

pointing to target legitimate sites which results in maximum frequency domain set to foreign domain. This differentiates the behavior of legitimate sites with phishing sites. Hence, we extract maximum frequency domain for both login and home page for calculating the similarity score.

f) *P6: TF-IDF terms similarity :* Term Frequency - Inverse Document Frequency (TF-IDF) gives the most important words related to the document. Term frequency defines the number of times the word is repeated in a document. Inverse document frequency defines the number of documents that contain the above term.

The repetition frequency of brand names or identifiers of the websites is very high in the original website and it is very less in other websites of corpus. Earlier works (He et al. 2011; Ramesh et al. 2014; Xiang et al. 2011; Zhang et al. 2007) used TF-IDF for identifying the brand names of the document with corpus of huge database of all documents. This consumes more time and storage while calculating the TF and IDF for each document. Hence, we came up with modified TF-IDF by skipping the database of all documents with either 6 or 7 documents as corpus. We considered brand locations such as plaintext, title, description, copyright, URL terms, header text, maximum frequency domain (if not local domain) as each document for identifying the brand names of the given document as shown in the Algorithm 4.1.

The brand name of a website is usually present in locations such as, title of the page, Header texts with h1 and h2 tags, copyright section, and URL (because it makes the user familiar with brand and URL map). The reason for choosing maximum frequency domain is that some phishing attacks include anchor links pointing to target legitimate site. In these cases, we can get the target website brand name.

In the existing works, top TF-IDF words are considered due to the fact that TF value of brand name in a website results in high count whereas IDF value of same brand term results in the low count (because of non relevance with other documents in the corpus). Combining both high TF value and low IDF values for the calculation of TF/IDF ratio results in high value. Hence, terms with top TF-IDF values are more relevant to the specified document. But, in our case we considered the documents

Algorithm 4.1: FindLeastTFIDFWords

Input : A HTML document *doc*, URL of website *url*
Output: List of 5 least TF-IDF words (w_i)

```
Initialize a String array brandLocations[ ]
host=getHost(url)
title=getTitle(doc)
plaintext=getPlaintext(doc)
desc=getDescription(doc)
cr=getCopyright(doc)
h1h2=getHeadersText(doc)
maxfreqdomain=getMaxFreqDomain(doc)
if maxfreqdomain==getDomain(url) then
    Add(host, title, plaintext, desc, cr,h1h2,maxfreqdomain) to brandLocations
     [ ]
else
    Add (host, title, plaintext, desc, cr,h1h2) to brandLocations [ ]
end
words=getLeastTFIDFwords(brandLocations [ ])
return words
```

of corpus as brand locations of the given document which results in very high IDF value for the brand name. Note that, the brand name with high TF value and high IDF value results in low TF-IDF value. Hence, we considered the terms with low TF-IDF value as most relevant terms for the specified document.

The same procedure is followed to identify the most relevant terms from the home page. The resulted two sets of words are checked for the similarity degree by using the Jaccardian index.

g) *P7: NER brand similarity:* Sometimes, brand names of websites may exist in locations other than brand locations (title, copyright, header text and description etc). Hence, we apply Named Entity Recognition (NER) method to identify the brand names located anywhere in the website. NER is a field of natural language processing that uses sentence structure to identify proper nouns and classify them into a given set of categories. We used Stanford's coreNLP[2], java library for implementing the Named Entity Recognizer module. The module takes input as extracted plain text from website and labels the sequence of words as Person, Organization or Location.

[2] https://stanfordnlp.github.io/CoreNLP/#download

From the resulting output, we extract the words which are labeled as Organization and are considered as brand names related to specified website. This NER module is run with both login page and home page with plain text as input to extract set of brand names present in each page. The members of two sets are compared with Jaccardian index to calculate the similarity score between both pages.

h) *P8: Bond status:* Usually, all the pages in a website contain the link back to the home page. If this behavior is present then the bond status is set to true (1) else it is set to false (0). Phishing sites normally do not have anchor link pointing to home page. Note that this absence of bond behavior can also be present in legitimate sites too.

The above all similarity features are sufficient for identifying the phishing sites which are hosted on compromised domains but fails when the same page is used for both login page and home page. Attackers maintain a single login page or an additional login successor page for phishing campaign such that their effort is reduced. They might include all anchor links pointing to the same login page or single common page such that foreign domains are avoided. This is countered in the hyperlink based features (P9-P10).

i) *P9: Common page detection ratio :* This feature is adapted from our previous work (Rao and Pais 2019a; Srinivasa Rao and Pais 2017). It identifies the ratio of frequency of common anchor links to the total number of anchor links. Many of the times phishing sites are designed with only single login page. Hence, the attackers define all the hyperlinks of the login page redirect to the same login page with # links such that online user is not diverted until sensitive information is given. Sometimes we also observed, attackers redirecting all the hyperlinks to a common page other than current login page. Hence, phishing sites contain maximum of common page ratio which includes either use of # links or single common page. On the other side, the legitimate sites, usually maintains different pages for different anchor links which includes less pattern matching ratio.

j) *P10: Zero links presence :* This feature checks for the presence of hyperlinks in the

body of the HTML of suspicious site. If present it is set to 1 else set to 0. This feature is explained in Chapter 3 as HF7 (refer page no 44).

Machine Learning classifiers used: To evaluate the performance of our feature set, we have used Support Vector Machine (SVM) (Burges 1998; Cortes and Vapnik 1995), Proximal Support Vector Machine (PSVM) (Fung and Mangasarian 2005) , and Twin Support Vector Machine(TWSVM) (Jayadeva et al. 2007, 2017) in training our model. We performed comparison of all the classifiers with a primary goal of choosing the best classification model among them. From the extensive experiments, we found that TWSVM performed the best among the other algorithms.

4.2.2 Experimentation and Results

Our method takes suspicious URL as input and extracts the source code from both suspicious URL and home page of the URL using Selenium WebDriver [3]. Selenium is a web browser automation tool which opens Firefox browser externally to perform browser like action as an automated process. Note that, home page of any website is considered as hostname of given input URL. Sometimes, visiting of hostname leads to Not Found pages, in that case, domain of the input URL is considered as home page. All the features from the source code are extracted using the Jsoup library. Jsoup [4] is a java library which is used to parse and manipulate the HTML code of a given website. Once the features and similarity scores between page *S* and page *H* are calculated then the scores are used to construct feature vectors. These vectors are given as input to the machine learning classifiers for classifying the suspicious sites as phishing or legitimate. All of the machine learning methods SVM, PSVM, and TWSVM have been simulated in MATLAB 12.0 environment on DELL PRECISION T1700 CPU with 16 GB of RAM. The MATLAB code of SVM and PSVM are downloaded from URLs `http://www.svms.org/software.html`, `http://svms.org/tutorials/Gunn1998.pdf` and `http://research.cs.wisc.edu/dmi/svm/psvm` respectively. We have used the RBF kernel in the SVM, PSVM and TWSVM for the prediction of the phishing websites.

[3] http://docs.seleniumhq.org/download/
[4] https://jsoup.org/

Table 4.2: Tuning parameters for the experiments

Learning Algorithms	Parameters
TWSVM	P=0.1 and C=0.4
PSVM	P=0.3 and v=selftuned
SVM	P=2 and C=2

Table 4.3: Source of websites

Source	Total instances	Link
Phishing sites	1500	https://www.phishtank.com/ developer info.php
Legitimate sites	1500	http://www.alexa.com/topsites

A. Parameter tuning

We have tuned the parameters of all machine learning methods using Exhaustive search method (Rao and Mitra 1972). The value of the RBF kernel parameter P and C of SVM and PSVM has been searched in the set $\{2^i : i = -10, -9, ..., 9, 10\}$. For the TWSVM, we have taken $c_1 = c_2$ for the sake of convenience of experiments and their optimal values have been searched in the set $\{0.1, 0.2, ..., 1\}$. The final selected parameters are given in Table 4.2.

B. Dataset

We collected dataset from two sources as given in the Table 4.3. The phishing sites are collected from phishtank website and legitimate sites are collected from list of websites maintained by Alexa. Note that we have collected the 1500 legitimate websites randomly from the Alexa database such that bias is prevented towards high ranked websites unlike existing works (Dunlop et al. 2010; Huh and Kim 2011; Rao and Pais 2019a; Xiang et al. 2011; Xiang and Hong 2009). One of the limitation of existing works is that they considered top ranked legitimate sites and page rank as one of the attribute for the classification of legitimate and phishing. This results in bias towards high ranked benign sites which obviously increases the accuracy. But this is not a fair dataset for identifying the performance of our models. Hence, we have randomly chosen legitimate sites from the list of top 1 lakh websites from Alexa.

We have also evaluated our model by conducting various experiments with different feature sets and machine learning algorithms. We have included SVM and other versions of SVMs (Jayadeva et al. 2007, 2017) for the classification of suspicious sites.

We divided our features into 4 sets namely

- *FS1*- list of basic URL based features (R1-R7).
- *FS2*- list of proposed features including both similarity and hyperlink based features (P1-P10).
- *FS2′*- list of proposed features including only similarity based features (P1-P8).
- *FS3*- list of all features combining both FS1 and FS2 (R1-R7,P1-P10).

C. Experimental evaluation

We conducted various experiments with different feature sets and different machine learning algorithms for assessing the importance of the proposed features. All the experiments are conducted with the same dataset of 3000 instances (1500 legitimate, 1500 phishing) and 10-fold cross validation is used for each. In the Experiment-1, we have evaluated our model with all the features (FS3) using various classifiers. This experiment also results in identifying the best classifier suitable for our dataset. In Experiment-2, we evaluated different feature sets FS1, FS2, and FS3 for identifying the importance of each set in detecting the phishing sites. In the Experiment-3, we evaluated similarity based features FS2′ for identifying the effectiveness of detecting phishing sites hosted on compromised servers (PSHCS). This experiment reveals the richness of FS2′ in detecting PSHCS. Note that, we investigated the dataset of all phishing sites (1500) manually to identify the PSHCS. We observed 1170 PSHCS out of 1500 which contributes to 78% of total phishing sites. Finally, in the Experiment-4, we evaluated our model with each individual feature to identify the importance of each of them in detecting phishing websites.

Experiment-1: Evaluation of FS3 with various classifiers

In this experiment, we evaluated our feature set FS3 with SVM and observed an accuracy of 98.03% with 98.13% Recall and 97.94% Specificity. We also tested our features

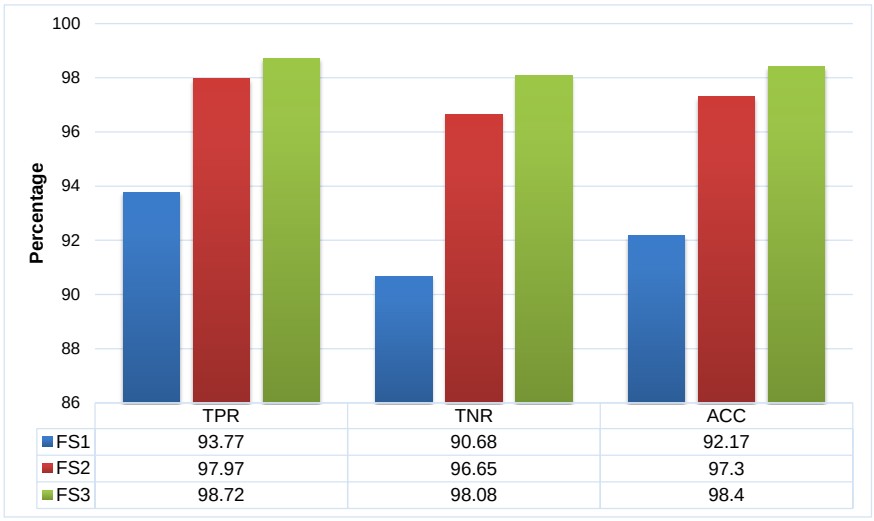
Table 4.4: Results of Experiment-1

Metrics	TWSVM	SVM	PSVM
Recall	98.72	98.13	97.74
Specificity	98.08	97.94	98.06
Precision	98.07	97.93	98.07
F Score	98.39	98.03	97.90
Accuracy	98.40	98.03	97.90

with other versions of SVM such as TWSVM, and PSVM to identify the best classifier for the detection of phishing sites in our dataset. From the results shown in Table 4.4, it is observed that all the applied machine learning algorithms achieved an accuracy of more than 97% which indicates the richness of our proposed features in classifying the phishing websites.

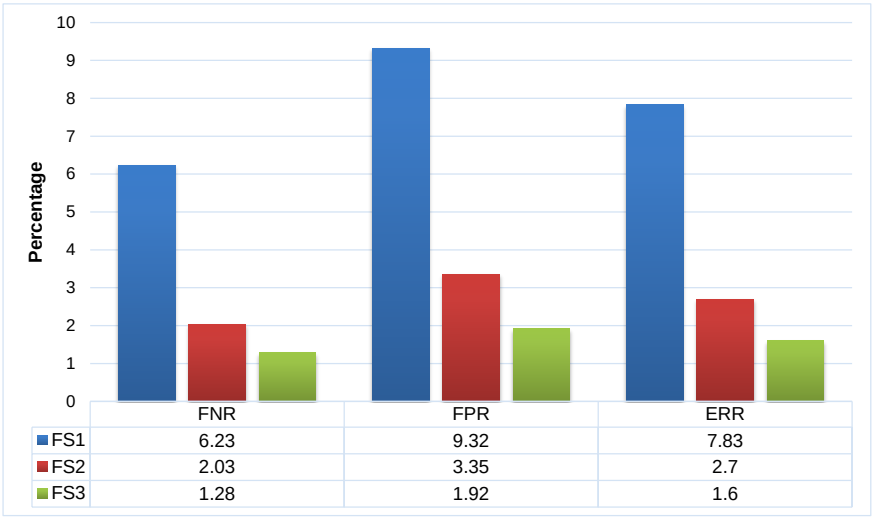
Experiment-2: Evaluation of FS1, FS2, FS3 with Twin Support Vector Machine

In this experiment, we evaluate the basic URL features FS1, proposed features FS2 and combined features FS3 by calculating the detection rate and error rate with above mentioned metrics as shown in Figure 4.4. Note that, the experiments with FS1, FS2 and FS3 are carried out with TWSVM classifier as it performed the best in detecting the phishing sites as discussed in Experiment-1. Firstly, we tested our model with FS1, and it resulted in TNR value of 90.68% and TPR value of 93.77% which shows the richness of selected existing features but the accuracy of 92.17% indicates that FS1 alone is not sufficient to counter the phishing sites. Later, we tested our model with feature set FS2, to evaluate the performance of proposed features. The results of the experiment demonstrate an increase in TPR to 97.97%, TNR to 96.65% and an accuracy to 97.3%. The significant metric values with FS2 dataset indicates the goodness of our proposed features. Finally, we tested our model with FS3 which resulted in TPR, TNR and Accuracy of 98.72%, 98.08% and 98.40% respectively. The significant increase of metric values indicate that addition of FS1 to FS2 complements the accuracy in detecting phishing sites. All the experimental results with feature sets can be seen in Figure 4.4 where 4.4 a) gives detection accuracy and 4.4 b) gives error detection of the system.



	TPR	TNR	ACC
FS1	93.77	90.68	92.17
FS2	97.97	96.65	97.3
FS3	98.72	98.08	98.4

(a) Detection Accuracy



	FNR	FPR	ERR
FS1	6.23	9.32	7.83
FS2	2.03	3.35	2.7
FS3	1.28	1.92	1.6

(b) Detection Error

Figure 4.4: Performance of our system with different feature sets

Experiment-3: Evaluation of FS2' with Twin Support Vector Machine

In this experiment, we evaluate only similarity based features which are used for detecting phishing sites hosted on compromised servers. For testing the performance of our model in detecting phishing sites hosted on compromised servers, we considered a dataset of 1500 legitimate sites and 1170 phishing sites which are hosted on compromised servers. Note that, we identified 1170 out of 1500 phishing sites really hosted on compromised servers. The experiment results in Table 4.5 reveal that percentage of correctly detected phishing sites reached to 99.66% which shows the importance of similarity based features in detecting the PSHCS. Our model could achieve an accuracy

Table 4.5: Evaluation of similarity based features

Classification output	Value
# of legitimate sites	1500
# of PSHCS	1170
Recall	99.66%
False Positive Rate	5.07%
Precision	93.43%
F measure	96.44%
Accuracy	96.78%

of 96.78% , precision of 93.43% and F measure of 96.44% which illustrate that addition of these features to other category of features complement the information and thereby improves the accuracy of the overall system.

Experiment-4 Performance of system with individual feature

The compromised behavior based features are very hard to be evaded by attackers because they cannot manipulate the content of home page of compromised domain; if so, the website owner will get a suspicion on it and might get to know that domain is compromised. In the other way, if an attacker tries to modify the content of his phishing page such that similarity between home and login page would be increased then the online user who is exposed to phishing site might get a suspicion on the differences in fake site and may leave the phishing page. In this section, we also discuss about the importance of individual feature in detecting phishing sites. The accuracies of our model evaluated with each feature is given in Figure 4.5. The feature P1 (Filename Similarity) has achieved an accuracy of 92.97% which indicates that the reuse of external resources like images, scripts, styles are most common in legitimate sites and less appealing in phishing sites in compromised domains. The remaining percentage of error is due to the websites which are registered maliciously or due to the poor designing of legitimate sites.

The features like P2 (Copyright), P3 (Title) and P6 (TF-IDF) have crossed an accuracy of above 80%. If we observe these three features, all of them are text-based features which considers textual content of the website for extracting unique descriptors of

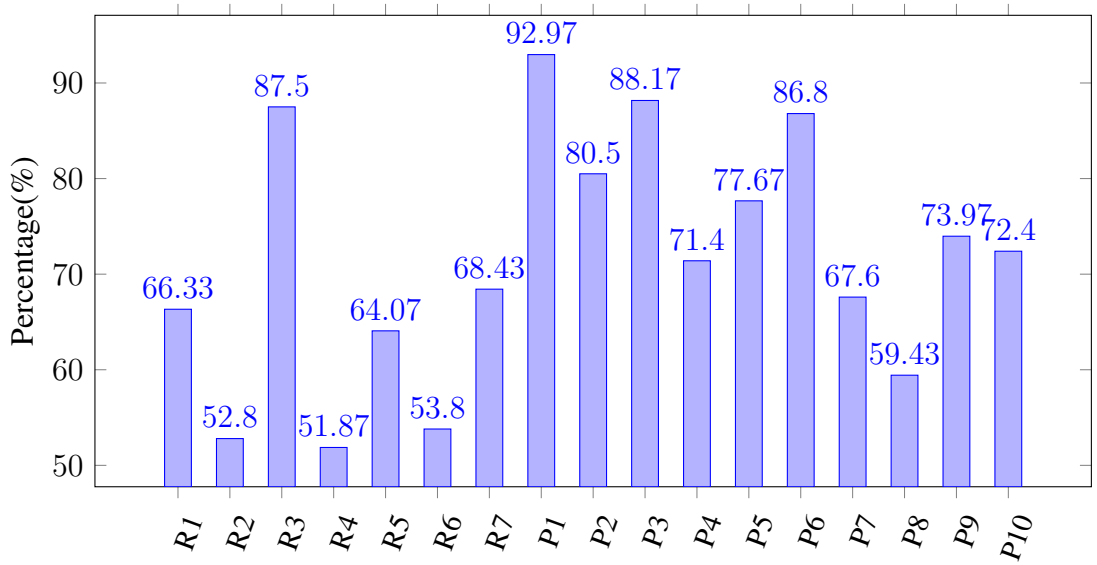


Figure 4.5: Classification accuracy of individual features

the website. Note that, feature P4 (description) is also textual based feature which has achieved an acceptable accuracy of 71.4%. The significant accuracy of these features indicates that textual-based features play a very vital role in detecting phishing sites in PSHCS. From the Figure 4.5, it is observed that most of the proposed features outperformed basic URL features with over 70% accuracy including P1-P6 and P9-P10. Out of all URL features, only R3 (HTTPS) could achieve an accuracy of above 85% which indicates the low usage of HTTPS phishing sites and also low number of PSHCS with HTTPS protocol. Hyperlink-based features P9-P10 also achieved an accuracy of more than 70% which indicates the strength of features in detecting the malicious registered phishing sites. It should be noted that the statistics are dependent on the quality and quantity of the training data used for the classification of phishing websites.

Experiment-5 Comparison of our work with existing works

In this experiment, we implemented existing benchmarked anti-phishing approaches such as Varshney et al. (2016b), Jain and Gupta (2018) for the comparison with our work based on the performance metrics of Accuracy, Sensitivity and Specificity. We used the same dataset used in Experiment 1 for the comparison of existing works. The main reason for choosing these techniques over other works is due to their database independence. Similar to our work, both the existing works also does not need initial database of images or DOMs or styles or URLs for phishing detection. To ease the presentation, we term the Jain & Gupta work as EW1 and Varshney et al. work as EW2 and the experimental results are given in Figure 4.6. The results demonstrates that our technique outperforms the existing works EW1 and EW2 with an accuracy of 98.4%. EW1 outperformed our work with respect to specificity but it has achieved the least sensitivity over the other works due to the use of hyperlink based features at the second level of filtering. There is a trade-off between sensitivity and specificity so a good anti-phishing system should provide balanced sensitivity and specificity. From the Figure 4.6, it is evident that our work achieved balanced sensitivity and specificity with 98.72% and 98.08% respectively.

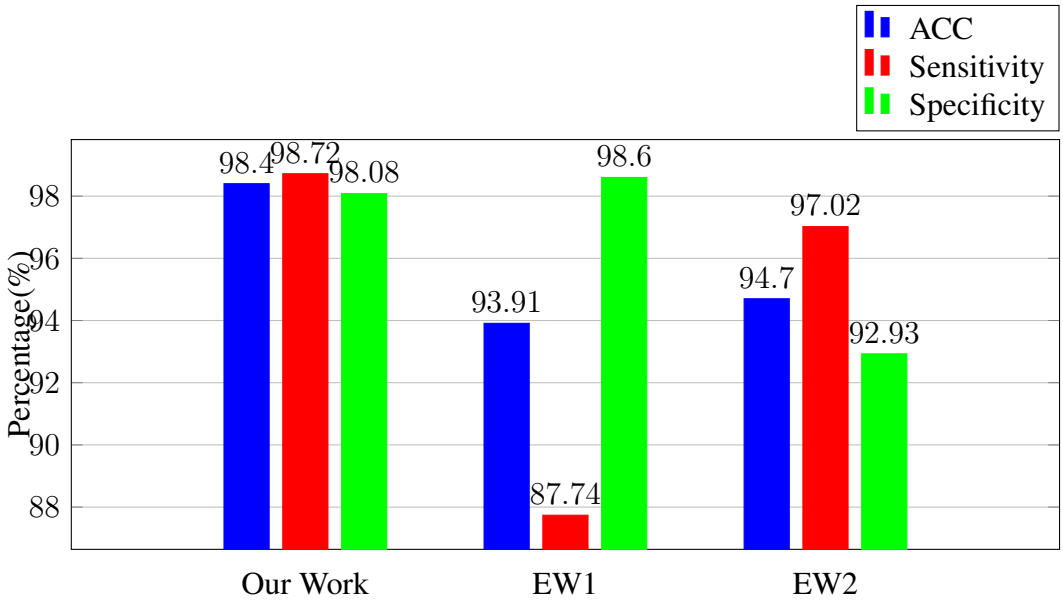


Figure 4.6: Comparison of our work with existing works EW1 and EW2

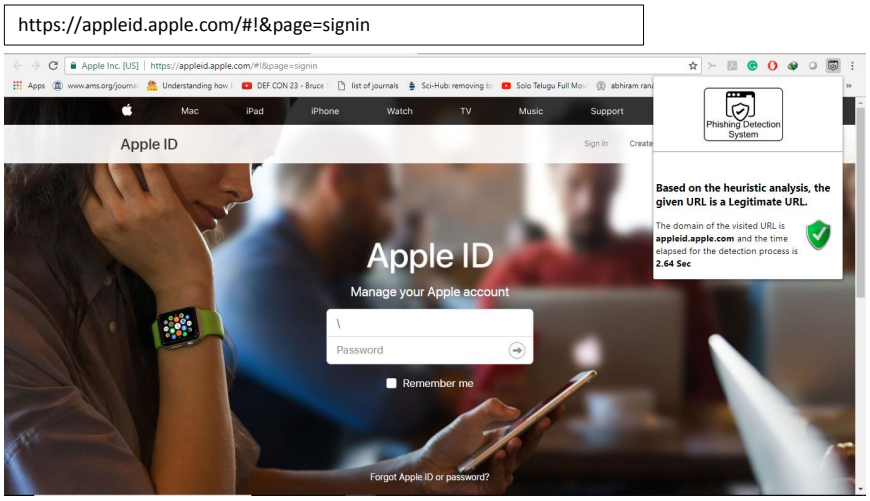
4.2.3 Discussion

A. Design of Chrome Extension Our goal in this work is to provide real time protection from website phishing attacks. Hence, we built a chrome extension which predicts the visited URL as a phishing or legitimate. The extension is designed such that no extra clicks or key press are required i.e. on single click of extension in Chrome browser, the extension displays the status of the website as legitimate or phishing. The Chrome extension is written in Javascript which extracts URL, Title and DOM of visited website from the browser and makes a connection to REST API running at the remote server where the actual execution of technique takes place. The REST API is hosted on an Intel Xeon 16 core Ubuntu server with 2.67GHz processor and 16GB RAM.

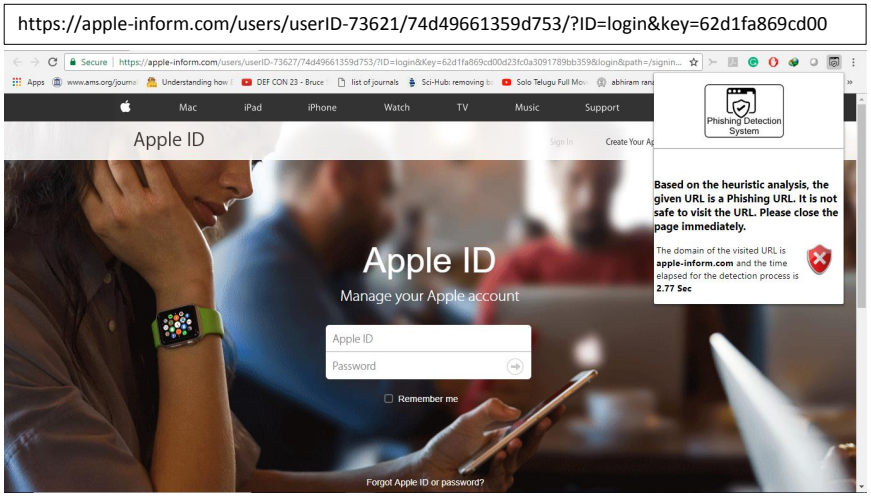
The REST API feeds the above values to our application running at the remote server and thus proceeds with extraction of features (Section 4.2.1) for the phishing detection. Note that, the REST API is implemented with Spring framework and POST method is used for transferring the DOM, whereas GET method is used for transferring the URL and Title to the server. Once the server receives these values it classifies the status of website as phishing or legitimate based on the outputs of modules discussed in Section 4.2.1.

Since, our goal is to provide real time phishing detection, the time for feature extraction and classification has to be very less. Hence, we parallelized the process of feature extraction for the proposed features with the help of multiple cores present in the system. Once, the classification is done the application returns the status to REST API which is further sent to Extension as a response.

This whole process of execution took an average time of 3.19 seconds. However, this time is dependent on various factors such as website hosted on the server, speed of the Internet, and speed of the system. In our case, we used a bandwidth of 100mbps, as mentioned earlier, system with Intel Xeon 16 core Ubuntu server with 2.67GHz processor and 16GB RAM. The output of the extension is shown as popup window containing the status of the website as shown in Figure 4.7. In image (a) of Figure 4.7, our extension detects the Apple website as legitimate with a response time of 2.64 sec



(a) Legitimate site detection



(b) Phishing site detection

Figure 4.7: Output of our extension

whereas image (b) shows the detection of phishing site with a response time of 2.77 sec in a popup window containing the status of website and textual content recommending the user to close the webpage immediately.

B. Effectiveness of presented model// Unlike list based or visual similarity based approaches, our technique is independent of the earlier storage of images, DOMS, Styles and URLs. Hence, when we extract homepage from the visited URL it sometimes results in false positives.

In our experiment 1, we observed a false positive rate of 1.92% due to the non-availability of home page for the visited URL. This can however be reduced by identify-

ing the home page with search results (SR) queried with search query=primary domain. The process of identifying the homepage with SR includes additional computation cost but increases the accuracy. We intend to attempt this procedure of identifying the home page in the future work.

Similarly, our model also included FNR of 1.28% in Experiment 1. This is due to the fact that phishing sites which are maliciously registered has their own homepage and login page or both represent the same page. This made our technique fail to detect such phishing pages. However, if we observe the Experiment 3 results, our model could detect 99.66% of PSHCS. Hence our model is more suitable for the detection of PSHCS.

If the attacker copies the contents of compromised site (code obfuscation) to his/her designed phishing page then the technique may be bypassed. But such a scenario is very rare because of the following cases.

- Attacker has to make an extra effort in making changes to the downloaded phishing kit.
- If he adds the content of compromised site into designed phishing site, then the phishing site might show some suspicious behavior to the online user.
- From the experimental analysis, we did not observe any single phishing instance copying the contents of compromised server to the designed phishing page.

The main focus of the work is to detect phishing sites which are hosted on compromised servers as they contribute to more than 70% of phishing websites (Moore and Clayton 2007). Hence, our presented work is designed to detect such phishing sites along with malicious registered phishing sites without the dependency on an initial database of resources (CSS, Images, DOM, URLs) or third-party services. With the use of TWSVM classifier, our model is able to achieve a significant accuracy of 98.4%.

Despite the *Curb-Phish* achieved significant detection rate, it has certain limitations. Firstly, the model may fail when home page of visited site is unavailable which might result in false positives. Due to this behavior, the model achieved a FPR of 1.92% which can be seen Experiment 1. Secondly, even though the model detected 99.66% of PSHCS

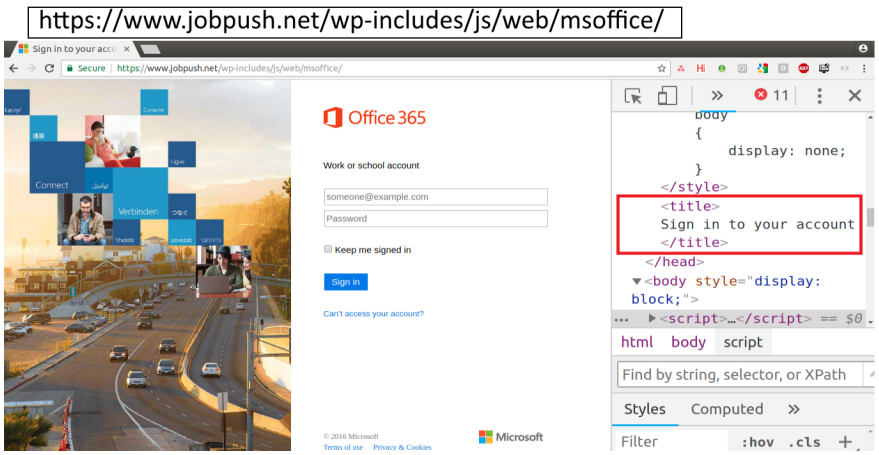
but it still resulted FNR of 1.28% in Experiment 1. This is due to the fact that phishing sites which are maliciously registered has their own homepage and login page or both represent the same page. Hence, we present a new method (Jail-Phish) to identify the related page of suspicious site using the search engine results with dynamic search query in the next section such that the technique detects both PSHCS and malicious registered sites.

4.3 JAIL-PHISH : AN IMPROVED SEARCH ENGINE BASED TECHNIQUE TO DETECT PSHCS

There exist many heuristic based techniques which rely on source code, URL, machine learning and third party services for the detection of phishing sites. Out of these, third party based features have played a vital role in achieving high true negative rate as the phishing sites are either less ranked or indexed by the third party services.

The third party services include the use of search engine, page ranking, WHOIS, etc. Many of the recent works (Chiew et al. 2015; Dunlop et al. 2010; Huh and Kim 2011; Jain and Gupta 2018; Tan et al. 2016; Varshney et al. 2016a,b; Xiang and Hong 2009) focused on using search engine results for the detection of phishing sites. The limitation of search engine based techniques is that they fail to detect phishing sites which are hosted on compromised servers. In this case, the phishing pages are termed as legitimate sites leading to high false negative rate. On the other side, the newly registered legitimate sites are classified as phishing sites due to its absence in search results leading to high false positive rate.

The working of the search engine based technique is as follows. Firstly, a query string is either generated with the unique key descriptors of suspicious website (Xiang et al. 2011; Xiang and Hong 2009; Zhang et al. 2007) or domain concatenated with title (Huh and Kim 2011; Jain and Gupta 2018; Varshney et al. 2016b) or an image (Chang et al. 2013; Chiew et al. 2015, 2018). The query string is fed to the search engine to output the relevant results containing anchor links with respect to the query. If the domain of input URL is present in the search results, it is classified as legitimate else classified as phishing. The search engine results when queried with title and domain is

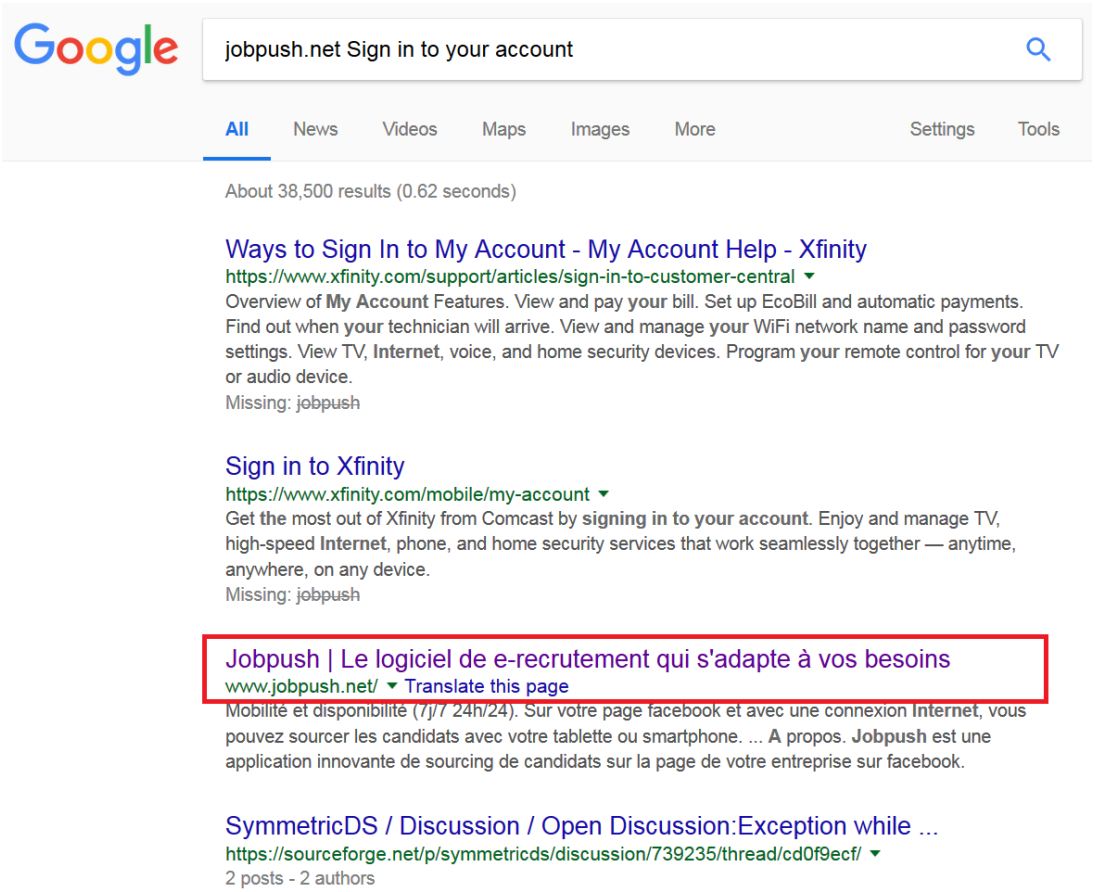


(a) Designed phishing site with its source code

(b) Compromised website

Figure 4.8: Phishing site hosted on compromised server

shown in Figure 4.8. Figure 4.8 (a) shows the designed phishing site with its source code highlighted with title. Note that the phishing site is hosted on the compromised server and it is shown in Figure 4.8(b). Feeding the query string to search engine, the domain of phishing page is returned in search results which classifies the input phishing URL as legitimate, as shown in (c). On the other side, newly registered or high Alexa ranked legitimate site is absent in the search results when queried with title and domain.(shown



(c) Search results when queried with domain + title

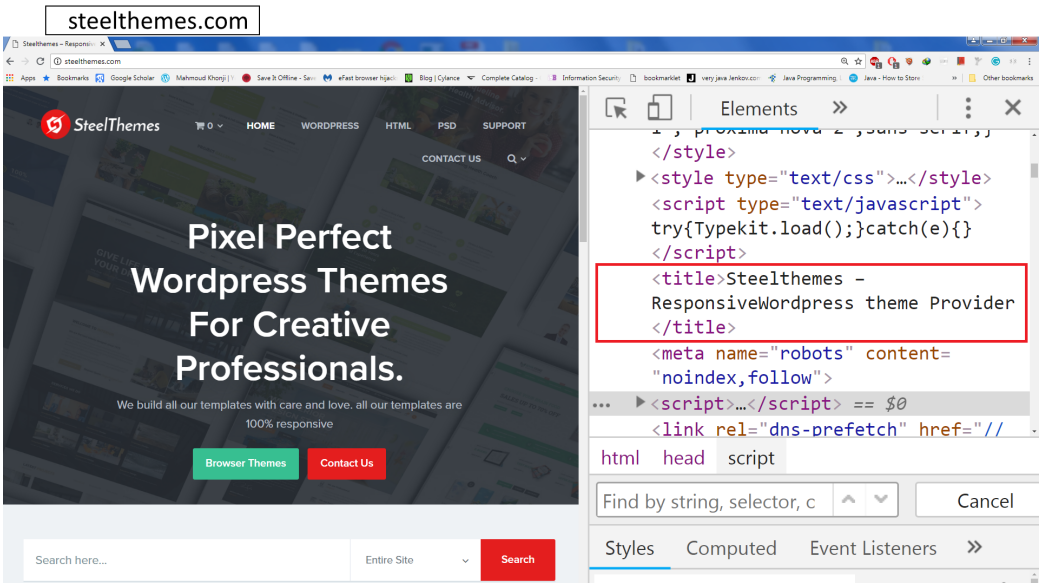
Figure 4.8: Phishing site hosted on compromised server

in Figure 4.9). Figure 4.9(a) shows the legitimate site and its source code with title highlighted and 4.9 (b) shows the absence of domain in the search results which leads to classification of legitimate as phishing.

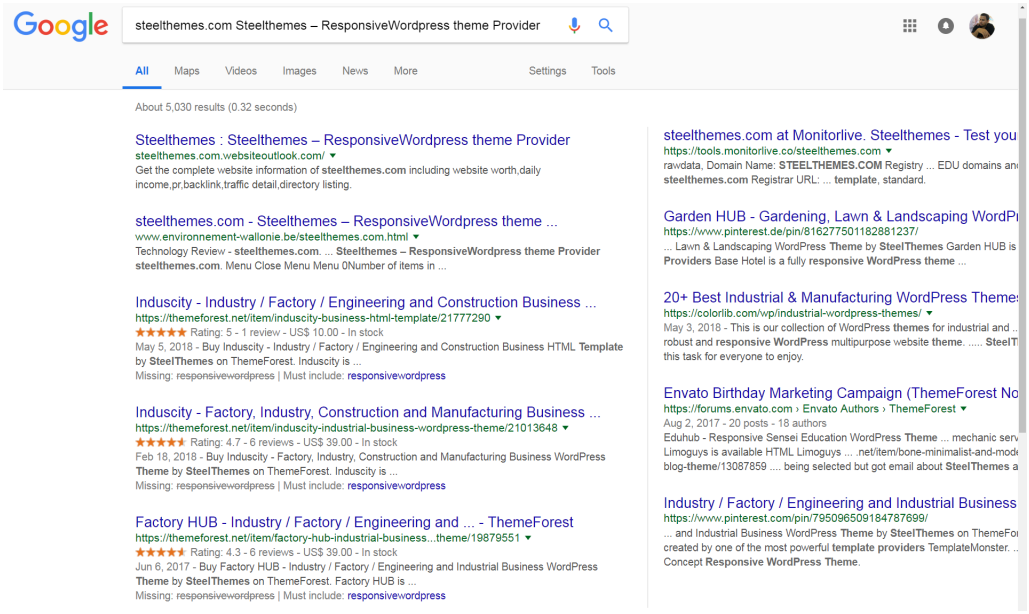
In this section, we address the gaps found in the search engine based techniques and some of the gaps are as follows.

- The search engine based techniques (Ramesh et al. 2014; Tan et al. 2016; Xiang et al. 2011; Zhang et al. 2007) which use identity keywords extracted from copyright, plaintext, header texts fail in extracting relevant terms when the text in phishing site is replaced with an image.

- Many search engine based techniques (Jain and Gupta 2018; Nguyen et al. 2014; Rao and Pais 2019a; Varshney et al. 2016b) fail when the phishing sites hosted on



(a) Legitimate site with its source code



(b) Search results when queried with `domain + title`

Figure 4.9: High Alexa ranked legitimate site and its absence in search results when queried with `domain + title`

legitimate web server are encountered. It is due to the fact that these legitimate domains have been running for a long time and hence there is more chance that they are indexed by the search engine.

- Some benign sites are not displayed in the search results when queried with `domain + title` or website identity keywords. Based on this behavior, the

existing techniques (Jain and Gupta 2018; Varshney et al. 2016b) classify legitimate as phishing.

We address these gaps with a dynamic search query such that the legitimate sites with no title or vague title or repetitive brand in the search query are handled to return query-domain in the search results. This dynamic search query also gives the advantage of returning high Alexa ranked legitimate domain in the search results. We also include the similarity based mechanism to counter the phishing sites which are hosted on compromised servers. The contributions of our work are summarized as follows.

- Proposed a dynamic search query string generation for returning the relevant search results.
- Proposed a similarity based mechanism to detect the phishing sites hosted on compromised servers (PSHCS).
- Proposed technique is robust to different URL masquerade techniques and can also detect zero-day phishing attacks.
- Proposed technique detects non-English phishing websites and also, it does not depend on prior access to the database of target website resources or web history.
- Comprehensive evaluation of our work with fresh and old phishing data shows that Jail-Phish is adaptable for detection of long-lived phishing sites.

4.3.1 Methodology

The working of *Jail-Phish* is given in Algorithm 4.2. and the architecture is given in Figure 4.10. The basic idea of our work is simple but effective in detecting phishing sites which is explained as follows.

Our tool is implemented in Java which takes Webpage (V) as input and extracts the title, domain and source code from the website as shown in Figure 4.10. The extracted domain and title are appended to generate a search query. On feeding the search query, the top n search results from the search engine results page (SERP) are considered for checking the query domain in the search results. If the query domain is present

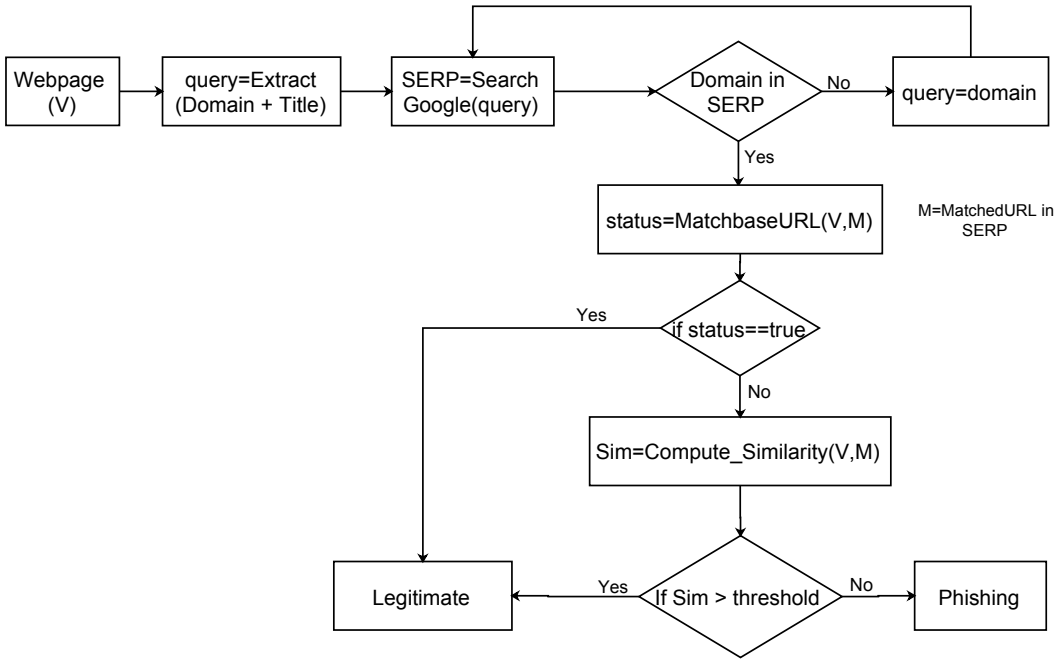


Figure 4.10: Architecture of Jail-Phish

in search results then the similarity between the content of query URL and matched domain in SERP are calculated. If the similarity is above a threshold then it is classified as legitimate else it is PSHCS.

The other possibility is that the absence of query domain in search results and is eventually classified as a phishing site. This assumption also leads to incorrect classification of newly registered or non-popular legitimate sites as phishing but due to the usage of dynamic search query the false positive rate (legitimate to phishing) is reduced to a major extent. Hence, the query is set to only domain when the domain is absent in SERP to get the new SERP. If the domain is absent in the new SERP then the webpage V is classified as phishing else similarity between the matched result and webpage V is calculated for identifying the status of the webpage. The Jail-Phish work is briefly explained with different stages involved in the detection process in next section.

Stages in Jail-Phish

We divide our work into 6 stages and are given as follows

Stage 1: Extraction of Domain and Title– In this module, title and domain are

Algorithm 4.2: Algorithm of Jail-Phish

Input: URL of the webpage

Output: Status of the webpage

```
1: Legit_Status ← False, Domain_Status ← False
2: Query_Domain ← Extract_Domain(URL)
3: Parse the Source code extracted from URL
4: Query_URL_DOM ← getDOM(URL)
5: Query_baseURL ← getbaseURL(URL)
6: Query_Title ← Extract_title(Source_Code)
7: if Query_Title! = null then
8:     Search_query ← Query_Domain + Query_Title
9:     Top_10_Search_results[] ← Perform_search(Search_query)
10:    ∀n ∈ Top_10_Search_results
11:    Domain_i ← Extract_Domain(n)
12:    baseURL_i ← getbaseURL(n)
13:    for i ← 1 to 10 do
14:        if Query_Domain == Domain_i then
15:            Domain_Status ← True
16:            if Query_baseURL == baseURL_i then
17:                Announce Legitimate
18:            end if
19:            Matched_Domain_DOM ← getDOM(n) /* where n is the URL of
               matched domain in search results                          */
20:        end if
21:    end for
22:    if Domain_Status == False then
23:        Search_Query = Query_Domain
24:    else
25:        goto step 40
26:    end if
27: else
28:    Search_Query ← Query_Domain
29: end if
30: Top_10_Search_results[] ← Perform_search(Search_query)
31: ∀n ∈ Top_10_Search_results
32: Domain_i ← Extract_Domain(n)
33: for i ← 1 to 10 do
34:    if Query_Domain == Domain_i then
35:        Domain_Status ← True
36:        Matched_Domain_DOM ← getDOM(n)
37:    end if
38: end for
39: if Domain_Status == True then
40:    Similarity_Score ← Compute_Similarity(Matched_Domain_DOM,
       Query_URL_DOM)
41:    if Similaity_Score > Threshold then
42:        Legit_Status ← True
43:    else
44:        Legit_Status ← False
45:    end if
46:    if Legit_Status == True then
47:        Announce Legitimate
48:    else
49:        Announce Phishing
50:    end if
51: else
52:    Announce Phishing
53: end if
```

extracted from the given suspicious URL. We have used Selenium WebDriver[5] to visit the given URL and thereby the source code of the suspicious URL is extracted by the same WebDriver. Jsoup library is used to parse the source code to extract the title and other relevant DOM elements. Jsoup [6] is a java library which is used to parse and manipulate the HTML code of a given website. The reason behind choosing the Jsoup library is due to its availability of API which provides functionality to select elements of DOM and also it handles non-well formed HTML very effectively. We have also used Google Guava library for the extraction of Domain from the URL. The baseURL is generated by appending the hostname and pathname of the given URL which is further used for the classification of websites.

Stage 2: Search Query String Preparation– This module provides dynamic search query string for different kinds of websites. As discussed in the previous section, we choose `domain + title` as a search query for performing the searching mechanism. Generally, domain gives brand name of the respective website and title gives a brief description about the website to online users and search engines. Based on the experimentation, we observed some issues pertaining to search query for the Google search engine.

Overall, we conclude that title is a must in the search query for some set of legitimate sites and should be skipped for other set of legitimate sites. Hence, we consider the search query as either `domain + title` or `domain` to obtain the results.

But the limitation of choosing only domain is that, phishing sites hosted on compromised servers will also be returned in search results which leads to high false negative rate (phishing to legitimate). Also, domain with regular keywords (sign in, log in, login, untitled, home) as titles in PSHCS results in false negative rate (FNR). This limitation is countered by the similarity based mechanism which is discussed in **Stage 5**.

Initially `domain + title` is considered as search query. This search query is modified at two instances. Firstly, if the domains of returned search results do not match with query domain then search query is changed to only query domain.

[5]http://docs.seleniumhq.org/download/
[6]https://jsoup.org/

Secondly, if there exists no title in the source code of the given URL then also search query is changed to only query domain.

Stage 3: Search Processing– This stage feeds the assembled search query to search engine for returning the relevant search results. Google is used as a search engine for our experimentation. Our technique is adaptable to any search engine such as Bing, Yahoo or Baidu but based on the prior study by Huh and Kim (2011); Jain and Gupta (2018); Varshney et al. (2016b), we have chosen Google for the experimentation. Based on the query string, the number of returned results would be changing. From the experimental results, we observed that the optimal value of search results to be considered is T = 10. (shown in Figure 4.11). We have considered 1500 legitimate sites randomly chosen from our legitimate dataset (LD1-LD3 as given in Table 4.6) and 1500 phishing sites from PhishTank as dataset for the experimentation. To find the optimal T, we considered various T values ranging from 1 to 15 and search query = `domain + title` to calculate the TNR on the dataset. It is observed that the Jail-Phish has achieved maximum TNR of 97.87% at threshold T=10. We also conducted experiment on phishing sites for calculating optimal T with greater TPR. From the results, it is observed that TPR of 99.67% was achieved at all values of T. Hence, we chose the threshold of search results as 10. It should be noted that small value of threshold may miss the matching domain which might lead to high false positive rate and large value of threshold may result in unnecessary matching of search results thus leading to increase in computation cost. Majority of the existing techniques (Jain and Gupta 2018; Ramesh et al. 2014; Varshney et al. 2016b) used less than or equal to 10 search results. Due to the shutdown of Google global site search,[7] we have automated the google searching process using Selenium WebDriver and the search results are parsed using Jsoup.

Stage 4: Initial Decision Making– On feeding the query string to Google, if the query domain is returned in the top 10 search results then domain status is changed to true which indicates that the query domain is either a pure legitimate page or a PSHCS. To identify the status of pure legitimacy, the baseURL of the matched domain is compared with baseURL of the query domain. If matched then the queried domain of the URL is classified as legitimate and execution of further steps are skipped else query

[7] https://enterprise.google.com/search/products/gss.html

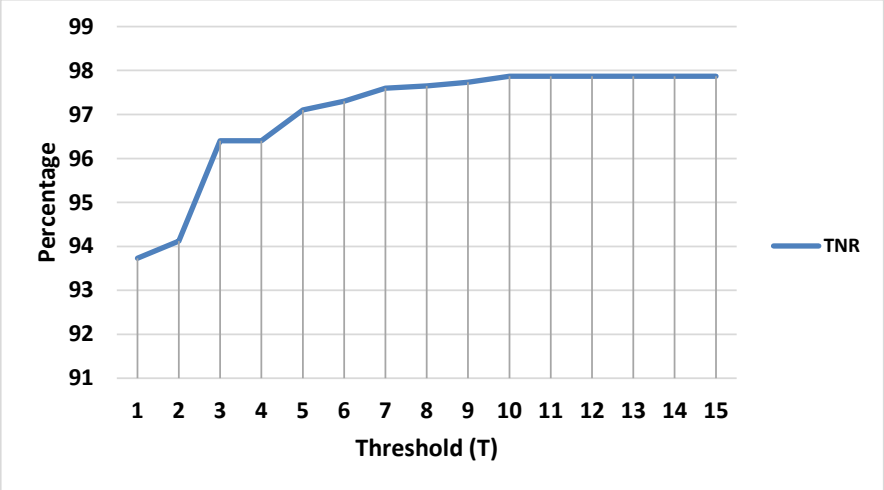


Figure 4.11: Comparison of TNR at different Threshold

domain undergoes next filtering. As most of the popular legitimate sites are better indexed in Google, they fall into the above category. To differentiate between legitimate page and PSHCS, we perform similarity computation between the matched domain and given suspicious page as explained in **Stage 5**. There exists some query domains which matches the primary domains in search results but differs in country code top level domain (ccTLD) or global top level domain (gTLD). These kind of websites also undergoes similarity computation stage. Note that, websites with same domain but different ccTLD or gTLD does not guarantee that both of them belong to same organization or company. The combination of primary domain and gTLD or ccTLD or both contribute to domain of the URL. For example, in URL `https://www.regist.login.nextpm.com.ru/web/log/secure/index.php`, we have, protocol as https, subdomain as regist.login, primary domain as nextpm, gTLD as com, ccTLD as ru, Domain as nextpm.com.ru, Hostname as regist.login.nextpm.com.ru and pathname as web/log/secure/index.php

If the query domain is not matched with any domains of the returned results then query string is reset with only domain to perform search processing (Stage 3). Even then, if the query domain is absent in returned search results then it is classified as phishing else the matched domain URL undergoes further stages of filtering.

Stage 5: Similarity Computation- This module checks whether the given URL is

a phishing page hosted on compromised server or not. Usually, there exists a level of similarity between the pages within the website. This hypothesis is used to detect the PSHCS. There exist some resources like Javascript, CSS, images, logos common to all the pages within the website. Hence, we extract these resource files from the suspicious website *S* and also matched domain *M* in search results to calculate the similarity between them. The similarity computation mechanism is given in algorithm 4.3 and the similarity based features are given as below.

F1: URL - We extract all the URLs which are local corresponding to *S* and *M* from their respective source codes. The local anchorlinks are extracted from the inspection code with `a[href]` and are saved in different sets for each page.

F2: Styles - We extract all the local styles corresponding to M and S from the inspection code with `link[href]`. Note that, these styles files contain `CSS` extensions in the link URL. These style files describes the visual look of the HTML elements displayed in the browser.

F3: Javascript - We extract all the javascript files embedded in both the pages and are stored in different sets. These files have `js` extension and are extracted from inspection of `src` attribute of script tag i.e `script[src]`. These files provide dynamic actions to the website.

F4: Images - We extract all the images corresponding to both pages from the inspection of `src` attribute of `img` tag i.e `img[src]`. These files can have various extension such as png, ico, jpeg, jpg etc. The pages within the website might mostly involve some of the common images like logos or favicons.

Once all these resource URLs from JS, CSS, images, anchorlinks are extracted, they are combined to form two large sets corresponding to *M* and *S*. These two sets are fed as input to Jaccardian similarity module to calculate the similarity between the pages.

Stage 6: Final Decision Making - In this module, the final decision on the status of the given URL is taken based on the similarity score. If the similarity score is above the threshold then it is classified as legitimate. For simplicity, we have taken threshold as 0 i.e. if the similarity score is above 0 indicating that there exists some level of similarity then it is classified as legitimate else it is classified as a phishing site hosted on compro-

Algorithm 4.3: Similarity computation algorithm of Jail-Phish

Input: Matched_Domain_DOM, Query_URL_DOM
Output: Similarity_Score

1: Feed the Domain_DOM to Jsoup API for parsing the source code.
2: Extract the similarity based features (URL, Styles, Javascript, Images) required for the detection of phishing website hosted on compromised server.
a) Save all the features in an ArrayList where each feature consists of set of words.
3: Feed the Query_URL_DOM to Jsoup API for parsing the source code.
4: repeat step 2
5: Combine all the resources with respect to each page i.e. Matched page *M* and suspicious page *S* in two different sets
6: Calculate the Jaccard similarity coefficient for *M* and *S*.

$$J(M,S) = \frac{|M \cap S|}{|M| + |S| - |M \cap S|} \tag{4.2}$$

where J(M,S)$\in$ [0,1] and J(M,S)=1 when both M and S are empty.
7: Return Similarity_Score=J(M,S)

mised server. The reason for choosing the threshold as 0 is that legitimate pages within a website have some level of similarity including either images, scripts anchorlinks or styles. We are aware that this threshold also makes the phishing sites bypass the detection techniques when they reuse the resources of compromised legitimate server in their designed phishing site. But, we observed zero instances of phishing sites which reused the files of compromised legitimate server. The other reason for choosing the threshold as 0 is to reduce the number of false positives, because there exists some legitimate sites where there is low level of similarity between the pages within website. For example, similarity between home page and about page of same website may not include large number of common files (scripts, anchorlinks, CSS or images). Hence, Jail-Phish is designed to uncover some phishing sites but not to misclassify the legitimate sites as phishing. Interestingly, the selected threshold score resulted in promising true negative rate of 99.38% and true positive rate of 97.53% as explained in next section.

4.3.2 Experimentation and Results

A. Dataset

To evaluate the performance of Jail-Phish, we used real datasets from two sources.

Table 4.6: Collected dataset information

	LD1	**LD2**	**LD3**	**PD1**	**PD2**
Raw Instances	1000	3000	3000	3770	7170
invalid websites including duplicate	150	369	414	1115	4441
Final dataset	850	2631	2586	2655	2729
Alexa Ranking (for legitimate)	1-1K	1K-100K	500K-1000K	-	-
Phishing submission dates	-	-	-	2 - 10 Sept, 2018	01 Jan - 30 Nov, 2017

The phishing dataset is extracted from PhishTank and the legitimate dataset is collected from Alexa. The properties of the datasets are given in Table 4.6. The phishing dataset is collected at different time interval to check whether the techniques are adaptable to old and live phishing sites. Similarly, the legitimate datasets are also collected at different ranges of Alexa Page ranks to check the performance of our application with both popular, medium and newly registered domains. We divided our datasets into 5 sets namely

- *LD1*- list of low ranked legitimate sites with Alexa Rank ranging from 1 - 1000.
- *LD2*- list of medium ranked legitimate sites with Alexa Rank ranging from 1K - 100K.
- *LD3*- list of high ranked legitimate sites with Alexa Rank ranging from 500K - 1000K.
- *PD1*- list of new phishing sites collected during 02 - 10 Sept, 2018.
- *PD2*- list of old phishing sites collected during 01 Jan - 30 Nov, 2017.

Note that, low Alexa ranked sites are popular websites whereas high Alexa ranked sites are either unpopular or newly registered sites.

B. Experimental evaluation

We conducted various experiments with different datasets for assessing the model. In the Experiment-1, we have evaluated our model with only legitimate dataset to identify the misclassification rate of legitimate data. We used the same legitimate dataset for comparing our work with existing work in Experiment 2. Similarly, we conducted

Experiment 3 to evaluate our model with phishing datasets whereas the same phishing dataset is used for the comparison with existing work in Experiment 4. We also conducted an experiment to evaluate the similarity computation module in Experiment 5. Finally, we did overall comparison with existing works using both legitimate and phishing datasets in Experiment 6.

Experiment 1: Evaluation of Jail-Phish with Legitimate datasets

In this experiment, we attempt to identify the true negative rate of our application. This experiment is carried out using different legitimate sets namely LD1, LD2 and LD3. The motive of conducting experiment on different variations of legitimate sites is to observe the behavior of our application when encountered with popular and newly registered or non-popular sites. From the experimental results, we observed that Jail-Phish achieved a TNR of 100% with LD1, 99.39% with LD2 and 99.11% with LD3 datasets respectively as shown in Table 4.7. The average TNR of 99.50% shows the efficiency of Google search engine to detect the legitimate domains. The decrease in TNR from LD1 through LD3 is due to the fact that low Alexa ranked sites are better indexed by the search engines than the high ranked sites. Note that, for a better anti-phishing system usable in real world scenario, the TNR should be very high i.e. misclassification of legitimate to phishing sites should be low.

Experiment 2: Comparison of Jail-Phish with Varshney et al. (2016b)with legitimate datasets

We also implemented Varshney et al. (2016b) work on the above mentioned legitimate datasets (LD1, LD2 and LD3) for the comparison with our work. The main reason for choosing the Varshney et al. work over the other works is due to the similarity with our method, where the main idea is to leverage the Google search engine with search query as `domain + title` rather than keywords extracted from the textual content. As mentioned earlier, keywords extracted from textual elements have constraints of language dependency and failing cases with image based phishing sites (Figure 3.3). Varshney et al. (2016b) used Google Custom Search Engine (GCSE) for their experimentation but the weakness of GCSE is that it does not include universal search and real

time results [8]. Due to this behavior, the technique results in high false positive rate when non-popular or newly registered domains are encountered. This technique also includes static search query including `domain + title` but it fails to detect legitimate sites with empty or vague titles.

To determine the importance of Google Web Search (GWS), we also implemented the Varshney work with GWS. To ease the presentation, we term the Varshney work with GCSE and GWS as M1 and M2 respectively. GWS is developed using Selenium Webdriver and the search results are parsed using JSoup library.

The results are given in the Table 4.7. It is observed that Jail-Phish outperformed M1 and M2 with an average TNR of 99.50%. M2 ranked better than M1 with an average TNR of 96.38%. Jail-Phish and M2 performed better than M1 due to the use of GWS as it includes real time search results. Since M1 uses GCSE, it resulted in very poor TNR and high FPR. M1 with LD3 reached the maximum FPR of 16.01%. The difference of TNR between Jail-Phish and M1, M2 increases with the increase in high ranked websites and inclusion of non-English websites. This difference is due to the unavailability of efficient search query applied for different kinds of websites. Not all the time, title and domain concatenation would be useful for identifying the legitimate sites because some times high Alexa ranked sites may not get returned in the search results when title is appended to domain but it might work when only domain is used as search query. Some legitimate sites use vague titles like untitled, home, and index etc. leading to their absence in SERP.

Experiment 3: Evaluation of Jail-Phish with phishing datasets

In this experiment, we test our application on two phishing datasets namely PD1 and PD2. The dataset PD1 contains the fresh phishing sites which are collected from 2 Sept 2018 (the day of experimentation) to 10 Sept 2018. The dataset PD2 contains old phishing sites which were still alive on the day of experimentation i.e 10 Sept 2018. The old phishing sites are collected with submission dates starting from Jan 1, 2017 to Nov 30, 2017. The reason for choosing different phishing datasets at different time slots is that, the phishing sites which are alive for a longer time has a greater chance to

[8]https://support.google.com/customsearch/answer/70392?hl=en

Table 4.7: Evaluation of Jail-Phish with LD1, LD2 and LD3 and comparison with M1 and M2

Datasets	**Jail-Phish**			**M1**			**M2**		
	LD1	**LD2**	**LD3**	**LD1**	**LD2**	**LD3**	**LD1**	**LD2**	**LD3**
Final number of instances	850	2631	2586	850	2631	2586	850	2631	2586
Non-English instances	284	1323	1383	284	1323	1383	284	1323	1383
Misclassified instances	0	16	23	20	226	414	7	98	163
Misclassified non-English instances	0	7	9	5	119	220	1	42	75
False Positive Rate	0	0.6082	0.8894	2.3529	8.5899	16.01	0.8235	3.7248	6.3032
True Negative Rate	100	99.3918	99.1106	97.6471	91.4101	83.99	99.1765	96.2752	93.6968
Average TNR		99.5008			91.0157			96.3828	

be indexed by the search engines. The phishing sites hosted on compromised servers might live for a longer time when not noticed by the owner of the website. This results in indexing of phishing sites by the search engines. We countered the phishing sites which are hosted on compromised sites with similarity based features.

We attempted to find the percentage of PSHCS, phishing sites hosted on free hosting server (PSHFHS) in both phishing datasets PD1 and PD2 by manually checking the suspicious URL and its home page. The proportions of PSHCS and PSHFHS are given in Table 4.8. The table also includes total instances of non-English phishing sites, misclassified instances of PSHCS, PSHFHS, non-English phishing sites with Jail-Phish, M1 and M2. From the results, we found that 1889 out of 2655 are PSHCS in PD1 contributing to 71.15 % of total websites and 2237 out of 2729 as PSHCS in PD2 contributing to 81.97 % of total websites. Note that, phishing sites which are hosted on free hosting servers are not considered as PSHCS as the domain is not compromised rather it provides a service for free hosting. From the experimental results of Jail-Phish, we observed that there exists only 9 false negatives leading to 99.66% of TPR. The higher TPR on fresh phishing sites shows that our application is highly adaptable to detection of fresh phishing sites in real time including phishing sites developed with vague titles. We also tested our application on PD2 which resulted in 95.93% TPR.

The difference of TPR with PD1 and PD2 is due to the indexing of phishing sites in the search engine. Because, when the phishing site itself is indexed by the Google search engine as shown in Figure 4.12 then the Jail-Phish classifies the site as legitimate (false negative). There exist three types of phishing sites which can be indexed by the search engine. Firstly, PSHCS, the behavior of phishing sites with Google indexing is mostly observed in PSHCS as shown in Table 4.8. From the Table, it is evident that majority of the misclassified instances are PSHCS i.e. 4 out of 9 in PD1 and 67 out of 111 in PD2. From these results, we conclude that there exists more chance of PSHCS to be indexed by search engine and thus leads to misclassification. The Jail-Phish has misclassified only 4 PSHCS out of 1889 and 67 out of 2237 PSHCS showing the importance of similarity computation module in detecting the PSHCS. Secondly, PSHFHS, similar to the PSHCS, PSHFHS has a chance to get indexed by the Google

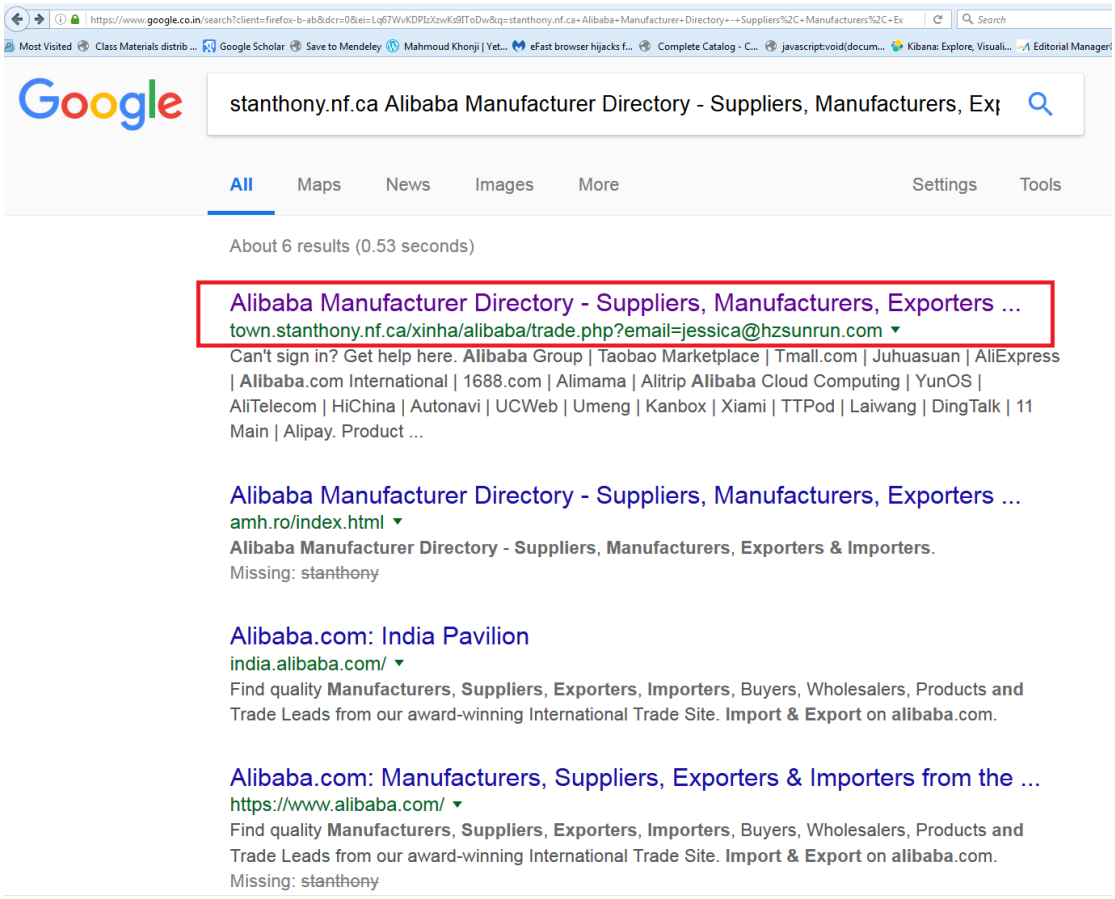


Figure 4.12: Phishing site hosted on compromised server and indexed by search engine

but the number of such instances is not so high compared to PSHCS. We observed two misclassified PSHFHS in PD1 and 29 in PD2. Finally, phishing sites hosted on paid server might get indexed by the search engine but the number is very low. In Jail-Phish, we have encountered only 3 instances (total misclassified - misclassified (PSHCS+PSHFHS)) in PD1 and 15 in PD2.

Experiment 4: Comparison of Jail-Phish with M1 and M2 with phishing datasets

We compared our work with M1 and M2 with respect to PD1 and PD2 datasets. The results are shown in Table 4.8. From the results, it is observed that Jail-Phish outperformed the existing works with an average TPR of 97.7968%. M1 ranked better than M2 with an average TPR of 95.6465%. As M1 uses GCSE, it excludes real time results hence the probability of new phishing sites resulting in search results is lower compared to M2 with GWS. Despite the use of GWS, our Jail-Phish outperformed the other works due to the additional component of similarity computation between the suspicious site

Table 4.8: Evaluation of Jail-Phish with PD1, PD2 and comparison with M1 and M2

Datasets	Our work		M1		M2	
	PD1	**PD2**	**PD1**	**PD2**	**PD1**	**PD2**
Final number of instances	2655	2729	2655	2729	2655	2729
Non-English instances	796	963	796	963	796	963
PSHCS instances	1889	2237	1889	2237	1889	2237
PSHFHS instances	350	113	350	113	350	113
Misclassified instances	9	111	94	141	173	328
Misclassified Non-English instances	3	31	16	28	34	62
Misclassified PSHCS	4	67	19	94	48	223
Misclassified PSHFHS	2	29	33	39	61	56
False Negative Rate	0.3390	4.0674	3.5404	5.1667	6.5160	12.0191
True Positive Rate	99.6610	95.9326	96.4596	94.8333	93.484	87.9809
Average TPR	97.7968		95.6465		90.7325	

and matched site in the search results. Table 4.8 also shows that Jail-Phish could achieve TPR of 99.6610% for PD1, which drops to 95.9326% for PD2. The reason for the drop is due to the limitation of the search engine i.e. if the search engine itself indexed the phishing site as shown in Figure 4.12 then Jail-Phish would misclassify the suspicious site as legitimate. Although, TPR of Jail-Phish with PD2 is lesser than PD1 but still it can be considered as significant detection rate. Note that, TPR of Jail-Phish performed better than both M1 and M2 with a significant difference as shown in the Table 4.8. Even with respect to non-English and PSHCS sites, Jail-Phish outperformed M1 and M2 with a significant detection rate.

Experiment 5: Evaluation of Similarity computation module

In this experiment, we attempt to identify the importance of similarity computation module in detecting phishing websites. Jail-Phish checks the domain of the suspicious URL in SERP with `domain + title` as search query, if not found, then it again checks the domain of suspicious URL in SERP with only domain. If there is a match in SERP then the similarity is computed between the suspicious domain and matched domain.

Jail-Phish without similarity computation module is divided into two cases. `Case 1`: With SQ= `domain + title` , Jail-Phish checks the matching status of suspicious Domain with domains in SERP, if matched then legitimate else phishing. `Case 2`:

Table 4.9: Evaluation of Jail-Phish with and without similarity computation module

	LD1	**LD2**	**LD3**	**PD1**	**PD2**	**Total misclassified instances**
Final dataset	850	2631	2586	2655	2729	-
Misclassified instances with Case 1	7	98	163	173	328	769
Misclassifed instances with Case 2	0	16	23	2249	2350	4638
Misclassified instances with Case 3	0	16	23	9	111	159

Initially, with SQ=`domain + title` is checked for matched domains in SERP if not found then SQ=`domain` is used for the matching of suspicious domain in SERP. If a match is found then the site is classified as legitimate else it is classified as phishing. Jail-Phish with inclusion of similarity computation module is considered as `Case 3`.

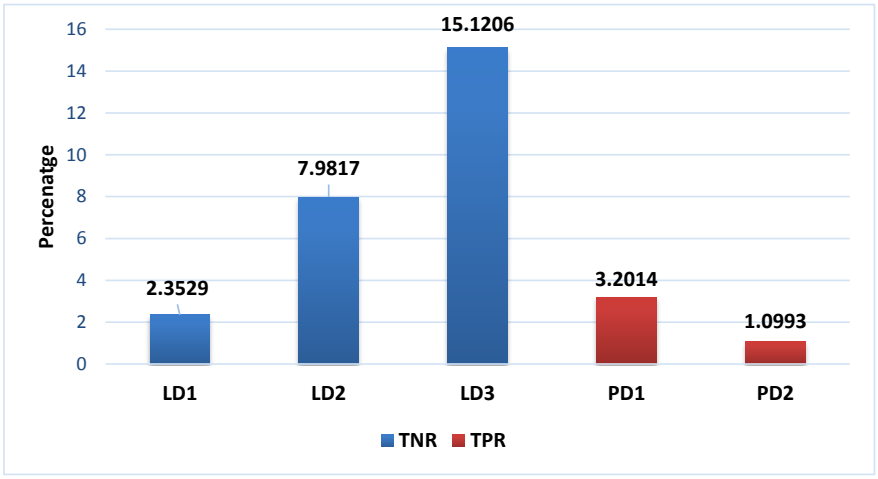
The results of each case are given in Table 4.9. `Case 3` is already discussed in previous experiments 1 and 3 whereas `Case 1` is good for phishing sites detection but fails in detecting non-English and non-popular sites. To overcome the limitation of `Case 1`, search processing is extended with `SQ=domain` for the domains absent in SERP with SQ=`domain + title`. This might help in improving the detection of legitimate sites but it also creates a major problem of classifying the PSHFHS and PSHCS sites as legitimate. Since most of the compromised and free domains are popular, they are returned in SERP when queried with only `domain` leading to classification as legitimate. From the Table 4.8, it is observed that 1889 PSHCS and 350 PSHFHS instances are present in PD1. When these sites are tested with Case 2, all of them are classified as legitimate due to their indexing in Google. Also, we observed 10 instances which are non-PSHCS and non-PSHFHS classified as legitimate. Overall misclassified instances in PD1 with `Case 2` reached to 2249 (1889+350+10). Similarly, PD2 with `Case C2` reached to 2350 due to the existence of PSHCS (2237) and PSHFHS (113). Overall, the number of misclassified instances with `Case 3` is the least among the other Cases which demonstrates the significance of inclusion of similarity computation module in Jail-Phish.

Experiment 6: Overall Comparison of existing works with Jail-Phish

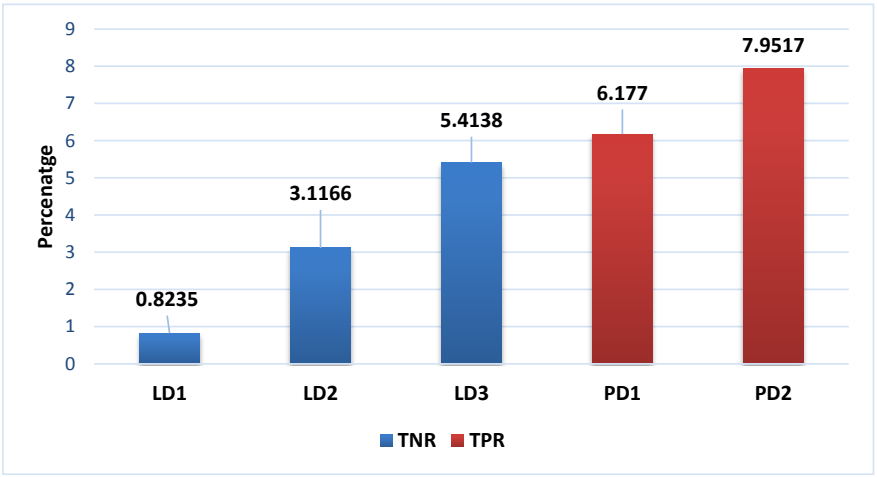
From the earlier experimental results, it is shown that our work outperformed existing

work with a significant gain in TNR and TPR with respect to various legitimate (LD1-LD3) and phishing datasets (PD1-PD2). In this section, we compare Jail-Phish with the existing works with respect to overall legitimate and phishing sites including LD1 to LD3 and PD1-PD2. The metrics which are mentioned earlier are used for comparison. Before discussing the metrics on overall data, we show the statistics of percentage gain in TPR and TNR of both phishing datasets and legitimate datasets with respect to our work and existing works. From the Figure 4.13 (a), it is observed that Jail-Phish has very high percentage gain in TNR with LD3 when compared with M1. This shows that our Jail-Phish method performs effectively when non-popular or newly registered legitimate sites are encountered. Also, we observed that there is a significant improvement in detecting old (PD2) and new phishing sites (PD1) with a percentage gain of TNR of 1.0993% and 3.2014% respectively. Similarly, in Figure 4.13 (b), it is observed that our application performed the best with old phishing sites with a percentage gain of TPR of 7.9517%. This significant improvement is achieved due to the additional component of similarity computation mechanism which is used for detecting phishing sites hosted on compromised servers. From the previous Experiment 5, it is evident that similarity computation module played a significant role in detecting PSHCS. The results in Table 4.9 shows that the technique with `Case 3` (Jail-Phish) achieved least misclassification rate followed by `Case 1` (M2). Also, the Figure 4.13 shows that there is very less difference of performances in TPR of both M2 and Jail-Phish when tested with LD1. This is due to the fact that the popular legitimate sites are returned in search results even with an inefficient search query such as domain with vague titles. For the popular legitimate sites, only single domain as search query is sufficient to return in the search results.

In order to get a clearer and better understanding of detection performance, we considered accuracy, precision and F measure as additional performance metrics. The TPR, FPR for combined PD1, PD2 and LD1, LD2, LD3 for Jail-Phish, M1 and M2 are calculated and are given in Table 4.10. From the table, it is observed that Jail-Phish outperformed M1 and M2 with an accuracy of 98.61% followed by M2 with 93.28%. The precision in the table 99.26% shows that Jail-Phish correctly predicted 99.26% of



(a) Gain in TPR and TNR with M1



(b) Gain in TPR and TNR with M2

Figure 4.13: Gain in TPR and TNR with existing works M1 and M2

phishing sites out of total predicted phishing sites. Since, the total instances in legitimate and phishing classes are not balanced but are equally important, we considered F-measure to evaluate our model. Jail-Phish has achieved a F-measure of 98.51% and performed better than the existing M1 and M2 methods. The TPR of M1 is somewhat closer to Jail-Phish due to the use of GCSE whereas TNR of M2 is closer to Jail-Phish due to the use of GWS. The promising results of Jail-Phish with low FPR of 0.64% and high TPR of 97.77% justifies the effectiveness and feasibility to detect phishing sites using search engines. Note that, for phishing detection, in a real word scenario, a model is considered usable when it has very low FPR (legitimate to phishing) and high F-measure (Jain and Gupta 2018; Whittaker et al. 2010).

Table 4.10: All metrics comparison of our with existing works

Metrics	Jail-Phish	M1	M2
# of legitimate instances (LD1+LD2+LD3)	6067	6067	6067
# of phishing instances (PD1+PD2)	5384	5384	5384
True Negative Rate	99.36	89.12	95.58
True Positive Rate	97.77	95.64	90.69
False Positive Rate	0.64	10.88	4.42
False Negative Rate	2.23	4.36	9.31
Precision	99.26	88.64	94.80
F Measure	98.51	92.00	92.70
Acc	98.61	92.18	93.28

4.3.3 Discussion

A. Jail-Phish Chrome Extension

Our goal is to provide real time protection from website phishing attacks. Hence, we built a chrome extension which classifies the visited URL as a phishing or legitimate. The extension is designed such that no extra clicks or key press are required i.e. on single click, the extension displays the status of website as legitimate or phishing. The chrome extension is written in Javascript which extracts URL, Title and DOM of visited website from the browser and makes a connection to REST API running at the remote server where the actual execution of technique takes place. The REST API is hosted on an Intel Xeon 16 core Ubuntu server with 2.67GHz processor and 16GB RAM.

The REST API feeds the above values to our Jail-Phish application running at the remote server and thus proceeds with 6 stage execution (section 4.3.1) for the phishing detection. Note that, the REST API is implemented with Spring framework. We used POST method for transferring DOM and GET method for URL and Title from extension to Jail-Phish. Once the Jail-Phish receives these values, it classifies the status of website as phishing or legitimate based on the outputs of modules discussed in Section 4.3.1.

Since, our goal is to provide real time phishing detection, the time for visiting matched results and classification has to be very less. Hence, we parallelized the ac-

cessing of each matched result with the help of multiple cores present in the system. Once, the classification is done Jail-Phish returns the status to Rest API which is further sent to Extension as a response.

This whole process of execution took an average time of 4.23 seconds. However, this time is dependent on various factors such as number of matched results, speed of Internet, and speed of the system. In our case, we used a bandwidth of 100mbps and a system with Intel Xeon 16 core Ubuntu server with 2.67GHz processor and 16GB RAM. The output of the extension is a popup window containing the status of the website as shown in Figure 4.14. In image (a) of Figure 4.14, Jail-Phish detects the paypal website as legitimate with a response time of 2.35 sec where as image (b) shows the detection of phishing site with a response time of 3.60 sec in a popup window containing the status of website and textual content recommending the user to close the webpage immediately.

We calculated the time delays at different cases. Case `C1`, `C2` correspond to time taken to detect status of the website with search query : `domain + title` and Case `C3`, `C4` correspond to time taken to detect status of the website with search query: domain.

`Case C1`**:** baseURL of suspicious site matched with baseURL of returned results (it is classified as legitimate). Here, baseURL is termed as protocol + hostname + pathname of given URL.

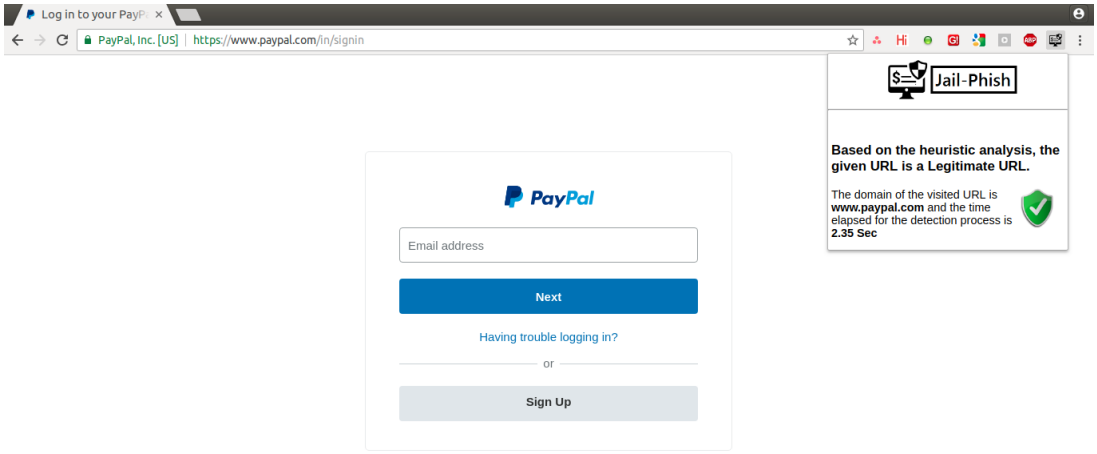
`Case C2`**:** domain of suspicious URL matched with domains in SERP (SQ=`domain + title`)

`Case C3`**:** domain of suspicious URL matched with domains in SERP (SQ=`domain`)

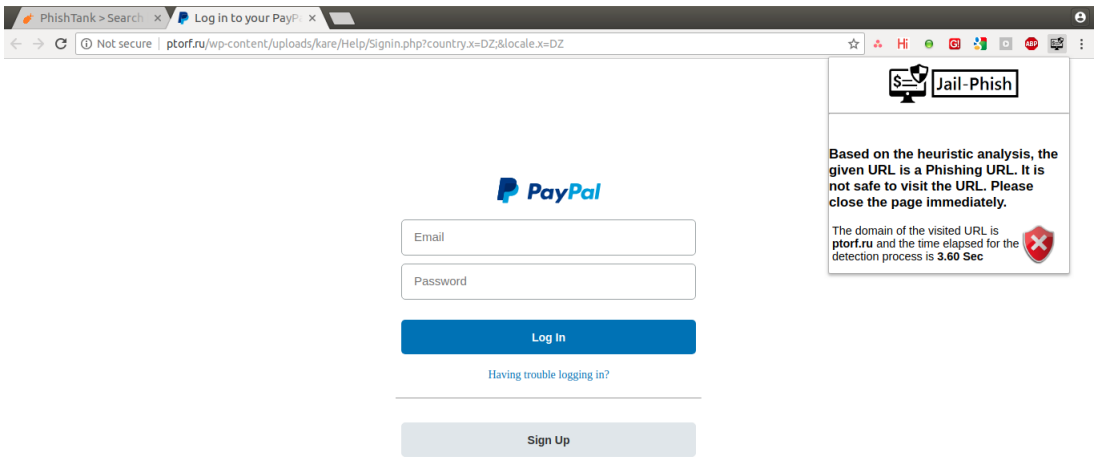
`Case C4`**:** domain of suspicious URL not matched with any of the domains in SERP.

Table 4.11: Average time elapsed for the detection process

Search Query	**baseURL matched in SERP**	**Domain matched in SERP**	**no Domain matching in SERP**
SQ=domain + title	3.41 sec (Case C1)	7.52 sec (Case C2)	-
SQ=domain	-	9.89 sec (Case C3)	5.63 sec (Case C4)



(a) Legitimate site detection



(b) Phishing site detection

Figure 4.14: Output of Jail-Phish extension

For the calculation of time taken for detection process, we tested our chrome extension with 100 phishing sites and 100 legitimate sites. The average time delay at each case is given in Table 4.11. During the experimental study, we observed that majority of popular legitimate sites fell under the `Case C1` and very less number in `C2`. The non-English websites and non popular legitimate fell into the category of `Case C3`. Finally, majority of the phishing sites fell under the category of `Case C4` which took less time than `C2` and `C3`. The time delays for the legitimate sites can be further reduced with help of whitelist of legitimate domains. Jail-Phish requires the computation cost of accessing search engine and maximum cost of accessing matched results in SERP.

B. Effectiveness of Jail-Phish In this section, we discuss the effectiveness of our

technique in detecting phishing websites hosted on compromised servers and the generic phishing sites. Our primary objective of this work is to design a real time application for the phishing detection with high TNR (Legitimate to Legitimate). It is also crucial that the time for the detection process should be very less. We have used Google search engine for the detection of phishing sites. The rationale behind choosing search engine to detect phishing sites is that search engine either least indexes or does not index the phishing sites. The main reason for choosing the Google is due to its wide usage in existing literature. Also, according to Net Market Share[9], 74.54 % of the Search Engine market has been acquired by Google in 2017. As it is more popular and powerful in search engine market, we have chosen the Google search engine to implement our method. We did not test our method with other search engines but it is adaptable to any search engine such as Bing, Yahoo, DuckDuckGo or Baidu etc.

The textual language of the website is a bottleneck in most of the search engine based techniques which leads to failure cases with non-English website. Moreover, techniques relying on textual content of the website for the keywords extraction results in empty keywords extraction with image based phishing. Hence, we have chosen a search query as `domain + title` which is independent of textual content of the website. But, the considered query also has resulted more false positives when encountered with majority of non-English and low reputed website. Hence, we extended the search processing with a new search query containing only Domain which made the technique more powerful in detecting non-English and low reputed legitimate sites.

But, due to the use of `domain` or `domain + title` as search query, the technique also returned PSHCS in SERP. This made the technique to classify the PSHCS as legitimate thus leading to increase of false negative rate. Hence, we used similarity computation between the visited website and matched domain in SERP. If the similarity is above a certain threshold then it is classified as legitimate else phishing. The rationale behind this decision is that the webpages within a website holds some level of similarity with respect to CSS, Javascript, images or anchor links. We have intentionally chosen the similarity threshold as zero for the classification of website. As mentioned earlier

[9]https://www.netmarketshare.com/search-engine-market-share.aspx

in Stage 6, the reason for choosing zero threshold is to reduce the number of false positives. But due to this zero threshold, Jail-Phish might loose some phishing sites which use web resources of compromised server.

4.4 SUMMARY

In this chapter, we have presented two techniques for the detection of PSHCS. Out of which, one technique (Curb-Phish) detects phishing sites based on the dissimilarity between the visiting page and the home page of the given URL and the other technique (Jail-Phish) detects PSHCS by calculating similarity between the domain in SERP and suspicious site.

Firstly, the Curb-Phish is presented to detect PSHCS using similarity based feature and in addition to these, the basic URL and hyperlink-based features are proposed to counter the remaining malicious registered phishing sites. A non-parallel plane classifier, Twin Support Vector Machine is used for the classification of phishing and legitimate sites.

With the help of aforementioned heuristic features, we achieved a TPR of 98.72%, TNR of 98.08% and an accuracy of 98.4%. We also achieved TPR of 99.66% in an experiment on PSHCS which shows the richness of similarity-based features in detecting phishing sites hosted on compromised sites. But, the overall accuracy of an experiment on PSHCS resulted in 96.78% due to the FPR of 5.07%.

Secondly, to reduce the FPR and thereby improve the accuracy of the model, we have presented a heuristic technique (Jail-Phish) which uses search engine results and similarity-based features to detect the phishing sites. The main advantage of Jail-Phish is that it not only detects phishing sites with malicious registrations but also detects phishing sites which are hosted on compromised sites or free website builders. Note that, both the Curb-Phish and Jail-Phish are language independent and does not depend on prior access to the database of target website resources or web history.

We evaluated Jail-Phish with popular legitimate set, unpopular legitimate set and observed a TNR of 99.36%. Similarly, our application is also tested on old phishing set, new phishing set and observed a TPR of 97.77%. These results indicate that this

application is adaptable to all kinds of legitimate sites and phishing sites specifically as it performs the best with long lived phishing sites. The increase in true negative rate is achieved through variable search query whereas high true positive rate is achieved through similarity based mechanism.

Though, these techniques achieved a significant detection rate, they have certain limitations of accidental download of malware by visiting the suspicious URL. Hence, we present an URL based mechanism in the next chapter which detects phishing sites without even visiting the URL. As the technique does not need source code, the technique alerts the user at a very low response time.

CHAPTER 5

URL BASED PHISHING DETECTION

There exist many anti-phishing techniques which use source code-based features and third party services to detect the phishing sites. These techniques have some limitations and one of them is that they fail to handle drive-by-downloads. They also use third-party services for the detection of phishing URLs which delay the classification process. Hence, in this chapter, we present two lightweight techniques which detect the status of the URL without even visiting the URL. The first technique named as **CatchPhish** which predicts the URL legitimacy by only analyzing the URL with the use of hand-crafted features and Term Frequency - Inverse Document Frequency (TF-IDF) features. The hand-crafted features are extracted from full URL and hostname of the given URL. For the extraction of TF-IDF features, a well-known information retrieval algorithm named as Term Frequency and Inverse Document Frequency is applied on the set of URLs (URL dataset). The technique is implemented as a web application where it takes a set of URLs as input and gives the status of the URLs as output.

To further improve the performance, we present the second technique (**PhishDump**) which uses deep neural networks for the extraction of features from the input URLs. The deep learning algorithm not only outperforms start-of-the-art machine learning methods but also comes with the benefit of omitting difficult feature engineering. The PhishDump is based on the multi-model ensemble of Long Short Term Memory (LSTM) and Support Vector Machine (SVM) classifier. It is designed for the phishing detection in mobile devices.

5.1 INTRODUCTION

Some of the existing techniques use a blacklist to detect phishing websites. Limitation of this approach is that it fails to detect phishing websites which are not listed in the blacklist (i.e. zero day attacks) and also websites with content similar to that of blacklisted phishing site. Many heuristic feature-based techniques extract third party-based and source code-based features to detect the phishing sites. But using third party services such as page rank, search engine indexing, and WHOIS (age of the domain) has some limitations. One of the limitation is that use of third-party services may be insufficient to detect phishing sites hosted on compromised servers and these websites are incorrectly classified as legitimate sites because they are listed in the search results. Lifespan of the websites hosted on compromised servers is generally more than a day unlike other phishing sites which is just few hours. Also, this approach incorrectly classifies the new legitimate site as fake due to the absence of age of the domain. Source-code based feature extraction is time consuming and it is difficult to handle pages with iframes.

Moreover, in addition to the above limitations, the techniques in the literature designed for the desktop may not work for the mobile devices due to the hardware limitations. It is also to be noted that the number of mobile users has increased rapidly due to their portability, smaller size and support of wide range of applications. Using mobile devices for online payments, buying and financial activities have made attackers to target mobile users. Attacker attempts to trick the mobile users to disclose information such as credit card, bank account details etc. which is used for his financial benefit. Due to lack of awareness, mobile users become victims of social engineering attacks. The mobile phishing can be performed through SMS, webpages, E-mails and mobile applications. Many users install applications from the third-party app stores which provides the door for the attacker to perform an attack.

Mobile webpages differ from their desktop versions in terms of content, layout, and functionality. Due to the small screen size of mobile devices, the visibility of URL, page layout, content in the browser is limited whereas the users can have a relatively larger view in desktop browsers. The attackers create phishing site which looks similar to that

of legitimate site. Thus, it is difficult for any mobile user to distinguish between legitimate and fake website. For instance, websites may be blacklisted in desktop browsers but same websites may not be blacklisted in mobile web browsers. Most of the anti-phishing techniques were designed to detect malicious websites in desktop computers and they may not work for mobile webpages due to the hardware limitations, difference in screen size and layout.

The presented URL based models have some advantages over existing works and are given as follows.

- **Language independent:** As our models does not rely on the textual content of the website, the models detects the phishing sites independent of the language used in its content.
- **Third-party independent:** The URL-based features are extracted without any dependency on third-party services.
- **Client-side adaptability:** In our presented models, the extracted features from URL are light-weight and hence, adaptable at client-side.
- **Target independent detection:** Unlike image-based techniques (Ardi and Heidemann 2016; Mao et al. 2017; Rao and Ali 2015a), our technique does not depend on pre-stored database of target legitimate sites. This makes our technique capable of detecting phishing sites targeting any legitimate site.
- **Drive-by-download independent:** As the feature extraction does not require visiting the website, the download of malicious software is avoided.

Hence, in this chapter, we have presented two techniques which detects phishing sites only by inspecting the URLs which can be used for both mobile and desktop devices. But in our methods, we designed CatchPhish as a web application which takes input as set of URLs and gives status as output whereas phishdump is implemented as an android application to detect the phishing sites in the mobile devices.

5.2 CATCHPHISH : DETECTION OF PHISHING WEBSITES BY INSPECTING URLS

It is observed that there exist many techniques which are semi automated and hence they cannot be used in real-time. Thus there is a need for a technique which provides automated detection of phishing sites. In this section, we present a model which detects the phishing sites at the client side using URL-based features. Following are the contributions of our model:

- We have proposed a light-weight model with 12 novel hand-crafted features (H6, F6, F8-F9, F11-F18) along with 23 existing features (F1-F5, F7,F10, F19, H1-H5, H7-H16) adopted from the literature.
- We have proposed a model, learning large-scale Term Frequency - Inverse Document Frequency (TF-IDF) features for the classification of suspicious URLs either as legitimate or phishing
- We have proposed Phish-hinted blackwords for the detection of phishing sites.

In this chapter, we have presented a model to predict the URL legitimacy by only analyzing the URL with the use of hand-crafted features and TF-IDF features. The hand-crafted features may include presence or count of special characters, words in the URL. For the extraction of TF-IDF features, a well-known information retrieval algorithm named as TF-IDF (Salton and McGill 1986) is applied on a set of URLs. TF-IDF assigns weight for each of the words based on its frequency within a document and its existence in other documents of a corpus. The combined set of hand-crafted and TF-IDF features are fed to machine learning algorithm for URL classification. Note that we applied TF-IDF algorithm on URLs rather than website content used in the existing literature of phishing detection.

5.2.1 Methodology

Generally attackers' intention is to construct the phishing URLs in such a way that they appear as legitimate sites to the users. Attackers use various URL obfuscation techniques to trick the users to reveal the personal information which can be misused.

Table 5.1: Components of URL with an example

Component	Example
URL	http://www.support.example.co.in/software/?tag=net&order=new#top
Protocol	http
Subdomain	support
Primary Domain	example
Domain	example.co.in
Base URL	http://www.support.example.co.in/software
gTLD	co
ccTLD	in
Hostname	support.example.co.in
Path	/software
Query	tag=net&order=new
Fragment	top

The main idea of this work is to detect phishing sites on the fly using light-weight features. This is achieved by extracting features only from URL string without visiting the website. Before going to the architecture of our model,a brief description about URL components is discussed in the section below.

A. URL components

URL stands for Uniform Resource Locator. It is used to locate a resource such as hypertext pages, images, audio files etc. on the web. Table 5.1 shows the various components of a URL with an example. The first component of the URL is protocol (https, http, ftp etc.) which is used to access the resource. The Second component indicates hostname or IP address where the resource is located. Hostname is subdivided into Subdomain, Primary Domain and Top-level Domain (TLD). The primary domain and TLD together represent domain name of the URL. TLD is further subdivided into generic TLDs (gTLD) and country-specific TLDs (ccTLD). Hostname is followed by a Port number which is an optional field. The third component, Path is used to identify the specific resource within the domain requested by the user. Path is followed by two optional fields i.e, query and fragment. Query is always preceded by question mark (?) and fragment is preceded by hash (#). Base URL is calculated by concatenating protocol, hostname and path of the URL. The generic format of a URL is as follows:

```
<Protocol>://<Subdomain>.<Primary domain>.<TLD>/<Path
```

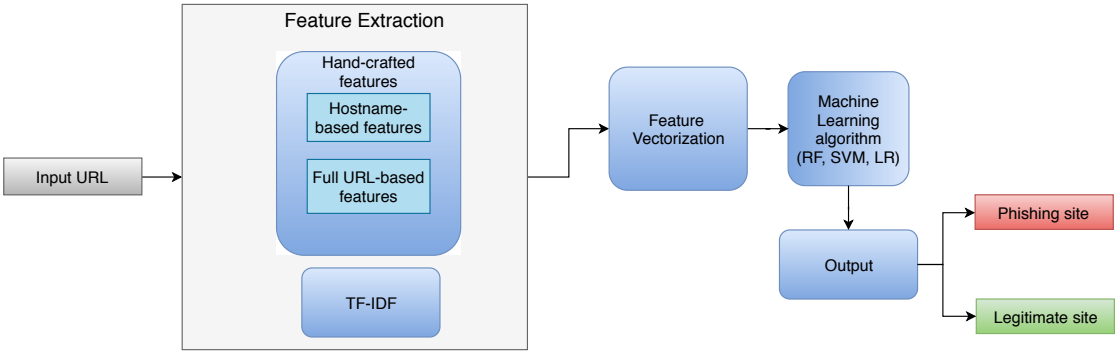


Figure 5.1: Architecture of CatchPhish

```
domain><?query><#fragment>
```

The architecture of our work is shown in Figure 5.1.

We have divided the dataset into training and testing sets. The training set is given as input to the feature extractor for extracting URL-based features. These features are used to train the machine learning model to detect the phishing sites. To evaluate the performance of the trained model, we extract the URL-based features from testing set which are further fed to the trained model.

B. Feature extraction

A Python program has been implemented which takes URL of a website as an input, extracts features from different parts of URL and the features can be either hand-crafted or obtained from TF-IDF. These extracted features are fed to the trained model using machine learning algorithms to classify the website either as legitimate or phishing.

The extracted URL-based features are classified into two categories:

1. Hand-crafted features
2. TF-IDF features

1. Hand-crafted features

These are the features that are manually crafted and are obtained from python program when URL is given as input. Based on the URL element (hostname or path) used for extraction of features, hand-crafted features are broadly classified into two cate-

gories which are as follows:

- Full URL-based features
- Hostname-based features

Table 5.2 provides a list of Full URL-based and Hostname-based features.

Full URL-based features: These features are extracted from the entire URL string and some of these features such as F1-F2 (Lin et al. 2013; Ranganayakulu and Chellappan 2013), {F3-F4, F10} (Chu et al. 2013), F5 (Marchal et al. 2016; Shirazi et al. 2018), F7 (Choi et al. 2011), F19 (Marchal et al. 2016; Shirazi et al. 2018) are adopted from existing works and F6, F8-F9, F11-F18 are the newly proposed features. Full URL-based features are classified into three categories:

- **Rare tokens in legitimate but frequent in phishing sites:** There exists many tokens or special characters which appear most frequently in URL string of phishing sites but not in legitimate sites. This characteristic feature is used to distinguish between phishing sites and legitimate sites. These features check the presence of special characters (@, *, $, :, —, , ’,’, _), phish-hinted words either in the base URL or path of the entire URL string. List of phish-hinted words are extracted from phishing URLs and is discussed in the further section. For example, consider a feature that checks the presence of @ in the URL. If present then it might be phishing site else a legitimate site. Features F1, F6, F12-F17 come under this category.

- **Count-based features:** These features get the count of particular characters or length of tokens either in the path or base URL. The importance of these features is that, if the count or length is higher, then it indicates that the input URL is possibly a phishing URL. For example, consider the feature that gets the count of dots in the URL string. More the number of dots, there is more probability that the URL is phishing as dots are used to hide the brand name in the URL. Features F2-F5, F8-F11, F18 come under this category.

Table 5.2: List of features

Hostname-based Features	Full URL-based Features	Full URL-based Features
H1: Presence of @ in the hostname	F1: Presence of @ in URL	F17: Presence of White space in the base URL
H2: Digits count in the hostname	F2: Digits count in the path	F18: Ratio of hyphens
H3: Average word length in the hostname	F3: Average word length in the path	F19: Presence of HTTPS in the base URL
H4: Longest word length in the hostname	F4: Longest word length in the path	
H5: Hostname length	F5: Base URL length	
H6: Presence of Phish-Hinted words in the hostname	F6: Presence of Phish-Hinted words in the path	
H7: Presence of brand name in the Subdomain	F7: Presence of brand name in the path	
H8: Length of the domain	F8: Question mark count in the URL	
H9: Digits count in the domain name	F9: Slash count in the path	
H10: Number of dots in the hostname	F10: Tokens count in the path	
H11: Number of hyphens in the hostname	F11: HTTPS count in the path	
H12: Number of underscores in the hostname	F12: Presence of $ in the base URL	
H13: Check hostname for IP address	F13: Presence of comma in the base URL	
H14: Validate TLD	F14: Presence of * in the base URL	
H15: Presence of multiple TLDs	F15: Presence of OR symbol in the base URL	
H16: Presence of underscores in the hostname	F16: Presence of Semicolon in the base URL	

- **Other features:** Some of the other features which do not fall in any of the above categories are discussed in this section. Feature F7 comes under this category and this feature checks the presence of white-listed brand names in path of the URL.

Brand names are obtained from PhishTank and are stored in a file. PhishTank maintains a list of phishing URLs and their target legitimate websites. These target legitimate sites are considered as brand names for feature extraction.

Hostname-based features: These features are extracted from the hostname or domain name of the URL. Features {H1-H2, H5, H8-H9} (Lin et al. 2013), H3-H4 (Chu et al. 2013), H7 (Choi et al. 2011), {H10, H13} (Ranganayakulu and Chellappan 2013), H11 (Wang and Shirley 2015; Xiang et al. 2011), H14-H15 (Gowtham and Krishnamurthi 2014; Zuhair et al. 2016),{H12, H16} (Bottazzi et al. 2015; Patil and Patil 2018) are adopted from the existing techniques and H6 is newly proposed feature used for detection of phishing sites. Hostname-based features are divided into three classes and they are:

- **Rare tokens in legitimate but frequent in phishing sites:** These features are similar to that of full-URL based features that check for presence of rare tokens as mentioned above. The only difference is that features in this category check the presence of tokens or special characters either in hostname or domain name of the URL but not in the entire URL string. Features H1, H6, H16 fall under this class.

- **Count-based features:** These features are similar to that of full URL count-based features. The difference is that length or count of a token or special characters are calculated either from hostname or domain name of the URL unlike full URL-based features. Features H2-H5, H8-H12 come under this category.

- **Other features:** These features do not fall in any of the previous classes. Feature H7 checks the presence of brand name in the sub domain of the given URL whereas H13 checks if hostname is an IP address. H14 checks the position of TLD in the hostname and validates it. Similarly H15 checks the presence of multiple TLDs in the hostname of the URL.

Phish-Hinted words: There exists few words or tokens that are common to most of the phishing URLs. We obtained a list of 16 such words called as Phish-Hinted words

(above predefined threshold) from 40,668 phishing URLs (PhishTank) and Table 5.3 provides the list of phish-hinted words with their frequencies. This list is independent of protocols, gTLDs and ccTLDs. These phish-hinted words are used as indicators of phishing behavior. The extraction of phish hinted words include tokenization of phishing URLs using $comma, ., -, /, (,), [,]$ as delimiters. The most frequent words above a certain threshold **(500)** are considered as phish-hinted words. We also calculated the most frequent words from the legitimate URLs to check the negative impact of phish hinted words on legitimate URLs. We observed that, none of the phish-hinted words listed in the Table 5.3 were frequent in legitimate URLs. Features F6 and H6 mentioned in the Table 5.2 checks if these phish-hinted words are present in the input URL and hostname respectively. If it is present, then the probability of input URL being phishing is more than input URL being legitimate.

Table 5.3: Phish-Hinted words

Phish-Hinted word	**Frequency in Phishing URLs**	**Frequency in legitimate URLs**
wp	4592	85
login	3108	304
includes	1950	4
admin	1812	14
content	1655	233
site	1556	157
images	1491	72
js	1172	5
alibaba	984	0
css	827	6
myaccount	818	2
dropbox	639	3
themes	606	16
plugins	606	7
signin	581	4
view	534	316

2. TF-IDF features TF-IDF weight is a statistical measure used to extract information and it tells the importance of a word in a collection of documents. Importance of a word is directly proportional to the number of times a word appears in a corpus or

collection of documents. It is computed using 2 terms and they are as follows:

- **Term Frequency (TF):** It gives frequency of a term in a document. Since each document is of different size, it is divided by the terms count in the document. If *t* is a term, *d* is a document then term frequency is given as:
$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \epsilon d} f_{t',d}} \tag{5.1}$$
where $f_{t,d}$ is frequency of term t in document d.

- **Inverse Document Frequency (IDF):** It gives the importance of a term in corpus. It is calculated as logarithm of ratio of total number of documents to number of documents in which term is present. If *t* is a term, *D* is a corpus or collection of documents then inverse document frequency is calculated as:
$$IDF(t,D) = \log \frac{n}{|d \epsilon D : t \epsilon d|} \tag{5.2}$$
where n is documents count in D.

Thus a term with high TF-IDF means that it has high term frequency in a document and low document frequency in a corpus. TF-IDF is product of TF and IDF. TF-IDF is given by the equation:

$$TF - IDF(t,d,D) = TF(t,d) * IDF(t,D) \tag{5.3}$$

It is observed that TF-IDF has been applied to get the indirect associated links (Ramesh et al. 2014), target website (Xiang et al. 2011) and legitimacy of suspicious website (Zhang et al. 2007). Though TF-IDF extracts prominent keywords from the textual content of the website, it has some limitations. One of the limitations is that the technique fails when inappropriate keywords are extracted i.e. non brand names are selected using TF-IDF when the brand names are misspelled, skipped or replaced with images. The other limitation is that TF-IDF is language specific due to its dependency on textual content.

In our presented work, we applied TF-IDF on URL instead of content of the website where each URL is termed as document and set of URLs are termed as corpus *D* . The term *t* is calculated by tokenizing the URL with delimiters(-) and set of these tokens are considered as words in a document. These tokens of entire corpus are fed to TF-IDF

algorithm to extract the TF-IDF features.

C. Feature Vectorization

In this section, we integrate novel, existing hand-crafted features with TF-IDF features to generate feature vector required for training our model. The hand-crafted features component generates 35-dimensional feature vector as $H_v =< f_1, f_2, f_3....., f_{35} >$ whereas TF-IDF component generates k-dimensional feature vector as $T_v =< c_1, c_2, c_3, ..., c_k >$ where k is the size of dictionary calculated from URL corpus. From the experimental analysis, we observed the size of dictionary k=328992. The above two feature vectors are concatenated to form final feature set F_v which is fed as input to machine learning algorithms for the URL classification. $F_v = H_v + T_v =< f_1, f_2, f_3...., f_{35}, c_1, c_2, ..., c_k >$. As the size of TF-IDF dictionary increases with increase in number of URLs, we converted the final feature vector into sparse vector to avoid the memory problem.

D. Machine learning algorithms

To evaluate the performance of hand-crafted features and TF-IDF features, we applied various machine learning classifiers such as XGBoost, Random Forest, Logistic Regression, K-Nearest Neighbour, Support Vector Machine and Decision tree to train our model. The main intention of comparing various classifiers is to choose the best classifier suitable for our feature set. To implement various machine learning classifiers, Scikit-learn package[1] is used and python is used for feature extraction. From the experimental results, we observed that RF outperformed other classifiers.

5.2.2 Experimentation and Results

CatchPhish takes suspicious URL as input and outputs the status of the URL as legitimate or phishing. A Python program has been implemented to extract the features from the input URL. These features are either obtained from TF-IDF algorithm or hand-crafted from the URLs which are further combined to form a feature vector. This feature vector is fed to machine learning algorithm i.e, Random forest (RF) to detect the legitimacy of the URL.

A. Dataset

[1] http://scikit-learn.org

We have collected the dataset from three different sources. Legitimate sites are collected from common-crawl and Alexa database whereas phishing sites are collected from PhishTank. We have divided the dataset into 5 sets where D1, D2, D3 are our sets and D4, D5 are datasets used in existing literature as shown below:

- *D1:* Legitimate sites from Alexa database and phishing sites from PhishTank
- *D2:* Legitimate sites from common-crawl and phishing sites from PhishTank
- *D3:* Legitimate sites from both common-crawl and Alexa database whereas phishing sites from PhishTank
- *D4:* Legitimate sites from Yandex and phishing sites from PhishTank (Sahingoz et al. 2019)
- *D5:* Legitimate sites from DMOZ and phishing sites from PhishTank (Marchal et al. 2014)

The extracted features are divided into three different feature sets as shown below.

- *FS1:* Hand-crafted features
- *FS2:* TF-IDF features
- *FS3:* Hand-crafted + TF-IDF features

B. Experimental evaluation

We have conducted 5 experiments on the datasets mentioned above. Distribution of legitimate and phishing sites from three sources are shown in Table 5.4. To evaluate the performance of our model, we have used traditional metrics such as Sensitivity or TPR, Specificity or TNR, FPR, FNR, Accuracy (Acc), Precision (Pre), F-measure (F).

Experiment 1: Evaluation of FS3 on D3 with various classifiers

In this experiment, we extracted hand-crafted and TF-IDF features from D3 and applied various classifiers such as XGBoost, Random Forest, Logistic Regression, KNN, SVM and Decision Tree. The main idea of this experiment is to choose the best classifier

Table 5.4: Dataset used in our experimentation

Type	Source	Sites	URL
Legitimate	Common-crawl	42220	http://index.commoncrawl.org/
	Alexa database	43189	http://www.alexa.com/topsites
Phishing	PhishTank	40668	http://www.phishtank.com/developer

Table 5.5: Evaluation of FS3 on D3 with various classifiers

Classifier	TPR (%)	FPR (%)	TNR (%)	FNR (%)	Accuracy (%)	Precision (%)	F-measure (%)
XG Boost	87.94	11.16	88.84	12.06	88.18	95.67	91.64
RF	93.31	3.35	96.65	6.69	94.26	98.59	95.88
LR	91.09	5.30	94.70	8.91	92.07	97.87	94.36
KNN	89.44	24.15	75.85	10.56	84.92	88.15	88.79
SVM	87.91	14.83	85.17	12.09	87.15	93.96	90.83
Decision Tree	93.23	11.66	88.34	6.77	91.71	94.62	93.92

suitable for our dataset. The experimental results are shown in Table 5.5. From the results, it is observed that Random forest outperformed the others with an accuracy of 94.26%. Since D1, D2 are subsets of D3 and RF performed better in classification, RF is used in the further experiments.

Experiment 2: Evaluation of FS1 on D1, D2 and D3

In this experiment, hand-crafted features extracted from legitimate and phishing URLs are evaluated using Random Forest (RF) classifier. This model is tested on all three sets of data i.e, D1, D2 and D3. Experimental results are shown in Table 5.6. Set D1 consist of legitimate URLs from Alexa database and Phishing URLs from PhishTank. From the experimental results, it is clear that FPR for D1 is 5.79% which is lesser than that of D2 and D3. It is also observed that the hand-crafted features for D1 has outperformed D2 and D3 with respect to accuracy.

Table 5.6: Evaluation of handcrafted features using RF classifier

Sets	TPR (%)	FPR (%)	TNR (%)	FNR (%)	Accuracy (%)	Precision (%)	F-measure (%)
D1	94.41	5.79	94.21	5.59	94.32	94.56	94.49
D2	88.60	10.69	89.31	11.40	88.95	89.86	89.23
D3	91.67	12.90	87.10	8.33	90.28	94.22	92.93

Table 5.7: Evaluation of TF-IDF features using RF classifier

Sets	TPR (%)	FPR (%)	TNR (%)	FNR (%)	Accuracy (%)	Precision (%)	F-measure (%)
D1	94.17	6.15	93.85	5.83	94.01	94.21	94.19
D2	91.04	5.68	94.32	8.96	92.58	94.76	92.86
D3	92.04	3.55	96.45	7.96	93.25	98.56	95.19

The reason behind more accuracy and less FPR for D1 is due to the use of legitimate URLs from Alexa database. Alexa provides a service of ranking the legitimate domains such as facebook.com, youtube.com, microsoft.com. These domains may not contain subdomain and path domain whereas the phishing sites from PhishTank contain path domain and special characters. This helps the classifier to clearly differentiate between the phishing and legitimate sites using the extracted hand-crafted features. As many of the existing works (Chiew et al. 2019; Marchal et al. 2016; Shirazi et al. 2017; Verma and Dyer 2015; Xiang et al. 2011) in the literature have used Alexa service for collecting the legitimate dataset, this motivated us to choose Alexa as a source of legitimate home page URLs. We have conducted this experiment on D2 i.e, legitimate sites collected from common-crawl and phishing sites from PhishTank in order to ensure that legitimate sites include other pages of the website apart from home page. In real-time, there exists legitimate sites with home page and also non-home pages. In order to evaluate the model in the real-time scenario, we have performed the experiment on D3 which include legitimate sites both from Alexa database, common crawl and phishing sites from PhishTank.

Experiment 3: Evaluation of FS2 on D1, D2 and D3

In this experiment, we have tokenized the input URL using special characters such as hyphen(-), slash(/), comma(,) and dot(.) as delimiters. These tokens are given to TF-IDF for the extraction of features and these features are fed to RF to classify the input URL either as legitimate or phishing. This experiment is performed on all the datasets D1, D2 and D3. Table 5.7 shows experimental results. From the results, it is clear that the accuracy for D1 is 94.01% which is higher than that of D2 and D3.

Experiment 4: Evaluation of FS3 on D1, D2 and D3

In this experiment, both hand-crafted features and TF-IDF features are fed to RF clas-

Table 5.8: Evaluation of hand-crafted and TF-IDF features using RF classifier

Sets	TPR (%)	FPR (%)	TNR (%)	FNR (%)	Accuracy (%)	Precision (%)	F-measure (%)
D1	95.32	3.96	96.04	4.68	95.67	96.31	95.81
D2	92.66	4.61	95.39	7.34	93.95	95.71	94.16
D3	93.31	3.35	96.65	6.69	94.26	98.59	95.88

Table 5.9: Comparison of our work with existing technique

Metrics (%)	W1 (Sahingoz et al.)	W2 (Our work(D4))	W2′ (Our work(D3))	W3 (Huang et al.)	W2" (Our work(D5))	W4 (Marchal et al.)
Precision	97.00	98.04	98.59	99.71	98.57	98.44
Sensitivity	99.00	98.42	93.31	69.49	96.49	91.27
F- measure	98.00	98.23	95.88	81.90	97.52	94.72
Accuracy	97.98	98.25	94.26	70.15	97.49	94.91

sifier. The classifier detects the phishing sites based on the input features obtained from three sets of URLs i.e, D1, D2 and D3. Results of the experiment are shown in Table 5.8. The reason behind combining hand-crafted and TF-IDF features is to check the diverseness of features. From the results, it is clear that accuracy obtained in this experiment is better compared to previous experiments. Accuracy of D1 is 95.67% and similar to the previous experiments, D1 outperformed D2 and D3 even in this experiment.

Experiment 5: Comparison of CatchPhish with existing techniques

In this section, we compare our work with Sahingoz et al. (2019), Huang et al. (2012) and Marchal et al. (2014) to evaluate the performance of our model. Note that, we implemented Huang et al. work on our dataset D3 whereas for the comparison with Sahingoz et al. and Marchal et al. works, we have applied our model on their datasets D4 and D5 respectively. For simplicity, we term Sahingoz et al. work as W1, Huang et al. work as W3, Marchal et al. work as W4 whereas our work with D4 is termed as W2, our work with D3 is termed as W2′ and our work with D5 is termed as W2". The main reason for the comparison with W1, W3 and W4 is due to similarity in type of URL features, and their extraction process.

Unlike content based phishing detection, these techniques detect phishing sites based on URLs. The results are shown in Table 5.9. Our model W2 outperformed existing work W1 with an accuracy of 98.25% and precision of 98.04%. It is also observed that

Table 5.10: Results of CatchPhish model on D4 and D5

Dataset	Set	TPR (%)	FPR (%)	TNR (%)	FNR (%)	Accuracy (%)	Precision (%)	F-measure (%)
D4	FS1	96.38	4.81	95.19	3.62	95.77	95.02	95.69
	FS2	98.12	2.75	97.25	1.88	97.68	97.16	97.64
	FS3	98.42	1.92	98.08	1.58	98.25	98.04	98.23
D5	FS1	91.62	6.39	93.61	8.38	92.59	93.78	92.68
	FS2	95.74	2.22	97.78	4.26	96.74	97.83	96.77
	FS3	96.49	1.47	98.53	3.51	97.49	98.57	97.52

the sensitivity of W1 is 0.58% greater than W2 due to the use of third-party services. Our model W2′ achieved an accuracy of 94.26% whereas W3 achieved an accuracy of 70.15%. From the experimental results, precision of W3 is 99.71% which is higher than that of our work W2′. But, sensitivity of W3 is 69.49% which is much lesser than the sensitivity of W2′ which makes W3 less efficient in detecting the legitimate sites. Finally, our model W2" performed better than W4 with an accuracy of 97.49% and sensitivity of 96.49% which shows the significance of detecting phishing sites over existing work. We have also tested our features sets FS1-FS3 on the existing dataset D4, D5 and the results are given in Table 5.10. From the results, it is observed that FS3 had outperformed other feature sets for both D4, D5 with a significant accuracy of 98.25% and 97.49% respectively.

5.2.3 Discussion

Early detection of phishing sites is the main idea of our presented work. Hence we developed an application `CatchPhish` which is based on client-server model. User Interface (UI) of the CatchPhish application is shown in Figure 6.4. The application takes input as single or set of URLs and gives the status of the URLs as Legitimate highlighted with Green color and Phish highlighted with Red color. The demo of CatchPhish application is available at `http://isea.nitk.ac.in/catchphishdemo/client.html` and the dataset D1, D2 and D3 are available at `https://tinyurl.com/catchphish`. The client gives a set of suspicious URLs as input through the UI, client-server connection happens through REST API residing on the server. The legitimacy of input URLs are verified at the server side and finally the result is sent back to the client. An Intel Xeon 16 core Ubuntu server with 2.67GHz processor and 16GB RAM

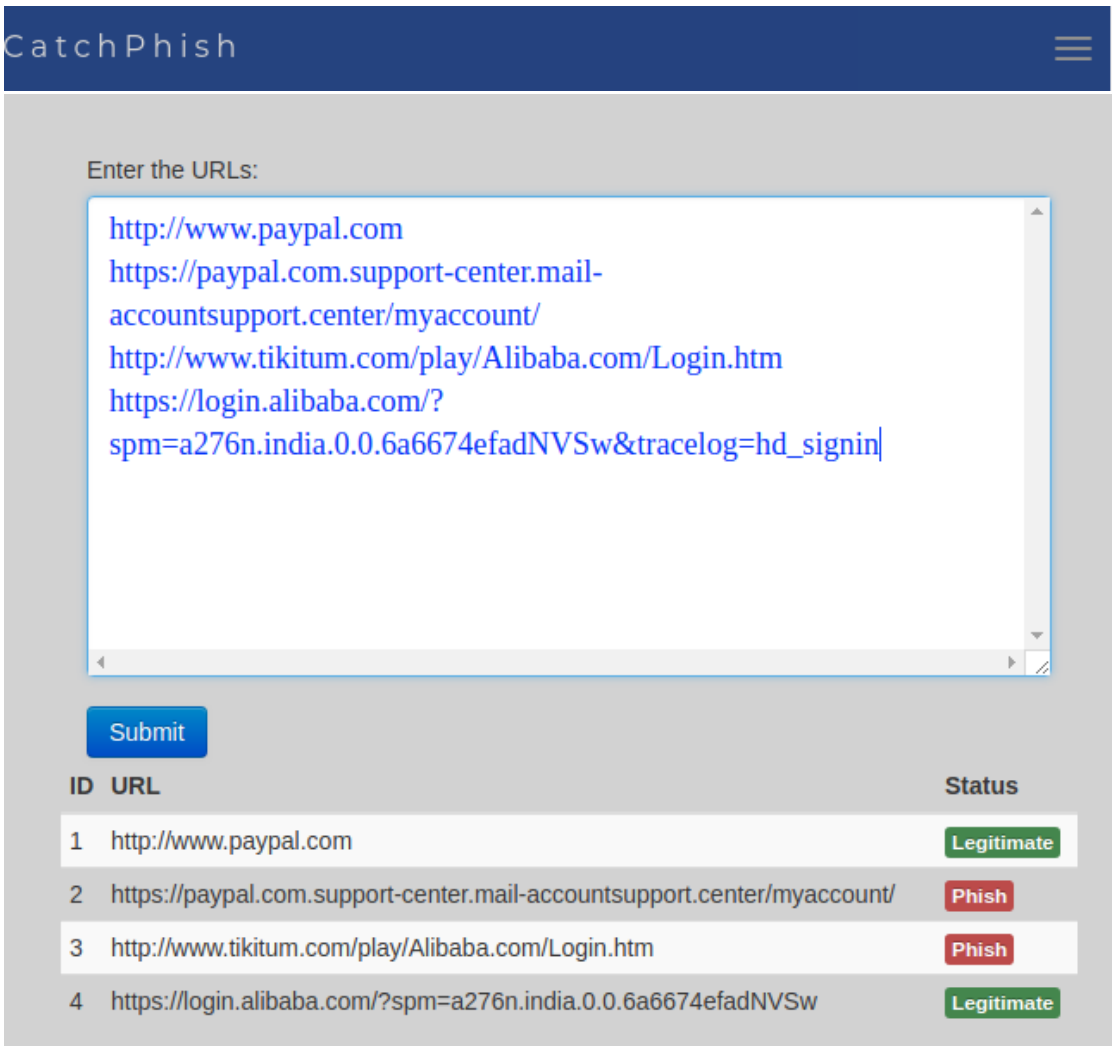


Figure 5.2: User interface of CatchPhish

is used for hosting REST API. Input URLs are sent as parameters to the REST API which are obtained by the application. This is followed by extraction of hand-crafted, TF-IDF features and they are fed to Random Forest classifier. This whole process of execution took different response times based on different input size of URLs. We calculated the average response times for various input size URLs and results are given in Table 5.11. The execution time for one URL input perfomed the best with 0.68 seconds average response time. We also observed a slight rise in average response time with increase in input size of URLs. The average response time for the execution of 10000 URLs is the highest among others with 124 sec. Note that, these response times are calculated with an average of 10 iterations for each input size. As we used TF-IDF

Table 5.11: Average response time for different input size

Input Size	Average Response time (secs)
1	0.68
10	0.86
100	1.91
1000	11.70
10000	124.20

Table 5.12: Accuracy of individual features

Features	Accuracy (%)	Features	Accuracy (%)	Features	Accuracy (%)
H1	67.74	H13	68.03	F9	71.97
H2	68.70	H14	68.03	F10	71.88
H3	67.96	H15	67.74	F11	67.78
H4	68.01	H16	67.75	F12	67.76
H5	68.46	F1	69.43	F13	67.76
H6	68.78	F2	71.19	F14	67.78
H7	67.74	F3	69.81	F15	67.74
H8	68.26	F4	74.20	F16	67.85
H9	68.30	F5	67.89	F17	67.77
H10	68.19	F6	77.10	F18	67.74
H11	67.78	F7	68.61	F19	67.74
H12	67.75	F8	68.46		

algorithm on a large dataset, we also attempted to calculate the time taken for training our model for the URL classification. The total time taken for training both handcrafted and TF-IDF features is 102 minutes however the detection time for the URL classification is only 0.68 sec (1 URL). Detection time is dependent on various factors such as speed of the Internet, server configuration. Bandwidth of 100mbps is used during our experimentation.

Even though the detection accuracy of our model with TF-IDF features is greater than hand-crafted features, we conducted an experiment on CatchPhish with each feature to investigate the importance of individual handcrafted feature for the URL classification. The results of the experiment is given in Table 5.12. From the table, it is observed that the accuracies of the proposed features (individually) are comparable with the existing features. Since the features are extracted only from the URL, it pro-

vides limited information to analyze and detect phishing sites less significantly. The individual accuracies do not prove the efficiency of the presented model but the collective features contribute for the higher accuracy (90.28%) of the model. To evaluate the impact of our newly proposed features H6, F6, F8-F9, F11-F18, we have performed two experiments. One with only existing hand-crafted features on D3 and another with both existing, newly proposed features on D3. In these experiments, RF classifier is used and results are shown in Figure 5.3. From the experimental results, it is observed that addition of our new features has increased the accuracy by 2.17%. The proposed features compliment the existing features in achieving better accuracy collectively.

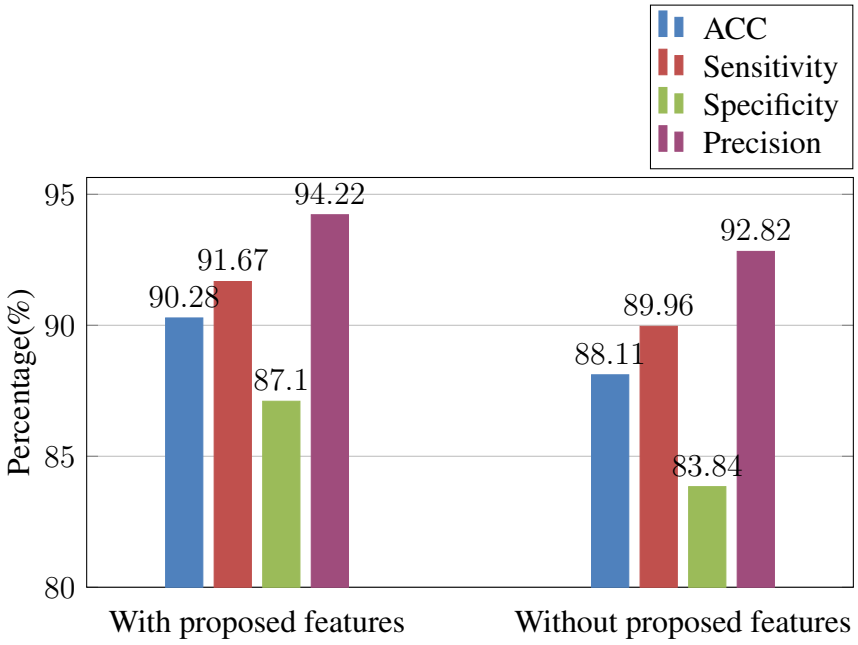


Figure 5.3: Performance of CatchPhish with the inclusion and the exclusion of proposed features

5.3 PHISHDUMP : A MULTI-MODEL ENSEMBLE BASED TECHNIQUE FOR THE DETECTION OF PHISHING SITES IN MOBILE DEVICES

Recent years have witnessed rapid growth in deep learning applications such as Health care, Image recognition, Automatic text generation, automatic coloring, prediction of natural calamities, traffic identification etc. Apart from these applications, deep learning has been widely used for text classification (Agarwal et al. 2018; Jiang et al. 2018), speech recognition (Chen and Mak 2015) and image classification (Gómez-Ríos et al.

2019; Ni et al. 2018). Recently, deep learning has also been used in various techniques (Le et al. 2018; Selvaganapathy et al. 2018) for the classification of malicious sites.

Many machine learning algorithms such as Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF) were used for the classification of suspicious sites. It is also observed that not many anti-phishing techniques used deep learning techniques for the detection of phishing sites in mobile devices. Hence, in our work, we chose an ensemble of both machine learning (SVM) and deep learning i.e. Long Short-Term Memory (LSTM) for the detection of phishing sites efficiently.

As there are no efficient techniques for the detection of phishing sites in mobile devices and since limited work is being done in this area, there is a need for a light-weight model to mitigate such attacks. In this section, we have presented a URL-based model to detect phishing sites in mobile devices and following are the contributions of our technique.

1. We have proposed a multi-model ensemble of LSTM and SVM to predict the legitimacy of the suspicious site and demonstrated its effectiveness as superior to other existing machine learning and deep learning techniques.

2. We conducted experiments on various datasets of different sizes (10K, 20K, 30K, 40K and 331K) to evaluate the behavior of the proposed model.

3. We implemented a real-time technique deployed as an android application named PhishDump to detect phishing sites in mobile devices.

To the best of our knowledge, there is no multi-model ensemble of machine learning and deep learning algorithms for the detection of phishing sites in the literature.

In recent years, the number of mobile users accessing web has increased which lead to a rise in the number of attacks in mobile devices. Existing techniques designed for desktop computers may not be suitable for mobile devices due to their hardware limitations such as RAM, screen size, low computational power etc. In this section, we propose a mobile application named as PhishDump to classify the legitimate and

phishing websites in mobile devices. PhishDump is based on multi-model ensemble of LSTM and SVM classifier. As PhishDump focuses on extraction of features from URL, it has several advantages over existing works such as fast computation, language independence and robust to accidental download of malwares. From the experimental analysis, we observed that our multi-model ensemble outperformed traditional LSTM character and word-level models.

5.3.1 Methodology

The main intention of our work is to develop a multi-model ensemble of machine learning and deep learning algorithms. A multi-model ensemble is a technique in which the features extracted from LSTM models at first-stage are fed as input to the second-stage SVM classifiers. These second-stage classifiers are trained and their combined predictions are considered to form a final set of predictions.

A. Long Short-Term Memory

Design of Neural network (NN) is inspired by the human brain and each neural network is composed of neurons. A neuron is a basic unit of NN which holds a number between 0 and 1. The value in the neuron corresponds to an activation. Activation in one layer determines activation of the next layer. A typical Deep Neural Network (DNN) is widely used for classification problems and it consists of mainly 3 types of layers: an input layer, one or more hidden layers and an output layer. Each layer in the network gets the connection from the previous layer and sends it to the next layer. It is the input layer which obtains the initial data and gives it to the subsequent hidden layers for further processing of the data. The output layer results in one of three or more classes in case of multi-class classification whereas one of two classes in case of binary classification. Recurrent Neural Network is one of the deep learning models which is capable of learning sequential patterns from text.

In the traditional Recurrent Neural Network (RNN), the error (neuron output) flowing back with time tend to gradually vanish preventing weights being updated and this problem is well-known as the vanishing-gradient problem. Long Short-Term Memory (LSTM) has been designed to address vanishing-gradient problem (Hochreiter and

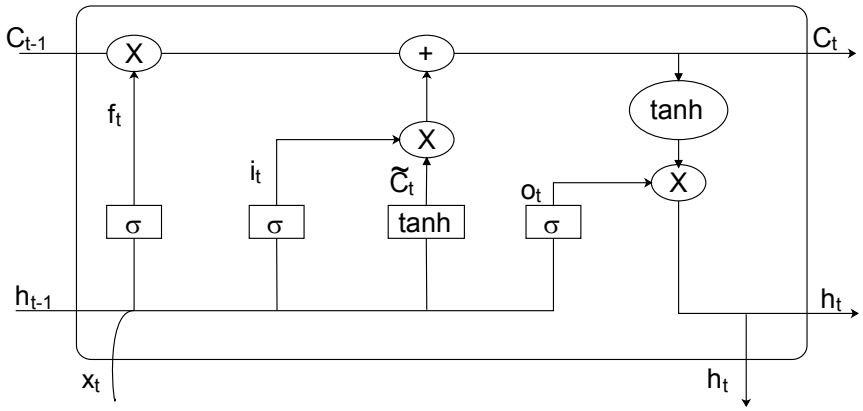


Figure 5.4: Architecture of LSTM Cell

Schmidhuber 1997).

Long Short-Term Memory is a variant of RNN architecture with a gradient-based algorithm designed for constant error backpropagation. Additional units were introduced apart from the standard units in LSTM to preserve long-term dependencies and it resulted in a complex structure called a memory cell. Each memory cell consists of input, output and memory. LSTM uses mainly three gates i.e. input, output and forget gates to control the cell state. Input gate uses sigmoid function which outputs values between 0 and 1 indicating the amount of each component to be let into the cell state. The input modulation gate $\tilde{C}_t$ uses Tanh activation function which outputs values between -1 and 1 and it helps to forget the memory.

Forget gate decides the information to be thrown away from a cell state and Equation (5.4) gives the forget gate activation. This gate takes inputs from previous hidden layer state h_{t-1} and LSTM input x_t at time t and outputs the value between 0 and 1 for each component in the cell state C_{t-1}.

$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f) \tag{5.4}$$

Input gate defined in Equation (5.5) decides the amount of input x_t to be written to the current cell state $\tilde{C}_t$. Both input and forget gates together decide what information

to be rewritten at each step.

$$\begin{cases} i_t = \sigma(W_i[x_t, h_{t-1}] + b_i) \\ \tilde{C}_t = tanh(W_c[x_t, h_{t-1}] + b_c) \end{cases} \tag{5.5}$$

Input gate decides the values to be updated to store the new information into the cell state. Tanh layer creates a vector C which needs to be added to reset the information from old subject's name to new subject's name.

Output gate decides the amount of activation to be output from the hidden layers and also helps the memory cell to store the information which is irrelevant at present but may be relevant in future. Output gate updates the old cell state C_{t-1} to new state C_t using Equation (5.6). The output is a cell state and a sigmoid function is responsible to decide what parts of the cell state will be shown as output. N neurons sharing same input, forget and output gates is a memory cell block of size N.

$$\begin{cases} C_t = i_t * \tilde{C}_t + f_t C_{t-1} \\ O_t = \sigma(W_o[x_t, h_{t-1}] + b_o) \\ h_t = O_t tanh(C_t) \end{cases} \tag{5.6}$$

The working of LSTM is presented in Algorithm 5.1 and the process of updating LSTM cell is given in Algorithm 5.2. Algorithm 5.1 takes suspicious URL as input and gives the status of URL either legitimate or phishing.

B. Support Vector Machine

SVM is a machine learning algorithm used for data classification and regression problems by defining a hyperplane to differentiate the data points corresponding to different classes. For a given training set, SVM defines optimal hyperplane so as to maximize the margin of training set for the correct classification of the new dataset.

Given a training set $T = \{(x_i, y_i) : x_i \in \mathbb{R}^n \text{ and } y \in \{-1, 1\},\ \ i = 1, 2, ..., l\}$, SVM finds the optimal separating hyperplanes $w^T\phi(x) + b = 0$ in feature space where $\phi : \mathbb{R}^n \rightarrow \mathcal{H}$ maps data points in the higher dimensional feature space $\mathcal{H}$. This separating

Algorithm 5.1: LSTM algorithm

Input: URL of the webpage
Output: Class of the webpage

$char\_seq \leftarrow Textpreprocessing(texttosequences)$
$ct, ht \leftarrow [0,0,0], [0,0,0]$
for each $char \in char\_seq$ **do**
    $ct, ht \leftarrow LSTMCELL(ct, ht, char)$
end for
if $ct = 0$ **then**
    $class \leftarrow Legitimate$
else
    $class \leftarrow Phishing$
end if
return $class$

Algorithm 5.2: LSTMCELL algorithm

Input: prev_ct, prev_ht, input
Output: ct, ht

$concat \leftarrow prev\_ct + input$
$ft \leftarrow forget\_gate(concat)$
$candidate \leftarrow candidate\_gate(concat)$
$it \leftarrow input\_gate(concat)$
$ct \leftarrow prev\_ct * ft + candidate * it$
$ot \leftarrow output\_gate(concat)$
$ht \leftarrow ot + tanh(ct)$
return ct, ht

hyperplanes has been obtained by solving following quadratic prime problem (QPP).

$$\min_{(w,b,\xi)} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i$$

subject to,

$$y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i \;\; i = 1, 2, .., l,$$

$$\xi_i \geq 0, \;\; i = 1, 2, .., l. \tag{5.7}$$

where $C \geq 0$ controls the trade-off between the margin width $\frac{1}{2} w^T w$ and the minimization of the training error. Using the Karush-Kuhn-Tucker (KKT) conditions, the dual problem of the QPP (5.7) has been obtained as follows

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \phi(x_i)^T \phi(x_j) \alpha_i \alpha_j - \sum_{i=1}^{l} \alpha_i$$

subject to,

$$\sum_{i=1}^{l} \alpha_i y_i = 0,$$

$$0 \leq \alpha_i \leq C, \;\; i = 1, 2, ..., l. \tag{5.8}$$

Using the Mercer condition (Mercer 1909; Vapnik and Vapnik 1998), it is possible to use a kernel function $K(x_i, x_j)$ to represent the inner product $\phi(x_i)^T \phi(x_j)$ in the feature space $\mathcal{H}$.

After obtaining the optimal values of w and b from (5.8), a test point $x \in \mathbb{R}^n$ has been classified using the decision function

$$f(x) = sign(w^T x + b). \tag{5.9}$$

In our model, SVM is used in the multi-model ensemble for the classification of suspicious URLs based on LSTM features.

C. LSTM-SVM

The architecture of the PhishDump is divided into two stages as shown in Figure 5.5. The first stage of the multi-model involves training and testing of the LSTM model, generation of top models which are used in the second stage of the detection process. In the second stage, the features are extracted from the top models and fed to individual base classifiers for the classification of suspicious URL. The components of the first stage are given below.

- **LSTM model:** The deep neural networks not only outperforms state-of-the-art machine learning methods but also comes with the benefit of omitting difficult feature engineering (Lample et al. 2016; Ma and Hovy 2016; Wu et al. 2018). On the other side, traditional methods such as Random Forest, XGBoost, Logistic Regression, and Support Vector Machine etc. takes input as hand crafted features and thereby uses feature selection algorithms to achieve better results with

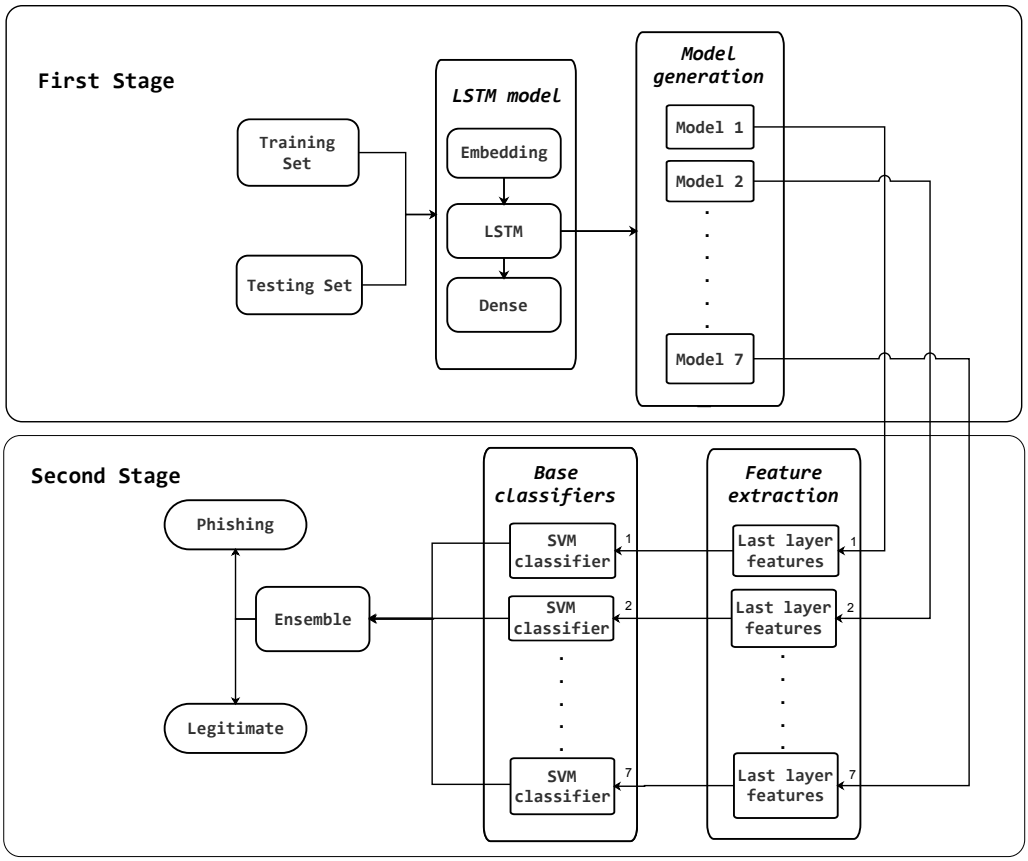


Figure 5.5: Architecture of the multi-model ensemble

less number of relevant features. Hence, in this work, we have used LSTM algorithm which is most widely used for the sequential data like URLs to detect the legitimacy of the suspicious site(Bahnsen et al. 2017; Chen et al. 2018; Yu et al. 2018).

This model takes a set of legitimate and phishing URLs as input and each URL is tokenized into characters which are fed to the embedding layer of LSTM model. The output of the embedding layer is fed to LSTM hidden layer with 100 units. This layer is followed by a dropout layer to avoid overfitting. From the experimental analysis, we observed that dropout of 0.2 is optimal for our dataset. The output of the LSTM model is obtained from dense layer with sigmoid activation. One entire pass of the training set through LSTM network corresponds to one epoch and many such epochs are considered to achieve an acceptable level of accuracy.

- **Model generation:** An LSTM model with respect to each epoch is saved and validation accuracies are considered as a basis to identify the top most models. Out of all the saved LSTM models, top `n` models are considered as input to the second stage.

The components of the second stage are given below.

- **Feature extraction:** In this process, features are extracted from the LSTM hidden layer of the top n models generated in the first stage of the detection process. We obtain feature set corresponding to each top n LSTM models which are used by the next component.
- **Base classifiers:** Aforementioned LSTM features can be fed to any machine learning classifier to predict the legitimacy of the suspicious URL. From the experimental analysis, we observed that SVM outperformed other classifiers with respect to LSTM features. Hence, we used SVM with Radial Basis Function (RBF) kernel as a base classifier in our multi-model to make predictions for the suspicious URLs.
- **Ensemble:** The predictions of each SVM classifier are combined to make the final prediction on suspicious URL. Majority voting strategy is used to combine the predictions of each base classifier to make the final decision on the legitimacy of the URL. In majority voting algorithm, the final class label is the class predicted by majority of the classifiers in the ensemble model. The rationale behind choosing the ensemble is that aggregation of base models into combined model outperforms every single model in it (Dietterich 2000; Ren et al. 2016).

5.3.2 Experimentation and Results

A. Dataset

We have collected the dataset from three different sources i.e, PhishTank, OpenPhish and Alexa as shown in Table 5.13. Legitimate sites are obtained from top 100 Google search results for each brandname given as search query. These brandnames are collected from PhishTank (206 brandnames are selected) and Alexa database (top 2000 out

Table 5.13: Dataset used for experimentation

Type	Source	Sites	URL
Legitimate	PhishTank	19510	http://phishtank.com/target_search.php
	Alexa database	170552	http://www.alexa.com/topsites
Phishing	Open Phish	7212	https://openphish.com/
	PhishTank	134278	http://www.phishtank.com/developer

of 1 million brandnames). The legitimate sites are preprocessed to remove duplicate sites. Phishing sites are obtained from PhishTank. We have collected 190062 legitimate sites and 141490 phishing sites which are divided into 5 subsets as shown below:

- *D1:* 5K legitimate sites and 5K phishing sites
- *D2:* 10K legitimate sites and 10K phishing sites
- *D3:* 15K legitimate sites and 15K phishing sites
- *D4:* 20K legitimate sites and 20K phishing sites
- *D5:* 190062 legitimate sites and 141490 phishing sites

A python program has been implemented to extract features from LSTM character-level model and these features are fed as input to the SVM for URL classification. Keras is a python package used to implement deep learning algorithms whereas machine learning algorithms are implemented using Scikit-learn.

B. Experimental evaluation

To evaluate the performance of our model, we have used traditional metrics such as Sensitivity (TPR), Specificity (TNR), FPR, FNR, Accuracy (Acc), Pre, F-measure (F). From the existing literature of deep learning, LSTM is more frequently used for text and URL classification. Hence in our model, we have used LSTM as classifier at the first stage. In Experiment 1, we attempted to identify which among the LSTM character and word-level models is suitable for our dataset. Experiment 2 identifies the best machine learning classifier used in second-stage. Experiment 3 calculates the threshold for choosing the top LSTM models resulted in experiment 1. We have also evaluated

Table 5.14: Evaluation of LSTM word-level and character-level models

Method	EV	BS	Metrics (%)						
			TNR	FPR	TPR	FNR	Precision	Accuracy	F1-Score
LSTM word-level	64	64	93.91	6.09	96.91	3.09	95.54	95.63	96.22
	64	128	93.95	6.05	96.89	3.11	95.57	95.64	96.23
	128	64	94.40	5.60	96.74	3.26	95.88	**95.74**	96.31
	128	128	94.30	5.70	96.74	3.26	95.81	95.70	96.27
LSTM character-level	64	64	97.01	2.99	96.68	3.32	97.81	96.82	97.24
	64	128	96.78	3.22	96.86	3.14	97.63	96.83	97.24
	128	64	97.27	2.73	96.76	3.24	98.01	**96.97**	97.38
	128	128	97.05	2.95	96.84	3.16	97.83	96.93	97.33

our PhishDump on various datasets with different size in experiment 4. Finally, the PhishDump is compared with the existing techniques in experiment 5.

Experiment 1: Comparison of LSTM character-level and word-level models

In this section, we compare the LSTM character and word-level models to identify the better model for our PhishDump. Note that the experiments on both the models were carried out on large dataset D5 with various combination of embedding vector and batch size. From the experimental results shown in Table 5.14, LSTM word-level model achieved an accuracy of 95.74% whereas LSTM character-level model achieved an accuracy of 96.97% with respect to 128 embedding vector (`EV`) and batch size (`BS`) of 64. We have considered `EV`= (64, 128) and `BS`= (64, 128). As `EV`=128, `BS`=64 outperformed the other possible combinations, so we use these parameters in the following experiments. It is observed that for each combination of EV and BS, the accuracy achieved by the character-level model is greater than the corresponding accuracy of the word-level model.

Experiment 2: Choosing the best ML classifier

In this experiment, we have fed the LSTM intermediate layer features obtained from experiment 1 to various classifiers such as Support Vector Machine (SVM), Logistic Regression (LR), Naive Bayesian (NB), XGBoost and Random Forest (RF). As mentioned in the above experiment, `EV`=128 and `BS`=64 performed better and hence, we used the same parameter to identify the best ML classifier. Note that we have tested dif-

Table 5.15: Evaluation of ML classifiers with LSTM features

Classifier	Metrics (%)						
	TNR	**FPR**	**TPR**	**FNR**	**Precision**	**Accuracy**	**F1 Score**
XGBooost	96.57	3.43	97.02	2.98	97.46	96.83	97.24
RF	96.76	3.24	97.05	2.95	97.61	96.93	97.33
NB	96.20	3.80	97.06	2.94	97.17	96.70	97.12
LR	96.70	3.30	97.13	2.87	97.56	96.95	97.34
KNN	96.83	3.17	97.03	2.97	97.66	96.95	97.35
Gradient Boosting	96.51	3.49	97.04	2.96	97.42	96.81	97.23
SVM	96.85	3.15	97.08	2.92	97.68	**96.98**	97.38

ferent classifiers with the same dataset D5 used in experiment 1. From the experimental results shown in Table 5.15, it is observed that the SVM outperformed the other classifiers with an accuracy of 96.98%. Hence, SVM classifier is chosen in our multi-model ensemble.

Experiment 3: Threshold for choosing the top LSTM models

This experiment calculates the threshold for top LSTM models used in second-stage of the detection process. The ensemble classifier with max voting combiner algorithm is used during the second-stage detection and the possible values for the number of models (`n`) in combiner algorithms are 3, 5, 7, 9 and 11. The results of the experiment with varying `n` is given in Figure 5.6. From the experimental results, it is observed that `n`=7 outperformed the other possible values. For the remaining experiments, we use LSTM character level model, SVM classifier with threshold `n`=7 for the first stage and second stage classification respectively.

Experiment 4: Evaluation of multi-model ensemble (LSTM+SVM)

In this experiment, we have extracted features from LSTM models corresponding to top 7 (`n`=7) validation accuracies and these features are fed to SVM. The ensemble of these seven SVM models is considered for the classification of URLs. We have considered 100 LSTM units, a learning rate of 0.001, sigmoid activation, and 30 epochs in this experiment. Unlike other experiments, we have conducted this experiment on different dataset with varying size (D1-D5) to evaluate effectiveness of the multi-model ensemble. From the experimental results shown in Table 5.16, PhishDump achieved an

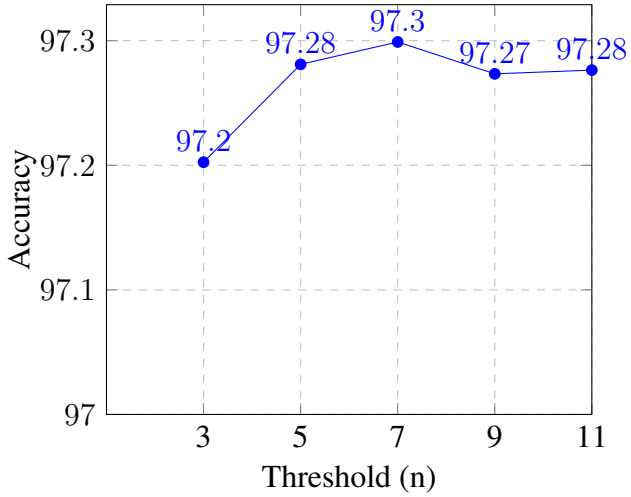


Figure 5.6: Calculation of Threshold for top LSTM models

Table 5.16: Evaluation of multi-model ensemble on D1-D5

Dataset	Top Model	LSTM-SVM Metrics (%)						
	Accuracy (%)	TNR	FPR	TPR	FNR	Precision	Accuracy	F1 Score
D1	95.25	95.58	4.42	95.21	4.79	95.40	95.40	95.31
D2	96.47	96.76	3.24	96.69	3.31	96.74	96.73	96.71
D3	96.35	96.64	3.36	96.20	3.80	96.62	96.42	96.41
D4	96.91	97.14	2.86	96.93	3.07	97.20	97.04	97.07
D5	96.97	97.28	2.72	97.31	2.69	97.99	**97.30**	97.65

accuracy of 97.30% for D5 which is higher than the accuracies of ensemble models with respect to other datasets. The results also demonstrated that ensemble outperformed all the individual models of datasets D1-D5 with respect to `BS`=64 and `EV`=128.

Experiment 5: Comparison of PhishDump with existing techniques

We compared our PhishDump work with existing works that use deep learning and machine learning algorithms for the detection of phishing URLs. We implemented the Bahnsen et al. (2017), Patgiri et al. (2019) works and applied the same dataset D5 used for our PhishDump work. We have also applied our multi-model ensemble on the dataset collected from Sahingoz et al. (2019) for comparison with their work and we term the dataset as D6. For simplicity, we term Bahnsen et al. work as W1, Sahingoz et al. work as W3, Patgiri et al. work as W4, our work with D5 as W2 and our work with D6 dataset is termed as $W2^{'}$. The reason behind choosing W1, W3 and W4 for

Table 5.17: Comparison of our work with existing techniques

Metrics(%)	W1 (Bahnsen et al. 2017)	W2 (Our Work (D5))	W2′ (Our Work (D6))	W3 (Sahingoz et al. 2019)	W4 (Patgiri et al. 2019)
Precision	96.81	97.99	98.17	97.00	84.99
Specificity	95.67	97.28	98.20	96.92	88.94
FPR	4.33	2.72	1.80	3.08	11.06
FNR	3.82	2.69	1.20	1.00	15.24
Sensitivity	96.18	97.31	98.80	99.00	84.76
F- measure	96.49	97.65	98.49	98.00	84.87
Accuracy	95.96	**97.30**	**98.50**	97.98	87.16

comparison is the similarity in using deep learning and machine learning algorithms for the URL classification.

From the experimental results shown in Table 5.17, it is observed that our presented multi-model W2 outperformed the existing works W1, W4 with an accuracy of 97.30%, TPR of 97.31% and TNR of 97.28%. Also, our work $W2^{'}$ performed better than the existing work W3 with an accuracy of 98.50%. It is also observed that sensitivity of W3 is 0.20% greater than that of $W2^{'}$ due to the use of third-party services.

We have also plotted the Receiver Operating Characteristics (ROC) curves for the existing works (W1, W4) and compared with our work as shown in Figure 5.7. The ROC gives the relative tradeoffs between costs (false positives) and benefits (true positives) . The area under the ROC curve (AUC) measures the area underneath the entire ROC curve. The closer the AUC of a model towards 1, the better it is. In Figure 5.7, PhishDump achieved the AUC of 0.998 which is much closer to 1 compared to the existing works. From the experimental observation, it is observed that our work outperformed existing works W1, W3 and W4 on different datasets with significant accuracy, F-measure and AUC. This shows that the multi-model ensemble of ML and DL has a significant influence on the URL classification than the individual ML and DL based techniques.

5.3.3 Discussion

Real-time protection against phishing sites in mobile devices is the main idea of the presented multi-model ensemble. It is required to have a light-weight mechanism for the early detection of phishing sites. Hence, we implemented PhishDump as an android

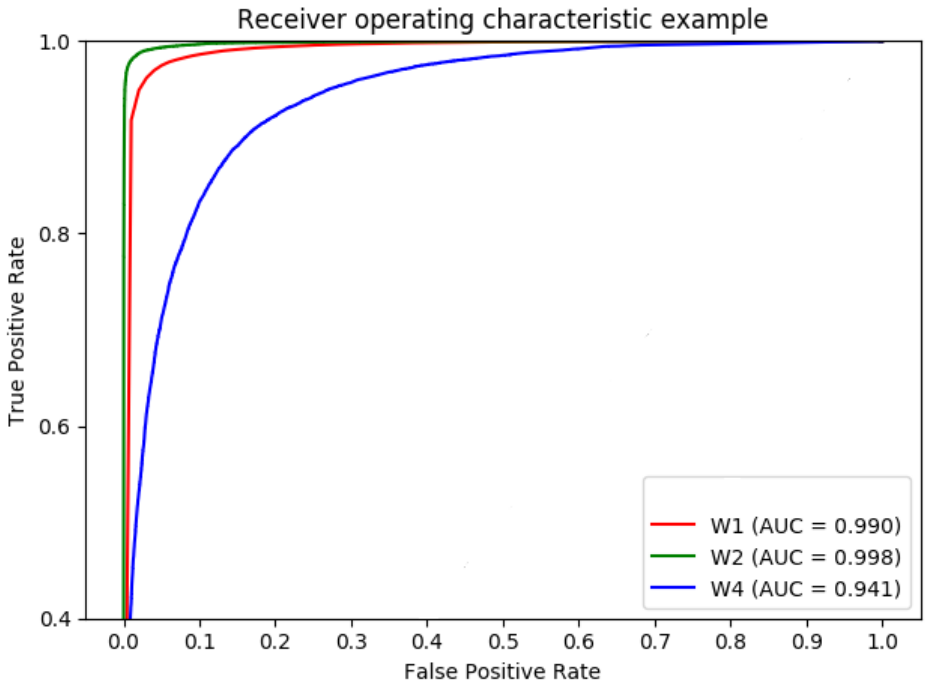
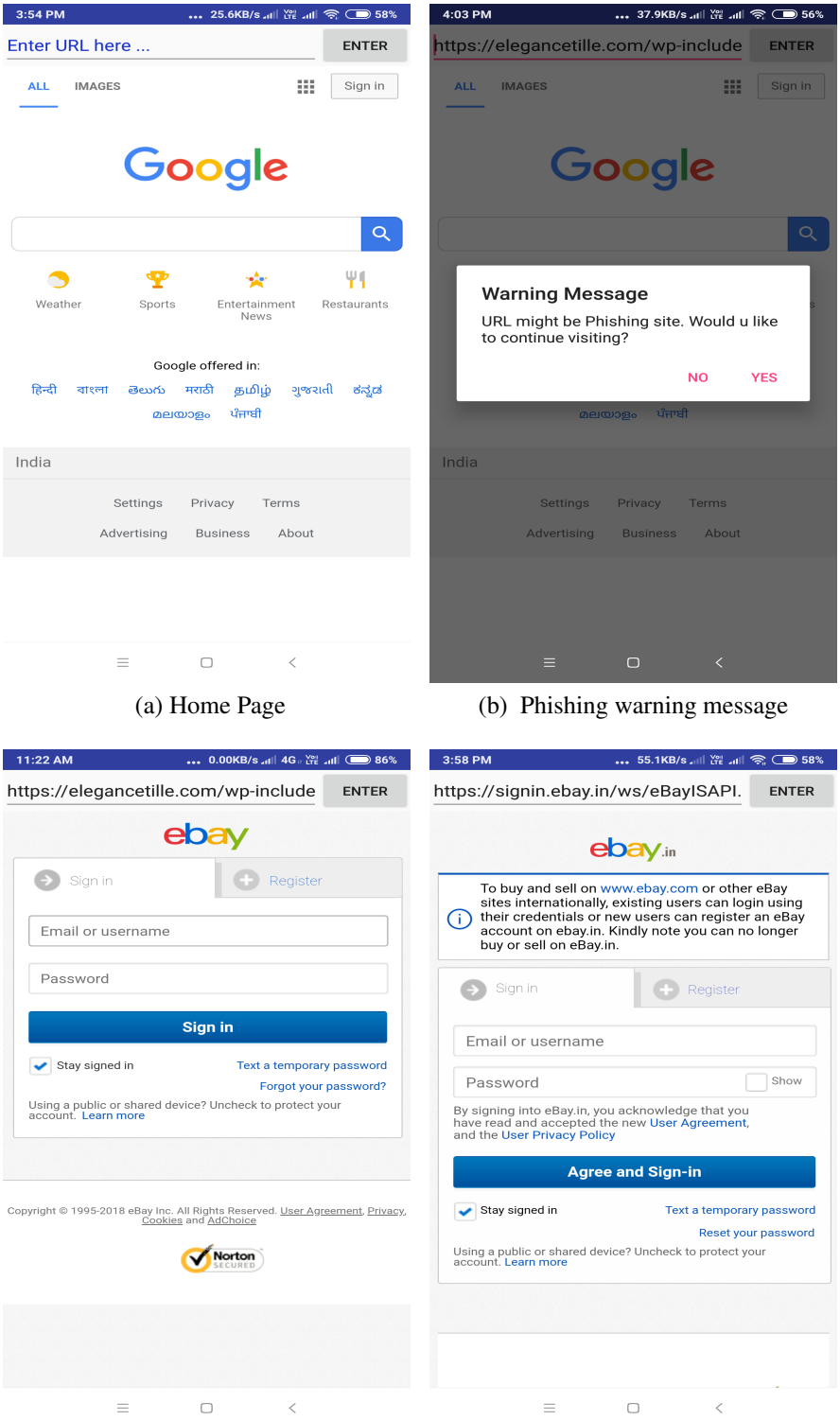


Figure 5.7: ROC of PhishDump with existing works

application on Google Nexus 5 running Android 5.1 (Lollipop) operating system. We have implemented a REST API which takes URL as an input and gives the status of the URL as an output. The API is implemented using Flask, a microframework for python and is run on Ubuntu system with Intel Xeon 16 core, 2.67GHz processor and 16GB RAM. We used GET method to transmit the URL status to the application. The User Interface (UI) of the application is shown in Figure 5.8(a). The UI consists of input field to insert the URL along with Google Search page as a default web view. Once the status of the URL is revealed, phishing URL is blocked with a warning message shown in Figure 5.8(b) if ignored, the user is redirected to phishing site as shown in Figure 5.8(c). If the status is legitimate, then the URL is rendered in the web view shown in Figure 5.8(d). To evaluate the real-time adaptability of our application, we have calculated the response time and it is observed that PhishDump takes an average time of 621 ms to complete the detection process. For achieving the low response time, we have parallelized the accessing of seven models with the help of multiple cores in the system. The output labels from the respective seven models are ensembled to return the final output as either legitimate or phishing. Though our model is based on URL



(a) Home Page (b) Phishing warning message

(c) Phishing Page (d) Legitimate Page

Figure 5.8: UI of PhishDump

Table 5.18: Comparison of our work with existing techniques

Technique	Response Time
Dunlop et al. (2010)	4310 ms
Huh and Kim (2011)	500 ms
Varshney et al. (2016b)	1530 ms
Amrutkar et al. (2017)	829 ms
Zuhair et al. (2016)	3100 ms
Ma et al. (2009)	3580 ms
PhishDump	621 ms

rather than source code, it still depends on few factors such as speed of the Internet, RAM in the mobile device. In our experiments, we used a bandwidth of 6 Mbps and RAM of 4GB.

We have compared the response times of existing techniques with PhishDump shown in Table 5.18. From the table, it is observed that PhishDump has faster detection rate compared to existing works except Huh and Kim (2011) work. But it should be noted that, Huh et al work is dependent on third party service(search engine results) which makes the technique fail when the service is unavailable. We argue that this improvement in low response time makes the PhishDump more suitable for mobile devices in real time.

Our multi-model ensemble used DL and ML classifiers and it is observed that ML and DL algorithms have both pros and cons. Though DL algorithms perform better without prior knowledge to generate features, it is difficult to interpret the actual working of the DL based models. In case of ML classifiers, it is relatively easy to understand the inner working of the classifiers but prior knowledge is required to input relevant features. The main intention of the presented multi-model ensemble is to increase the performance by using both ML and DL classifiers. Also, we observed that multi-model ensemble of ML and DL performed better than the existing techniques which used ML and DL algorithms individually for the URL classification. PhishDump achieved satisfactory results in phishing detection, but there is a possibility that the attacker might bypass the technique by making the structural changes to the URL such that it appears

as a legitimate URL. Also, the technique might fail to detect phishing sites with short URLs (bitly, goo, tiny etc).

5.4 SUMMARY

In this chapter, we have presented two techniques (CatchPhish, PhishDump) based on URL features for the detection of phishing sites. The first technique CatchPhish uses a hybrid feature set containing both manually crafted URL features and TF-IDF based features. In this method, phish-hinted blackwords are extracted from the phishing URLs and the Random Forest classifier is used for the classification of phishing and legitimate sites. The CatchPhish application achieved an accuracy of 94.26% on our dataset and an accuracy of 98.25% on a benchmark dataset.

Secondly, we presented a URL based multi-model ensemble of LSTM and SVM (PhishDump) for the classification of suspicious sites in mobile devices. We also compared our model with the existing baseline models on different datasets. PhishDump outperformed existing works with an accuracy of 97.30% on our dataset and 98.50% on the benchmark dataset.

The main advantage of these techniques is that they are independent of third-party, source code and pre-stored database. These advantages lead to the detection of phishing sites with a low response time and free of drive-by downloads.

In the earlier chapters, we have presented various phishing methods using either URL, source code and search engine results. But these techniques do not have the feature of showing the target legitimate site. The main advantage of alerting the user with a target site is that it provides more trust on the protection tool or extension. To achieve the same, we present a visual similarity based approach which detects the phishing sites along with its target domain in the next chapter. To reduce the detection latency in visual similarity approach, we have presented enhanced blacklist mechanism which uses fingerprints to represent the website.

CHAPTER 6

A LIGHTWEIGHT VISUAL SIMILARITY BASED APPROACH FOR THE DETECTION OF PHISHING SITES

The visual similarity-based techniques detect the phishing sites based on the similarity between the suspicious site and the existing database of resources such as screenshots, styles, logos, favicons etc. These techniques fail to detect phishing sites which target non-whitelisted legitimate domain or when phishing site with manipulated whitelisted legitimate content is encountered. Also, these techniques are not well adaptable at the client-side due to their computation and space complexity. Thus there is a need for light weight visual similarity-based technique detecting phishing sites targeting non-whitelisted legitimate resources. In this chapter, unlike traditional visual similarity-based techniques using whitelists, we employed a light-weight visual similarity based blacklist approach as a first level filter for the detection of near duplicate phishing sites. For the non-blacklisted phishing sites, we have incorporated a heuristic mechanism as a second level filter. The phishing sites which bypassed from the first level filter undergo second level heuristic filtering

In the first level filtering, we used two fuzzy similarity measures, Simhash and Perceptual hash for calculating the similarity score between the suspicious site and existing blacklisted phishing sites. For the second level filtering, we used comprehensive heuristic features including URL and source code based features for the detection of non-blacklisted phishing sites. Finally, the ensemble model with Random Forest (RF),

Extra-Tree and XGBoost is presented to evaluate the contribution of both blacklist and heuristic filters together as an entity.

6.1 INTRODUCTION

The heuristic techniques depending on the textual or source code fails when the phishing site replaces entire content with an image. Also, the attackers embed the phishing content in iframes such that source code is hidden for the anti-phishing techniques.

To counter these kind of phishing sites, the visual similarity based approach is used. The technique detects the phishing sites based on the similarity score between the visiting site and pre-stored database of target legitimate resources. The attacker designs a phishing site which has high visual resemblance with respect to target legitimate site so that the online user is easily convinced. The visual resemblance includes pixels, font styles, images, screenshots, page layouts, logos, text content, and font colors etc. The visual based technique compares the above features with pre-stored database of legitimate site contents for the similarity score calculation. If the similarity score is above a certain threshold with domains mismatched then it is classified as phishing else classified as legitimate site. Usually, these techniques maintain single version of target legitimate site but the attackers evade these techniques by designing different versions of target brand. Also, these techniques generate false negatives when out of list legitimate sites are encountered.

In this chapter, we present a two-level anti-phishing technique to detect the variations of blacklisted sites at filter 1 and non-blacklisted sites at filter 2. We used similarity based features (attribute values of tags, anchor links, paths, scripts, images, styles, filenames) to address the limitations associated with traditional blacklists. We used various fingerprints for the blacklist database generation to address the limitations of existing visual similarity techniques. Though this approach is efficient in identifying near duplicate phishing sites, it fails to detect the phishing sites that are dissimilar to replicas of phishing sites. Note that, a web page is termed as near duplicate web page, if it copies the content of another webpage with slight changes. The content might include textual data or CSS Styles or Layout or DOM. In this work, we also presented a heuristic

mechanism as second level filter to detect the phishing sites which are bypassed from the blacklist filter. The advantage of presented blacklist approach is that it makes the phish less appealing or more difficult to design. Because, the newly designed phishing sites which are replicas or variations of existing phishing sites are easily detected and added to the blacklist database. To bypass the technique, the attacker has to make an additional effort and seek more knowledge in designing the phishing site either from scratch or by manipulating the DOM. Similarly, heuristic filter with rich features attempts to reduce the impact of phishing attacks.

Following are the contributions of our technique.

1. We propose a lightweight enhanced blacklist based visual similarity technique with noisy data and screenshots to detect near duplicate phishing sites.

2. We designed an ensemble model combining RF, XGBoost and Extra-Tree classifiers to increase the phishing sites detection accuracy.

3. We deployed our technique as a Chrome extension named `BlackPhish` to provide real time protection at client side and thereby prevent the users from accessing phishing sites.

4. The BlackPhish provides additional target legitimate site information when the phishing site is detected by the blacklist filter.

As mentioned earlier, the current visual similarity based techniques use logos or favicons or screenshots etc which results in higher delay in detection process making it not suitable at client side. Our BlackPhish work differs from the existing literature in two ways. Firstly, we have avoided maintaining database of legitimate resources due to the ever increasing legitimate dataset. Also, maintaining only target legitimate resources block non-whitelisted legitimate sites. Secondly, we are not maintaining list of URLs or images for the legitimacy detection. The avoidance of images made our technique faster and by skipping URLs, the technique is robust to out of list URLs.

6.2 METHODOLOGY

The main intention of this work is to identify the near duplicate phishing sites which are variations of blacklisted phishing sites. Also, the missed non blacklisted phishing sites are filtered using heuristic based features. The architecture of the proposed work is given in Figure 6.1. The BlackPhish model consists of two level filtering namely blacklist filtering and heuristic filtering. The input URL is classified as legitimate only when it passes through both the filters.

6.2.1 Blacklist filtering

In this technique, each of the blacklisted phishing sites is stored as fingerprints such that near duplicate zero-day phishing sites are detected. We have considered 5076 phishing sites (Phishtank) to generate the blacklist of fingerprints. The technique uses Perceptual hash, Simhash algorithms for generating the fingerprint with input as HTML tags, textual content and screenshot of website. Blacklist filtering mechanism consists of two components namely Feature extraction and Blacklist database generation - Replica detection. First component extracts the features from the HTML, plaintext or screenshot of the website which are further used for the generation of blacklist database. The flowchart of this blacklist filtering mechanism is given in Figure 6.2.

A. Feature Extraction

There exist many toolkits which consists of required CSS, scripts, images that are zipped together. These zip files are used by naive attackers to design phishing sites without any difficulty. This indicates that file path of the phishing sites designed using same toolkit may remain same. Many a times, there may be a slight change in file path structure of URLs but not in the actual filenames. Attackers may misspell or skip some text (copyright, title, metadata), tags in the phishing sites in order to avoid disclosure of actual identity of the site. But the attributes of the tags remain same in order to maintain visual similarity between phishing and target sites using same styles, images as that of legitimate sites. Some phishing sites use same textual content but maintain different URLs such that blacklist techniques are failed to detect. The attackers also manipulate the textual content by adding date or default values such that textual based

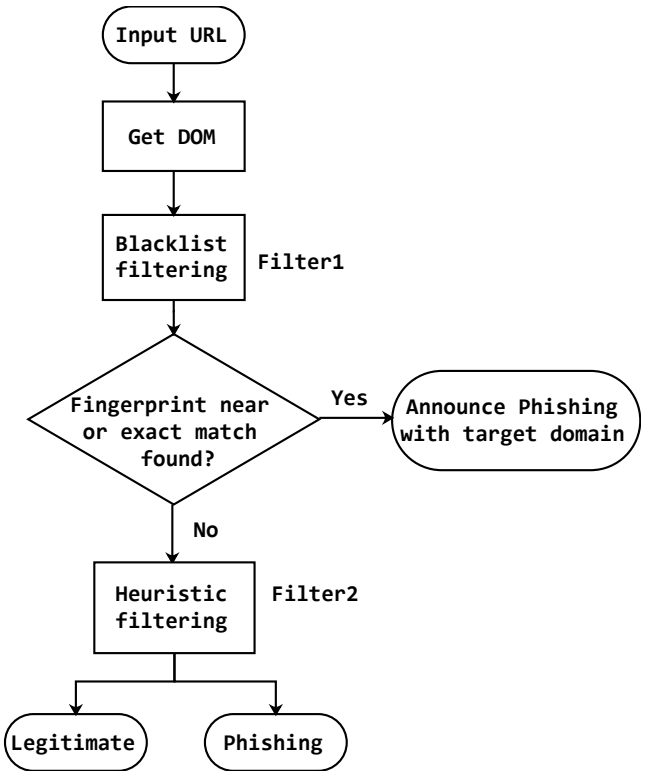


Figure 6.1: Flowchart of the BlackPhish

techniques using cryptographic hashes are bypassed. Finally, there exists some phishing sites which replaces textual content with an image such that textual based techniques are bypassed. Features are selected and extracted considering the observations mentioned above and these features are divided into 5 categories as given below.

F1 - Features based on attribute values: It is observed that most of the websites use div tags to define a block of website and also to group the blocks so as to apply styles on them. Since div tags exist in almost every website, it is considered for the feature extraction. We also observed that majority of the websites use heading tags h1 and h2 to define brandnames. Since style of body and form tags are unique, these elements are also considered for feature extraction. Class, id and name are some attributes which contribute to the action of elements. Hence these attribute values are extracted for div, h1, h2, body and form tags.

F2 - Features based on Pathnames: Many websites contain URLs requesting background images, styles and scripts. These request URLs can be anchor links, images,

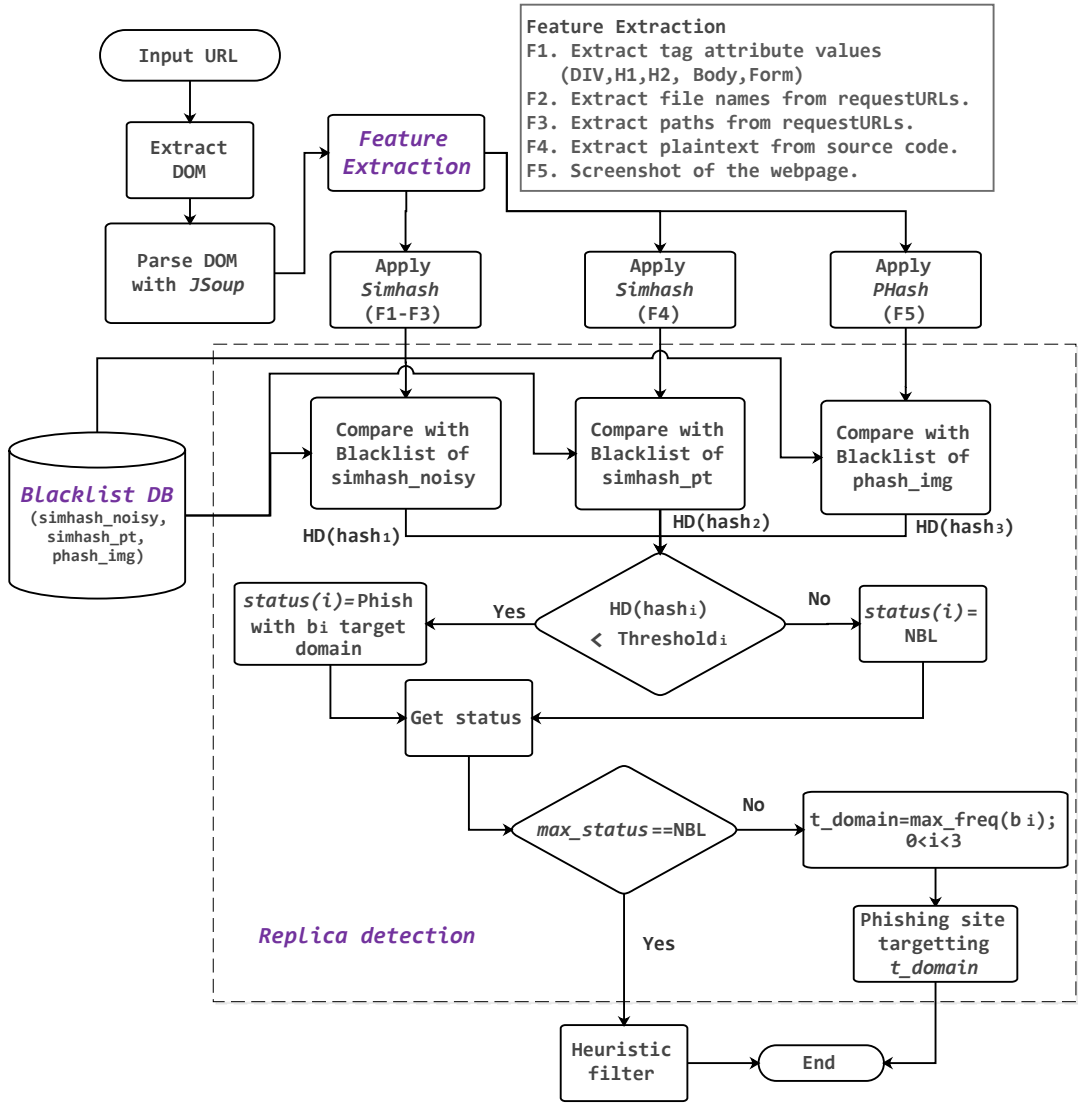


Figure 6.2: Flowchart of the Blacklist Filtering

scripts. The directory structure of requested URLs' pathnames are extracted from the source code in order to identify individuality of the websites.

F3 - Features based on Filenames: This feature extracts filenames of CSS, scripts and images from the request URLs of the website. These filenames are used to identify uniqueness of the website.

F4 - Feature based on plaintext: Unlike noisy data (F1-F3), this feature extracts plaintext from the given website using Jsoup parser. The Simhash algorithm is applied to this feature to generate fingerprint. The rationale behind choosing this feature is that the traditional textual based techniques (Xiang et al. 2011) used cryptographic hashes

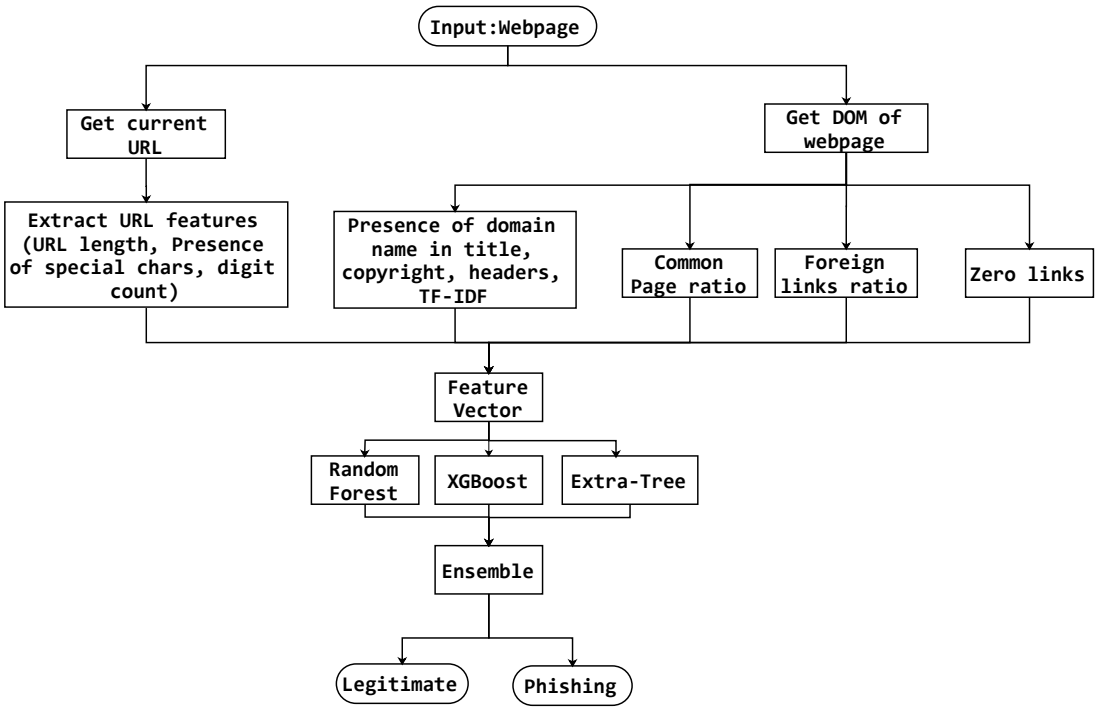


Figure 6.3: Flowchart of the Heuristic based Filtering

to detect the replicas of the existing phishing sites which fails when there is a small variation to the existing sites. Hence, we employed fuzzy similarity measure on the plain text of the website for the detection process.

F5 - Feature based on screenshot: This feature captures the screenshot of the visited website to generate a fingerprint representing the website. This feature counters the phishing sites which replaces the textual content with an image as shown in Figure 3.3.

Consider an example of phishing site targeting Paypal website and its HTML structure as shown in Listing 6.1. It is observed that the Paypal is misspelled as Paypa1 in the title and also attributes of meta data such as description, keywords are skipped to prevent presence of brandnames. Thus, these kind of websites bypass textual based anti-phishing techniques. Anchor links with # value redirecting to same page or not found pages are present in many phishing sites. Similarly images and scripts requesting URLs may point to internal or external pages. These abnormal behaviors are captured using the proposed features (F1-F5). From the sample code, it is observed that attribute

values of div tag contains "page logo main textinput actionsSpaced" etc. Similarly, attribute values of form tag contains "loginform" and body tag has "bview" as attribute value. The attribute values for other tags such as H1 and H2 are also extracted for each visited site and all these string of attribute values represent feature F1. The filename and pathname of resource URLs are also extracted from the given HTML. The pathnames are extracted from img, link and script tags i.e. "/css/style.css lib/signin.js img/css/fav.ico" etc to form feature F2. If there exists presence of non null anchor links then their paths are also added to the path name based features F2. The filenames from the above tags such as style.css, signin.js and fav.ico etc are considered to represent F3. The plaintext from the HTML i.e. "Log in to your PayPa1 account Log in Forgot your email or password?" etc is extracted using Jsoup library to generate feature F4.

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width" />
<title>Log in to your PayPa1 account</title>
<link rel="icon" href="img/css/fav.ico" />
<meta name="robots" content="noindex" />
<link href="/css/style.css" rel="stylesheet" type="text/css">
<script src="lib/signin.js"></script>
</head>
<body id="bview">
<div id="page">
<header>
<div class="logo"></div>
</header>
<div class="main">
<h1 class="headerText accessAid">Log in to continue</h1>
<form id="loginForm" method="post" action="checkpost.php">
<div class="textinput">
<input type="email" name="login_email" placeholder="Email">
</div>
<div class="textinput">
<input type="password" name="login_password" placeholder="
   Password">
</div>
<div class="actionsSpaced">
<button type="submit" name="BtnLogin" class="button">Log In</
   button>
</div>
<div class="forgotLink">
<a href="#" class="link">Forgot your email or password?</a>
</div></form>
<a href="#" class="btnSign">Sign Up</a>
</div></div>
<footer class="footer">
<ul><li><a href="#">Contact Us</a></li>
<li><a href="#">Privacy</a></li>
<li><a href="#">Legal</a></li>
<li><a href="#">Worldwide</a></li></ul>
</footer>
</body>
</html>
```

Listing 6.1: Source code of Paypal phishing site

We have extracted all the features and appended to a string without duplicates of feature values. This string is called as feature string which is given as input to hashing algorithms to obtain fingerprint or signature representing the website.

B. Blacklist database generation - Replica detection

In this component, DOM is extracted, parsed for the features F1-F4 and the screenshot of the website (F5) is captured using Selenium webdriver. Simhash technique shown in Algorithm 6.1 is applied on features F1-F3 representing the noisy part of the document to generate a unique signature. We term this signature as `simhash_noisy`. Similarly feature F4 representing plaintext is given as input to the same algorithm to generate the second fingerprint `simhash_pt`. Finally, Perceptual Hash (PHash) shown in Algorithm 6.2 is applied on the feature F5 representing screenshot of the website to generate the third fingerprint `phash_img`. All these extracted features undergo preprocessing prior to fingerprint generation. Preprocessing includes conversion of feature string to lowercase and removal of extra spaces. The ensemble of all the three hashes are used to identify the target domain of the phishing site. Note that, these hashing algorithms generate hash of similar documents that vary by less number of bits. Based on the empirical analysis, Simhash fingerprint of 64 bits and PHash of 256 bits are considered for the blacklist generation. Therefore, three unique signatures are generated for each of the phishing websites and they are stored in blacklist database. One of the advantages of fingerprint is that it provides a reduced dimensional representation of document. We have collected phishing sites at different time periods for the generation of blacklist database and for the replica detection. The dataset used for blacklist database generation was collected during April 2018 to October 2018 and the dataset for replica detection was collected during December 2018 to February 2019. The main intention to choose the dataset at different time slots is to illustrate that phishing sites get repeated with replica or manipulated content over time.

For the replica detection, the suspicious site undergoes feature extraction process to generate unique hashes with respect to Simhash and PHash. Once all the three hashes

Algorithm 6.1: Simhash algorithm to generate fingerprint of a website

Input : Feature string
Output: A fingerprint depicting the website

Tokenize the string into single words by using space delimiter
Initialize fingerprint size (64 bit)
Initialize an array $W[\ ]$ of zeros
For each token in the token-set, generate hash using hashing algorithm (MD5, SHA-1, SHA-256)
For each hash, for each bit j in the hash
if $j = 1$ **then**
    $W[j] = W[j] + 1$
else
    $W[j] = W[j] - 1$
end
For each bit k in the hash
if $W[k] > 0$ **then**
    $k = 1$
else
    $k = 0$
end
print output fingerprint in terms of bits

Algorithm 6.2: Generating Average Perceptual hash of an image

Input : Image I (MxN)
Output: Perceptual hash depicting the website

Resize the image I (MxN) to I (16x16)
Convert the image I to grayscale image
For each $pixel\_val$ in I:
sum =sum+ pixel_val
avg = sum/256
For each pixel_val in I:
if *pixel_val* $<$ *avg* **then**
    pixel_val = 0
else
    pixel_val = 1
end
print all bits of I together as final perceptual hash

of the given suspicious website are obtained, hamming distance (HD) is calculated for each of the three hashes with its corresponding blacklist of fingerprints. The hamming distance provides the similarity index between suspicious site and existing blacklisted sites. Lower the HD, greater the similarity between the webpages. We have conducted

experiments to calculate the HD threshold for comparing newly generated hashes with the existing blacklist. From the experimental analysis shown in Table 6.3, it is observed that HD=13 is used for Simhash comparison with respect to noisy part of the document. Similarly from Table 6.4, HD=6 is used as threshold for comparing plaintext-based Simhash. From the experimental results shown in Table 6.5, a threshold of HD=13 is used for PHash comparison. The detailed analysis of the threshold calculation with respect to all the three hashes is discussed in section 6.3, Experiment 1 (refer page no 163).

For each of the hashes generated for suspicious site, the status is calculated based on difference in bits between the suspicious site and blacklisted sites. If the difference in bits (HD) is below the calculated threshold, then the status of the given site is classified as phishing else it is marked as Non Blacklisted (NBL). As the existing blacklist contain target domain in addition to generated fingerprints, the target domain information is obtained for the confirmed phishing sites. Upon receiving the status for each of the hashes, the more frequent status (max_status) is calculated. The max_status contains either phish or NBL. Incase of max_status=NBL, the given suspicious site will undergo heuristic filtering else the target domain (t_domain) corresponding to the blacklisted hash is extracted.

6.2.2 Heuristic based filtering

This filtering module extracts features from URL and source code of the given webpage.

A. URL based features : In this section, we extract the lexical features to distinguish between legitimate and phishing URLs by inspecting the input URLs. These features are extracted either from full URL or hostname of the URL and they are listed in Table 6.1.

It is observed that the attackers sometimes use words such as login, signin, site etc. in phishing sites to confuse the users as these words are frequently found in login pages of the legitimate sites. Hence, we have considered a feature to check the URLs for such words. It is observed that many of the phishing sites have IP address instead of hostname and therefore we also check hostname for IP address in our proposed hand-

Table 6.1: Hand-crafted features

Sl. No.	Features	Values	# of features
1	Presence of PhishHinted words, HTTPS in base URL, IP address and Validate TLD	{0,1}	4
2	Presence of brandname in subdomain	{0,1}	1
3	Digits count in domain name, domain length, hostname length, average word length, longest word length, number of dots, number of hyphens, digits in hostname	Integer	8
5	Presence of PhishHinted words, brandnames in path of the URL	{0,1}	2
6	Digit count, slash count, longest word length, average word length, number of tokens in path of the URL	Integer	5
7	Length of base URL	Integer	1

crafted features. Attackers many a times include brandname of the target website as subdomain in the phishing sites and this will mislead the users to draw a conclusion that it is a legitimate site by observing the brandname. Hence, we also included the features such as presence of brandname in the subdomain, length of the URL, number of dots in the hostname in the URL-based features. There exists some words which are most common to the phishing URLs and less frequent in legitimate URLs. These words are termed as PhishHinted words. We adopted these words from our previous work (Rao et al. 2020) as features such as presence of PhishHinted words in hostname and path of the URL.

B. Source code based features : Since only structural properties of the URL are considered in URL based features, it may result in high false positive rate. Hence we have considered both URL and source code based features in our work. In this section, we extract features from the hyperlinks present in the source code of the website. The reason for choosing these features is that the attacker copies the content of target legitimate site into his designed phishing site resulting in hyperlinks redirected to target site. This results in more foreign links than local links. The local links are the hyperlinks which are pointing to local domain (domain of visited page) and foreign links are the hyperlinks pointing to foreign domain.

a) *(SF1,SF2,SF3) Presence of domain in key-terms*: This feature checks the presence of domain in key-terms. The brand-names of the websites usually appears in certain locations of webpage such as title, copyright, headers which renders the page searchable and recognizable. These locations of webpage are termed as key-terms. When the attacker copies the content of the target sites, he leaves the brand information intact by manipulating the anchor links with empty or null links (Patil and Patil 2018; Rao and Pais 2019a). This leads to mismatch between the domain and key-terms in phishing sites.

$$SF1 = \begin{cases} 0 & \text{if domain present in title} \\ 0.5 & \text{if title empty} \\ 1 & \text{otherwise} \end{cases} \tag{6.1}$$

$$SF2 = \begin{cases} 0 & \text{if domain present in copyright} \\ 0.5 & \text{if copyright empty} \\ 1 & \text{otherwise} \end{cases} \tag{6.2}$$

$$SF3 = \begin{cases} 0 & \text{if domain present in headers text} \\ 0.5 & \text{if headers text empty} \\ 1 & \text{otherwise} \end{cases} \tag{6.3}$$

b) *SF4-Presence of domain in least TF-IDF terms* : This feature checks the presence of domain in least TF-IDF terms. The Term Frequency - Inverse Document Frequency algorithm is used to extract the most prominent words related to the webpage. Term frequency defines the number of times a word appears in a webpage and number of webpages that contain the word is termed as inverse document frequency. The rationale behind this feature is that the frequency of brand name is very high in current webpage and is very less in other webpages of corpus.

Existing techniques (He et al. 2011; Ramesh et al. 2014; Xiang et al. 2011;

Zhang et al. 2007) used TF-IDF for identifying the brand names of the document from a corpus of documents. This consumes more time and space while calculating the TF and IDF for each document. Hence, we came up with modified TF-IDF by replacing all documents in a corpus by 6 to 7 documents. We considered each of brand locations such as plaintext, title, description, copyright, URL terms, Header text, maximum frequency domain (if not local domain) as individual document for identifying the brand names of the given document as shown in the Algorithm 6.3.

The brand name of a website is usually present in anyone or all of the following: 1. Title of the page, 2. Header texts with h1 and h2 tags, 3. Copyright section or 4. URL (because it makes the user familiar with brand and URL map). The reason for choosing maximum frequency domain is that some phishing attacks include anchor links pointing to target legitimate site; in that case we can get the target website brand name too.

In the existing works, top TF-IDF words are considered due to the fact that TF value of brand name in a website results in high count whereas IDF value of same brand term results in the low count (because of non relevance with other documents in the corpus). Combining both high TF value and low IDF values for the calculation of TF/IDF ratio results in high value. Hence terms with top TF-IDF values are more relevant to the specified document. But, in our case we considered brand locations of the given document as corpus which results in very high IDF value for the brand name. Note that, the brand name with high TF and IDF values result in low TF-IDF value. Hence, we considered the terms with low TF-IDF value as most relevant terms for the specified document.

$$SF4 = \begin{cases} 0 & \text{if domain present in} \\ & \text{least TF-IDF} \\ 1 & \text{otherwise} \end{cases} \tag{6.4}$$

c) *SF5-Common page detection ratio:* This feature calculates the ratio of number of hyperlinks redirected to a common page out of total number of hyperlinks in a web-page. This feature is explained in Chapter 3, Section 3.2.1 (*HF3*).

Algorithm 6.3: FindLeastTFIDFWords

Input : A HTML document *doc*, URL of website *url*
Output: List of 5 least TF-IDF words (w_i)

```
Initialize a String array brandLocations[ ]
host=getHost(url)
title=getTitle(doc)
plaintext=getPlaintext(doc)
desc=getDescription(doc)
cr=getCopyright(doc)
h1h2=getHeadersText(doc)
maxfreqdomain=getMaxFreqDomain(doc)
if maxfreqdomain==getDomain(url) then
    Add(host, title, plaintext, desc, cr,h1h2,maxfreqdomain) to brandLocations
     [ ]
else
    Add (host, title, plaintext, desc, cr,h1h2) to brandLocations [ ]
end
words=getLeastTFIDFwords(brandLocations [ ])
return words
```

d) *SF6-Zero links presence:* It checks the presence of anchorlinks in the body of HTML, if present, the attribute is set to 0 else it is set to 1. More details can be found in Chapter 3, Section 3.2.1 (*HF7*).

e) *SF7-Foreign Links ratio:* This feature calculates the ratio of foreign links to total number of anchor links from the source code of suspicious site. As mentioned earlier, the links which are redirected to non-local domains are termed as foreign links. Some of the phishing sites include the cloning of target legitimate site resulting in the traces of target legitimate site anchor links in designed phishing website. On the otherside, the legitimate sites might contain foreign links but the local anchor links ratio would be mostly higher than the foreign links.

$$SF7 = \begin{cases} \frac{al_f}{al_n} & \text{if } al_n > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

where al_f is the number of foreign anchor links and $al_n$is total number of anchor links in a website.

6.3 EXPERIMENTATION AND RESULTS

The BlackPhish takes URL as input and displays the status of the website as either legitimate or phishing site. The detection is done at two phases. If the phishing site is detected at first phase then BlackPhish also provides the target legitimate site. The phishing sites which go undetected in the first phase will undergo second level heuristic filtering.

A. Dataset

For the experimentation, we have collected dataset from two sources namely Phishtank and Google search results. Phishing sites are collected from Phishtank and for the legitimate sites, target legitimate brands are extracted from Phishtank and fed as search query to Google to return top 100 search results. These links in search results are preprocessed for the duplicates and *not found* pages for the generation of legitimate dataset. The source and denominations of both legitimate and phishing sets are given in Table 6.2.

Table 6.2: Dataset used for experimentation

Type	Source	Count	Link
Legitimate	Google Search results	5438	https://www.google.co.in/search?q=”+query+”&num=”+100
Phishing	PhishTank	4097	http://www.phishtank.com/developer

B. Tools used

We have implemented a Java program for the extraction of features from URL, source code and we used Javascript to develop a chrome extension for the detection of phishing sites. Jsoup library is used for parsing the HTML of the given URL. Finally, all the machine learning algorithms are implemented in python using Scikit[1] package.

C. Experimental evaluation

We have used traditional metrics such as TPR, TNR, FPR , FNR, Accuracy (Acc),

[1] http://scikit-learn.org

Precision (Pre), F1 score (F) and Matthews Correlation Coefficient (MCC) to evaluate the performance of our model.. We have conducted 5 experiments for assessing the blacklist filter and heuristic filter. Experiment 1 calculates the threshold with respect to `simhash_noisy`, `simhash_pt` and `phash_img` comparison in the blacklist filter. Experiment 2 deals with the effectiveness of blacklist filtering. Experiment 3 is further divided into 3 experiments to evaluate URL, source code based features and URL+source code based features respectively. These three experiments evaluate the heuristic filtering used in our model. The Performance of the BlackPhish is evaluated with two phase filtering processes in experiment 4. Finally, our work is compared with recent and popular phishing detection techniques in experiment 5.

Experiment 1: Threshold calculation for blacklist filtering:
In this section, we conducted experiments to calculate the thresholds for Simhash and PHash fingerprints which differentiate the legitimate sites from the existing blacklisted sites. Due to the uneven distribution of legitimate and phishing sites in our dataset, we have used MCC as a metric to decide the threshold with respect to each of the three hashes. Table 6.3 gives details regarding the threshold calculation for Simhash on noisy part of the document. It is observed that over decrease in similarity between the webpages, FPR and TPR are increasing. The rationale behind this experiment is to choose the threshold with less FPR and more TPR. From the experimental analysis, it is clear that the similarity <=80% resulted in highest MCC of 61.67% with 0.79% FPR and 53.60% TPR and hence we chose HD=13 bits as threshold for Simhash comparison.

Similarly, Table 6.4 provides threshold information with respect to Simhash on plaintext. Similarity <=90% resulted in 66.11% MCC which is better than the other values with respect to FPR (0.42%) and TPR (58.56%). Hence in our BlackPhish model, we have used threshold HD=6 bits with respect to simhash on plaintext.

The experimental results of PHash threshold calculation is shown in Table 6.5. At similarity <=95%, we obtained better MCC of 58.09% with 4.74% FPR and 56.94% TPR. Therefore, we have used the threshold HD=13 bits for comparing new PHash with the existing blacklist of PHashes. Based on these thresholds, we get the target brand with respect to each of the three hashes when a given URL is detected as phishing site

Table 6.3: Calculation of Threshold for simhash_noisy generation

Similarity(%)	HD (bits)	FPR	TPR	Acc	Pre	F-Score	MCC
=100	=0	0.13	13.06	62.57	98.71	23.07	27.64
<=95	<=3	0.13	18.06	64.72	99.06	30.55	33.04
<=90	<=6	0.15	25.53	67.92	99.24	40.61	40.07
<=85	<=10	0.24	42.35	75.09	99.26	59.37	53.87
<=80	<=13	0.79	53.60	79.61	98.08	69.32	61.67
<=75	<=16	10.92	67.12	79.64	82.24	73.91	58.30
<=70	<=19	78.56	95.51	53.27	47.81	63.72	24.07

Table 6.4: Calculation of Threshold for simhash_pt blacklist generation

Similarity(%)	HD(bits)	FPR	TPR	Acc	Pre	F-Score	MCC
=100	=0	0.09	42.47	75.23	99.71	59.57	54.25
<=95	<=3	0.22	50.67	78.68	99.43	67.13	60.39
<=90	<=6	0.42	58.56	81.95	99.05	63.60	66.11
<=85	<=10	4.95	65.02	82.15	90.83	75.79	64.44
<=80	<=13	24.29	72.35	74.26	69.17	70.72	47.82
<=75	<=16	60.68	87.11	59.85	51.96	65.09	29.15

Table 6.5: Calculation of Threshold for phash_img blacklist generation

Similarity(%)	HD (bits)	FPR	TPR	Acc	Pre	F-Score	MCC
=100	=0	0.2	29	69.38	99.08	44.86	42.99
<=95	<=13	4.74	56.94	78.79	90.04	69.77	58.09
<=90	<=26	15.17	71.44	79.08	78.01	74.58	57.02
<=85	<=38	30.38	80.45	74.27	66.61	72.88	49.61
<=80	<=51	54.47	89.70	64.51	55.37	68.47	37.92
<=75	<=65	81.98	97.56	52.20	47.27	63.69	24.34

by the blacklist filter. The undetected sites by the blacklist filter will undergo next level of filtering i.e, heuristic filtering.

Experiment 2: Effectiveness of Blacklist filtering

Based on the thresholds obtained in the previous experiment, we evaluated the effectiveness of blacklist filtering in detecting the phishing sites. In this experiment, we

Table 6.6: Effectiveness of Blacklist filtering

Information	Value
# of legitimate sites	5438
# of phishing sites	4097
# of legitimate sites misclassified by blacklist filter	4
# of phishing sites detected by blacklist filter	2277
# of phishing sites undergoing heuristic filtering	1820

have considered an ensemble of `simhash_noisy`, `simhash_pt` and `phash_img` in blacklist filtering. If at least any two hashing techniques classify the given site as "Phish" then we mark the given website as phishing site. The blacklist filter also provides the target brand information when a phishing site is detected. Surprisingly, the results in Table 6.6 demonstrated only 4 false positives out of 5438 legitimate sites which shows the effectiveness of the features used in generating blacklist fingerprints. It is also observed that, 2277 out of 4097 phishing sites were detected by the blacklist filtering and thus it implies, only 1820 out of 4097 phishing sites undergo heuristic filtering. Thus, the effectiveness of both blacklist and heuristic filter together as a model is discussed in experiment 4.

Experiment 3: Evaluation of heuristic filtering

In this section, we conducted three experiments to evaluate the features used in heuristic filtering. The first experiment deals with evaluation of URL based features, second experiment evaluates the source code based features and the third experiment deals with evaluation of both URL and source code based features. We have applied XGBoost, Extra-Tree, RF classifiers individually and also considered an ensemble of the three classifiers to evaluate the feature sets. We have applied 10 fold cross-validation and considered 100 estimators on our dataset during the classification process.

3.a Evaluation of BlackPhish with URL based features : From the experimental results shown in Table 6.7, it is observed that Random Forest outperformed other individual classifiers with an accuracy of 93.36%. Though RF performed better than other individual classifiers, it is demonstrated that the ensemble has achieved an accuracy of

93.39% with MCC of 86.52%. This shows that the application could detect an ample of phishing sites without even visiting the URL which demonstrates the importance of proposed URL based features.

Table 6.7: Evaluation of URL features

Metrics	XGBoost	RF	Extra-Tree	Ensemble
TPR	87.53	92.19	91.80	92.38
FPR	7.10	5.76	5.96	5.85
Accuracy	90.59	93.36	93.08	93.39
Precision	90.28	92.35	92.07	92.25
F measure	88.88	92.27	91.93	92.32
MCC	80.76	86.45	85.87	86.52

3.b Evaluation of BlackPhish with Source code based features : From the experimental analysis, we observed that the source code based features have more information of abnormal behavior in phishing sites over the URL based features. The experimental results with source code features are given in Table 6.8. Out of all the three classifiers, RF achieved a significant accuracy of 97.55%. From the results, it is also observed that the ensemble model has achieved an accuracy of 97.61% with TPR of 98.05% and FPR of 2.72% which shows the richness of proposed source code based features. Note that the ensemble of classifiers outperformed individual classifiers with respect to MCC (95.14%).

3.c Evaluation of BlackPhish with both URL and Source code based features: In this section, we integrated both the URL and Source code based features to check

Table 6.8: Evaluation of Source code-based features

Metrics	XGBoost	RF	Extra-Tree	Ensemble
TPR	96.92	98.10	97.34	98.05
FPR	3.79	2.87	2.76	2.72
Accuracy	96.52	97.55	97.28	97.61
Precision	95.07	96.26	96.38	96.45
F measure	95.99	97.17	96.85	97.24
MCC	92.93	95.02	94.47	95.14

Table 6.9: Evaluation of URL and Source code-based features

Metrics	XGBoost	RF	Extra-Tree	Ensemble
TPR	97.34	98.39	97.49	98.29
FPR	2.39	1.97	1.56	1.77
Accuracy	97.49	98.19	98.03	**98.26**
Precision	96.84	97.41	97.92	97.67
F measure	97.09	97.90	97.70	97.98
MCC	94.89	96.31	95.98	**96.45**

Table 6.10: BlackPhish with heuristic and blacklist filtering

Filter	**L**	**P**	**TP**	**TN**	**FP**	**FN**	**Acc**	**MCC**
BL filter	5438	4097	2277	5434	4	1820	80.87	64.41
Heuristic filters	5434	1820	1759	5377	57	61	98.37	95.67
BL + Heuristic	5438	4097	1759 + 2277=4036	5377	57+4=61	61	**98.72**	**97.39**

the complement behavior in detecting the phishing sites. From the experimental results shown in Table 6.9, it is observed that our model attained an accuracy of 98.26% with TPR of 98.29% and FPR of 1.77%. This shows that Source code based features complemented the application in achieving greater MCC (96.45%) than the other classifiers.

Experiment 4: Evaluation of BlackPhish with inclusion and exclusion of blacklist filtering

In this experiment, both blacklist and heuristic filters are evaluated together as a single model used in the BlackPhish. From the experimental results shown in Table 6.10, the blacklist filter is able to detect 2277 phishing sites. The remaining 1820 undetected phishing sites and 5434 legitimate sites undergo heuristic filtering. From the experimental analysis, the heuristic filter misclassified 57 out of 5434 legitimate sites and 61 out 1820 phishing sites. Finally both blacklist and heuristic filters together as an entity correctly classified 4036 out of 4097 phishing sites and 5377 out of 5438 legitimate sites and achieved an accuracy of 98.72% with MCC of 97.39%.

As discussed in the experiment 3, the ensemble model with only heuristic filter (URL+source code) achieved an accuracy of 98.26% with 96.45% MCC. This clearly shows that the addition of blacklist filter prior to the heuristic filter contributed in improving the performance of the BlackPhish i.e. 98.72% Acc and 97.39% MCC. Note that, by increasing the size of the blacklisted fingerprints, the performance of our model can be further enhanced.

Experiment 5: Comparison with Existing work

In this section, we compare our work with existing works using URL and source code based features for the detection of phishing sites. In this chapter, we have implemented Shirazi et al. (2018) work and Su et al. (2013) work with our dataset to evaluate the performance of the BlackPhish model. The main intention to choose these techniques is due to the similarity in features extracted from both URL and Source code. For simplicity, we follow the convention as W1 for Shirazi et al. work, W3 for Su et al., W2 for our work with URL and source code features and W2' for our work with URL features. Since W1, W2 deals with URL and source code based features, we compared W1 with W2. Similarly, W3 is compared with W2' as both W3 and W2' extract features from the URL. The comparison results of our work with existing works are shown in Table 6.11. From the results, it is observed that, W2 outperformed W1 with a significant accuracy of 98.26%, MCC of 96.45%. This shows the importance of heuristic features used in our work. Also note that, our work W2' achieved an additional accuracy of 12.78% more than the existing work W3 with respect to URL features.

Table 6.11: Comparison of our work with existing works

Metrics (%)	**W1**	**W2**	**W3**	**W2'**
TPR	93.04	98.29	75.01	92.38
FPR	4.10	1.77	15.17	5.85
Accuracy	94.67	98.26	80.61	93.39
Precision	94.47	97.67	78.84	92.25
F-measure	93.75	97.98	76.87	92.32
MCC	89.12	96.45	60.25	86.52

6.4 DISCUSSION

The presented model is constructed based on the two components namely blacklist and heuristic filtering modules. In this section, we discuss about the effectiveness of Black-Phish with respect to each of the above mentioned components and implementation details of our work as a chrome extension.

6.4.1 BlackPhish: Chrome Extension

The main goal of our work is to design a mechanism to enable real time protection against phishing sites at the client side. Hence, we implemented our model as a chrome extension which gives the legitimacy status of the website on click of the extension. At the server side, source code and screenshot are extracted for the generation of inputs to filtering modules. The REST API service is used by the model for transferring the URL to the remote server where the dual filtering takes place for the detection of phishing sites. The REST service is implemented with Spring framework and hosted on Intel Xeon 16 core Ubuntu server with 2.67GHz processor and 16GB RAM. We used GET method for transferring the URL to the BlackPhish application.

Once the URL is obtained at the server side, firstly an enhanced blacklist is checked for a match. If a match is found, then the server sends the status of the URL as phishing. If no match is found, the URL undergoes second level of filtering by extracting URL and source code-based features. These features are fed to the pre-loaded trained ensemble model to predict the legitimacy and finally the status of the URL is sent back to the extension.

Since the goal of our work is to ensure protection from phishing sites in real time, we performed analysis on time taken for the detection process. We observed that the BlackPhish took an average time of 2.43 sec with respect to blacklist filtering and 2.61 sec with respect to both blacklist and heuristic filtering. However, this time is dependent on various factors such as speed of the Internet, website hosted on server, and speed of the system. In our case, we used 100mbps bandwidth and as mentioned earlier system with Intel Xeon 16 core Ubuntu server with 2.67GHz processor and 16GB RAM. The output of the BlackPhish chrome extension is shown as a popup window containing the status of the website as shown in Figure 6.4. In image a) of Figure 6.4 BlackPhish detects the phishing site targeting Alibaba website with a response time of 2.27 sec. As the site is detected at blacklist filter, the target information of the phishing site is also revealed in the extension. The image b) and c) of Figure 6.4 shows the detection of phishing and legitimate sites by heuristic filter with a response time of 2.62 sec and 2.57 sec respectively.

6.4.2 Effectiveness of BlackPhish

In this section, we discuss the effectiveness of BlackPhish in detecting phishing sites in desktop computers. The traditional blacklist used in various existing techniques fail to detect phishing sites with small change in the URL. To address this problem, we have proposed an enhanced blacklist as a first level filter in BlackPhish so that the attackers have to put additional effort in order to bypass it. The phishing sites which go undetected by enhanced blacklist undergo second level of filtering i.e, heuristic-based filter. In this level of filtering, both URL and source code features are extracted and these features are fed to the ML classifiers i.e, RF, XGBoost and ExtraTree for the prediction. Use of two-level filtering resulted in less probability of phishing sites bypassing the technique. We have also considered the ensemble of three classifiers to increase the detection rate. From the experimental results, we observed that the ensemble of classifiers achieved an accuracy of 98.63% and MCC of 97.20% which clearly shows that the ensemble performed better than the individual classifiers in detecting phishing sites.

We also conducted an experiment to determine the effectiveness of blacklist filter in our presented model. It is observed that the blacklist filter alone is able to detect 55.58% (2277/4097) of phishing sites and the remaining phishing, legitimate sites are sent to heuristic filter. Note that, inclusion of blacklist with heuristic filter resulted in 98.72% accuracy, 97.39% MCC which is better than the accuracy (98.26%) and MCC(96.45%) corresponding to exclusion of blacklist filter. This clearly shows the contribution of blacklist filter in our BlackPhish model to detect the phishing sites. Note that, as the size of the blacklist increases there exists more chance of further enhancement in the performance of the model.

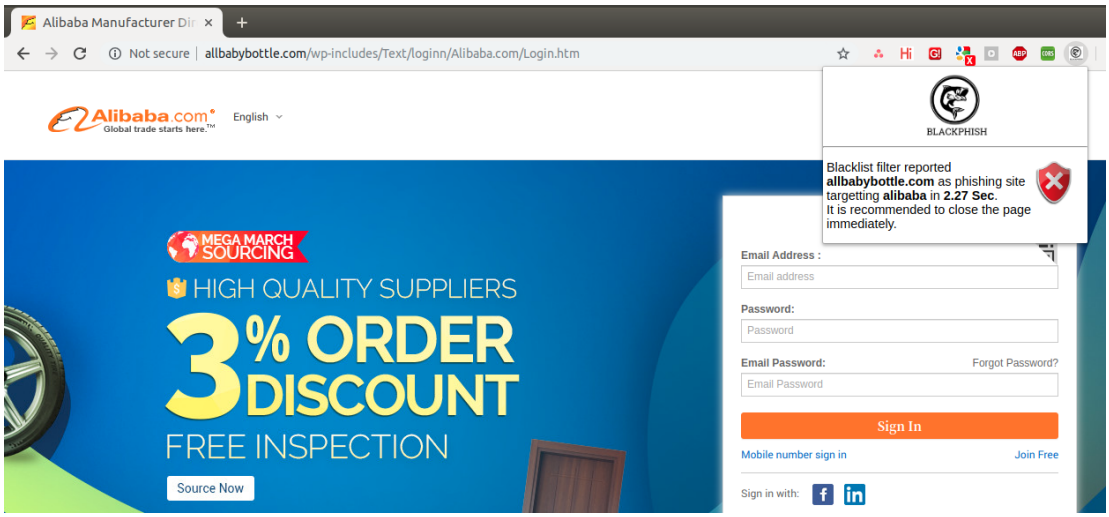
6.5 SUMMARY

In this chapter, we presented a comprehensive approach where same URL undergoes two level filtering to identify its legitimacy. The enhanced blacklist is used as first level filter to identify the phishing sites based on blacklist of fingerprints generated using Simhash of plaintext, noisy data and PHash of screenshot with respect to phishing sites. From the experimental analysis, it is observed that 55.58% of phishing sites

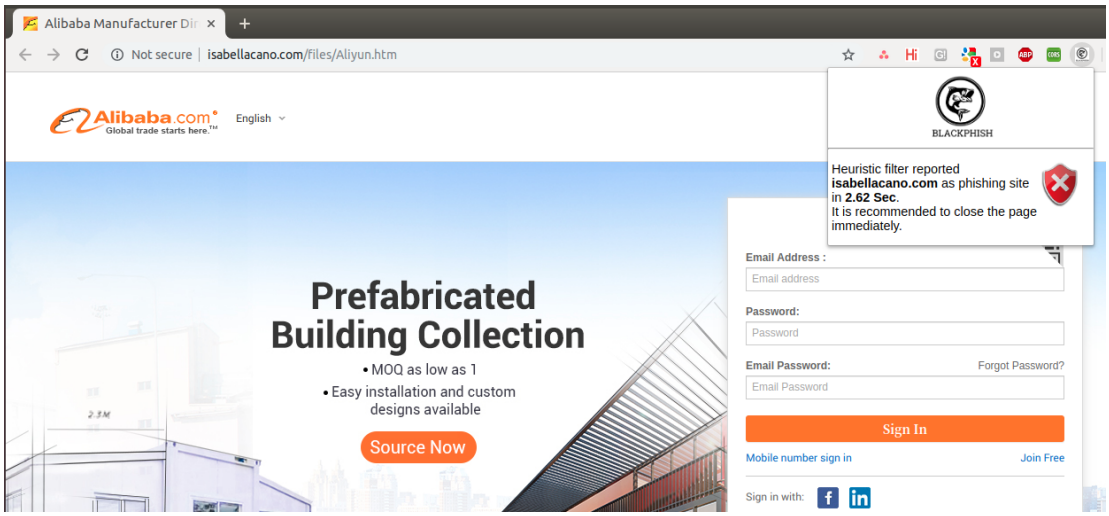
are replicated or manipulated from the existing blacklisted sites. Unlike traditional blacklist, the blacklist filter generates a set of three fingerprints using noisy part of HTML, plaintext and screenshot of the website. Ensemble of these 3 fingerprints is used to predict the target domain of the phishing site detected by the blacklist filter.

The sites which go undetected using blacklist filter undergoes heuristic filtering. This second-level filtering includes extraction of URL and source code based features which are fed to trained ensemble model. The ensemble model is designed by combining RF, XGBoost and Extra tree classifiers to detect the legitimacy of the given website. From the experimental results, the ensemble model with two level-filtering outperformed individual classifiers with an accuracy of 98.71% and MCC of 97.37%. Also, it is observed that the contribution of source code based features is higher than the URL based features but the addition of URL features to the source code based features complimented the detection of phishing sites with a significant accuracy.

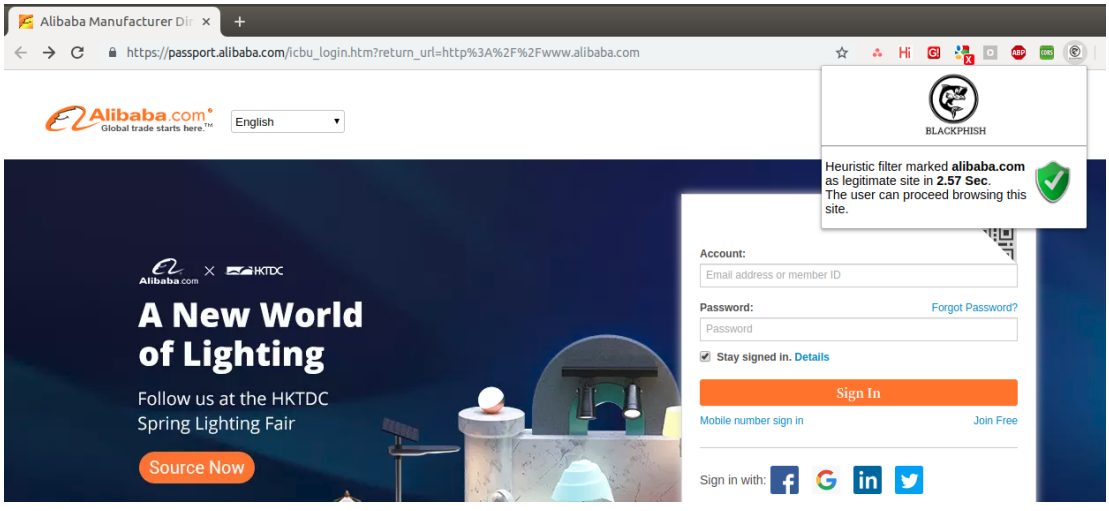
We compared our work with the existing URL and source code based anti-phishing techniques to assess the performance of our ensemble model. From the results, it is observed that our model performed better than the existing technique with a significant difference in accuracy and MCC. We have tested our model by deploying it as chrome extension locally.



(a) BlackPhish with Blacklist Filtering



(b) BlackPhish with Heuristic Filtering



(c) BlackPhish with Legitimate

Figure 6.4: UI of BlackPhish

CHAPTER 7

CONCLUSIONS AND FUTURE SCOPE

Phishing is an art of manipulating online users into performing actions or divulging sensitive information. Phishing attacks are on the rise. Lack of awareness is one of the major reason for the rise of phishing attacks. It is impossible to defend against every attack. User education is the strongest defense and at the same time it is the weakest link to counter phishing attacks. It has been a continual threat from 20 years after its appearance. It has become one of the most lucrative cybercrime accounting to billions of dollars in loss. This thesis does not address all the research challenges in phishing detection to provide a single silver-bullet solution to mitigate all the phishing attacks. However, it provides some relevant contributions to fight against phishing sites.

Firstly, in chapter 3, we presented a machine learning framework which includes rich and comprehensive features for the detection of phishing sites. We also used Random Forest for the classification of phishing sites. The variation of RF i.e. oblique Random Forest outperformed the existing baseline models Cantina and Cantina+ with an accuracy of 99.55%. The third party services (WHOIS, page ranking and search results) in the model played a significant role in the detection process but may fail when PSHCS are encountered. To counter these kinds of phishing sites, two novel techniques are presented in chapter 4.

In chapter 4, we presented two content-based techniques which detect phishing sites hosted on compromised servers with and without third party services. The similarity score between the visited page and home page is used as a parameter to detect the

legitimacy of the suspicious site.

1. Firstly, the third-party independent technique (Curb-Phish) is presented with the features extracted from URL and source code of the suspicious site. These features were fed to Twin SVM classifier resulting in accuracy of 98.4% with TPR of 98.72% and TNR of 98.08%.

2. We further presented an improved search engine based technique (Jail-Phish) to detect the PSHCS sites with dynamic search query. This technique is able to achieve an accuracy of 98.61% with TPR of 97.77% and TNR of 99.36%.

Since these techniques consider source code based features, it makes mandatory to visit the URL which might lead to accidental download of malware. Hence, we present two URL based techniques in chapter 5 which detects phishing URLs without even visiting the URLs.

In chapter 5, we presented two URL-based methods for the detection of phishing sites by inspecting the URLs using deep learning and machine learning algorithms.

1. Firstly, a web application named CatchPhish is presented which includes hybrid set of features containing both manually crafted URL features and TF-IDF features for training the model. RF is used as classifier for the model and could achieve an accuracy of 94.26%, TPR of 93.31% and TNR of 96.65%.

2. Later, we presented a multi-model ensemble based technique for the phishing detection in mobile devices. The multi-model ensemble of LSTM-SVM is used for training the model and could achieve an accuracy of 97.30%, TPR of 97.31% and TNR of 97.28%. The presented application is named as PhishDump which is lightweight and is able to detect the phishing sites with a low response time of 2.26 sec.

Since these techniques extract features from the URL and not by source code, they detect the phishing sites with low response time and free from drive-by-downloads. Hence, these techniques can be used as first level filtering of phishing sites. Note that,

all the presented techniques detect phishing sites with a significant accuracy but they lack the information of legitimate site which is targeted by the attacker. Hence, we presented a new technique in chapter 6 which detects the phishing site along with its target legitimate domain using visual similarity approach.

In chapter 6, finally we presented a comprehensive approach where same URL undergoes two level filtering to identify its legitimacy. The enhanced blacklist is used as first level filter to identify the phishing sites based on blacklist of fingerprints generated using Simhash of plaintext, noisy data and PHash of screenshot with respect to phishing sites. If the URL is detected by the blacklist filter, a warning is alerted to the user along with target legitimate domain. Otherwise, heuristic filtering is applied for the classification of non-blacklisted phishing sites.

All the works in this thesis are either deployed as web application, Chrome extensions or android application, they can be used by the online users in the real time phishing detection. The machine learning framework with comprehensive feature set resulted in significant performance and hence can be used by the industry to generate a system that is fast, highly reliable and adaptive to new attacks. Note that, human verified blacklist used by the industry do not generalize well with future unseen cases and moreover they are slow to respond to new phishing sites. The Jail-Phish and Curb-Phish detects phishing sites hosted on compromised domain. The Jail-phish detects newly registered legitimate or unpopular sites that are not covered by the existing search engine based techniques. Tech giants (e.g. Facebook, Google, Microsoft etc), banks, credit card companies, online payment service providers (e.g. PayPal), and even social networking sites can use these techniques in identifying the phishing sites on compromised servers. Techniques CatchPhish and Phishdump can be used by both online users and researchers as first level filtering of phishing site. Moreover, Catchphish can take more than one URL for the detection and Phishdump enables the online users to detect the phishing sites in their mobile devices. Finally, enhanced blacklist mechanism (Blackphish) can be used by the researchers and industry for the detection of near duplicate phishing sites. The target companies can maintain blacklist of phishing sites targeting them and can detect zero day phishing sites that are near duplicate to existing blacklists.

Future Scope

Although we evaluated a set of our presented techniques in detecting the phishing sites in this thesis, our techniques can be further improved in handling different kinds of phishing sites.

- In the presented schemes, the tuning parameters for the machine learning algorithms are calculated using grid search mechanism but the use of genetic algorithms for tuning the classifiers can be explored in the future scope of the thesis. We also would like to use clustering algorithms coupled with supervised algorithms to further improve the accuracy of the models.

- There exist many source-code based features in the existing techniques but lacks the features that detects javascript events on links or input fields. There exist very limited work for the detection of phishing sites hosted on compromised servers (PSHCS) hence exploring and identifying efficient mechanism for the PSHCS is a promising future research direction to look into.

- Feature selection techniques are used to select relevant features and discard redundant and irrelevant features. Like ensemble learning, which is used for improving the performance of the model, we would like to use feature selection ensemble to select the optimal features for feature reduction and better performance.

- The number of techniques to counter website phishing in mobile devices is very less compared to desktop devices. This is due to the fact that the detection needs more resources to classify the phishing pages. Hence, we would like to work on identifying lightweight content based features in detecting the phishing websites in mobile devices.

- In email phishing, the user is either redirected to website phishing or installation of malware. Moreover, from the Avanan research [1] it is claimed that more than half of the phishing emails (51%) contain links to malware. Hence, we would

[1] https://www.avanan.com/how-email-became-the-weakest-link

like to focus on detection of email phishing such that it prevents the user from clicking the malicious link in the email.

In conclusion, this dissertation presents rich feature set machine learning framework for the phishing detection. Further, we presented two novel techniques to detect PSHCS which is more efficient than third-party based solutions. We also presented two lightweight URL based solutions to be used as first level filtering of phishing URLs. Finally, we presented a lightweight visual similarity approach with fingerprints of blacklisted phishing sites which also alerts the user with targeted legitimate domain.

BIBLIOGRAPHY

Abdelhamid, N., Ayesh, A. and Thabtah, F. (2014). "Phishing detection based associative classification data mining." *Expert Systems with Applications*, 41(13), 5948 – 5959.

Afroz, S. and Greenstadt, R. (2011). "Phishzoo: Detecting phishing websites by looking at them." In *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*, IEEE, 368–375.

Agarwal, B., Ramampiaro, H., Langseth, H. and Ruocco, M. (2018). "A deep network model for paraphrase detection in short text messages." *Information Processing & Management*, 54(6), 922–937.

Aggarwal, A., Rajadesingan, A. and Kumaraguru, P. (2012). "Phishari: automatic real-time phishing detection on twitter." In *eCrime Researchers Summit (eCrime), 2012*, IEEE, 1–12.

Akinyelu, A. A. and Adewumi, A. O. (2014). "Classification of phishing email using random forest machine learning technique." *Journal of Applied Mathematics*, 2014.

Aleroud, A. and Zhou, L. (2017). "Phishing environments, techniques, and countermeasures: A survey." *Computers & Security*, 68, 160–196.

Almomani, A., Wan, T.-C., Altaher, A., Manasrah, A., ALmomani, E., Anbar, M., ALomari, E. and Ramadass, S. (2012). "Evolving fuzzy neural network for phishing emails detection." *Journal of Computer Science*, 8(7), 1099.

Amrutkar, C., Kim, Y. S. and Traynor, P. (2017). "Detecting mobile malicious webpages in real time." *IEEE Transactions on Mobile Computing*, (8), 2184–2197.

APWG (2014). "Global phishing reports second half 2014." `http://docs.apwg.org/reports/APWG_Global_Phishing_Report_2H_2014.pdf`). Accessed: 2016-06-01.

APWG (2016a). "Phishing attack trends reports, first quarter 2016." `http://docs.apwg.org/reports/apwg_trends_report_q1_2016.pdf`a). Accessed: 2016-06-01.

APWG (2016b). "Phishing attack trends reports, fourth quarter 2016." `http://docs.apwg.org/reports/apwg_trends_report_q4_2016.pdf`b). Accessed: 2017-03-03.

APWG (2017). "Phishing attack trends reports, first half 2017." `http://docs.apwg.org/reports/apwg_trends_report_h1_2017.pdf`). Accessed: 2018-01-01.

Ardi, C. and Heidemann, J. (2016). "Auntietuna: Personalized content-based phishing detection." In *NDSS Usable Security Workshop (USEC)*.

Bahnsen, A. C., Bohorquez, E. C., Villegas, S., Vargas, J. and González, F. A. (2017). "Classifying phishing urls using recurrent neural networks." In *Electronic Crime Research (eCrime), 2017 APWG Symposium on*, IEEE, 1–8.

Basnet, R. B., Sung, A. H. and Liu, Q. (2011). "Rule-based phishing attack detection." In *International Conference on Security and Management (SAM 2011), Las Vegas, NV*.

Bauer, E. and Kohavi, R. (1999). "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants." *Machine learning*, 36(1), 105–139.

Bottazzi, G., Casalicchio, E., Cingolani, D., Marturana, F. and Piu, M. (2015). "Mp-shield: a framework for phishing detection in mobile devices." In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, IEEE, 1977–1983.

Britt, J., Wardman, B., Sprague, A. and Warner, G. (2012). "Clustering potential phishing websites using deepmd5.." In *LEET*.

Burges, C. J. (1998). "A tutorial on support vector machines for pattern recognition." *Data mining and knowledge discovery*, 2(2), 121–167.

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*, ACM, 197–206.

Cao, Y., Han, W. and Le, Y. (2008). "Anti-phishing based on automated individual white-list." In *Proceedings of the 4th ACM workshop on Digital identity management*, ACM, 51–60.

Chang, E. H., Chiew, K. L., Tiong, W. K. et al. (2013). "Phishing detection via identification of website identity." In *IT Convergence and Security (ICITCS), 2013 International Conference on*, IEEE, 1–4.

Chen, D. and Mak, B. K.-W. (2015). "Multitask learning of deep neural networks for low-resource speech recognition." *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(7), 1172–1183.

Chen, W., Zhang, W. and Su, Y. (2018). "Phishing detection research based on lstm recurrent neural network." In *International Conference of Pioneering Computer Scientists, Engineers and Educators*, Springer, 638–645.

Chiew, K. L., Chang, E. H., Tiong, W. K. et al. (2015). "Utilisation of website logo for phishing detection." *Computers & Security*, 54, 16–26.

Chiew, K. L., Choo, J. S.-F., Sze, S. N. and Yong, K. S. (2018). "Leverage website favicon to detect phishing websites." *Security and Communication Networks*, 2018.

Chiew, K. L., Tan, C. L., Wong, K., Yong, K. S. and Tiong, W. K. (2019). "A new hybrid ensemble feature selection framework for machine learning-based phishing detection system." *Information Sciences*, 484, 153 – 166.

Choi, H., Zhu, B. B. and Lee, H. (2011). "Detecting malicious web links and identifying their attack types.." *WebApps*, 11, 11–11.

Chorghe, S. P. and Shekokar, N. (2016). "A solution to detect phishing in android devices." In *International Conference on Information Systems Security*, Springer, 461–470.

Chou, N., Ledesma, R., Teraguchi, Y., Mitchell, J. C. et al. (2004). "Client-side defense against web-based identity theft.." In *NDSS*.

Chu, W., Zhu, B. B., Xue, F., Guan, X. and Cai, Z. (2013). "Protect sensitive sites from phishing attacks using features extractable from inaccessible phishing urls." In *Communications (ICC), 2013 IEEE International Conference on*, IEEE, 1990–1994.

Cook, D. L., Gurbani, V. K. and Daniluk, M. (2008). "Phishwish: a stateless phishing filter using minimal rules." In *International Conference on Financial Cryptography and Data Security*, Springer, 182–186.

Cortes, C. and Vapnik, V. (1995). "Support-vector networks." *Machine learning*, 20(3), 273–297.

Dewan, P. and Kumaraguru, P. (2015). "Towards automatic real time identification of malicious posts on facebook." In *Privacy, Security and Trust (PST), 2015 13th Annual Conference on*, IEEE, 85–92.

Dhamija, R., Tygar, J. D. and Hearst, M. (2006). "Why phishing works." In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, ACM, 581–590.

Dietterich, T. G. (2000). "Ensemble methods in machine learning." In *International workshop on multiple classifier systems*, Springer, 1–15.

Dong, X., Clark, J. A. and Jacob, J. L. (2010). "Defending the weakest link: phishing websites detection by analysing user behaviours." *Telecommunication Systems*, 45(2-3), 215–226.

Drew, J. and Moore, T. (2014). "Automatic identification of replicated criminal websites using combined clustering." In *Security and Privacy Workshops (SPW), 2014 IEEE*, IEEE, 116–123.

Dunlop, M., Groat, S. and Shelly, D. (2010). "Goldphish: Using images for content-based phishing analysis." In *Internet Monitoring and Protection (ICIMP), 2010 Fifth International Conference on*, IEEE, 123–128.

Fernández-Delgado, M., Cernadas, E., Barro, S. and Amorim, D. (2014). "Do we need hundreds of classifiers to solve real world classification problems." *J. Mach. Learn. Res*, 15(1), 3133–3181.

Fette, I., Sadeh, N. and Tomasic, A. (2007). "Learning to detect phishing emails." In *Proceedings of the 16th international conference on World Wide Web*, ACM, 649–656.

Fung, G. M. and Mangasarian, O. L. (2005). "Multicategory proximal support vector machine classifiers." *Machine learning*, 59(1-2), 77–97.

Garera, S., Provos, N., Chew, M. and Rubin, A. D. (2007). "A framework for detection and measurement of phishing attacks." In *Proceedings of the 2007 ACM workshop on Recurring malcode*, ACM, 1–8.

Gastellier-Prevost, S., Granadillo, G. G. and Laurent, M. (2011). "Decisive heuristics to differentiate legitimate from phishing sites." In *Network and Information Systems Security (SAR-SSI), 2011 Conference on*, IEEE, 1–9.

Gómez-Ríos, A., Tabik, S., Luengo, J., Shihavuddin, A., Krawczyk, B. and Herrera, F. (2019). "Towards highly accurate coral texture images classification using deep convolutional neural networks and data augmentation." *Expert Systems with Applications*, 118, 315–328.

Gowtham, R. and Krishnamurthi, I. (2014). "A comprehensive and efficacious architecture for detecting phishing webpages." *Computers & Security*, 40, 23–37.

Han, W., Wang, Y., Cao, Y., Zhou, J. and Wang, L. (2007). "Anti-phishing by smart mobile device." In *Network and Parallel Computing Workshops, 2007. NPC Workshops. IFIP International Conference on*, IEEE, 295–302.

Hara, M., Yamada, A. and Miyake, Y. (2009). "Visual similarity-based phishing detection without victim site information." In *Computational Intelligence in Cyber Security, 2009. CICS'09. IEEE Symposium on*, IEEE, 30–36.

He, M., Horng, S.-J., Fan, P., Khan, M. K., Run, R.-S., Lai, J.-L., Chen, R.-J. and Sutanto, A. (2011). "An efficient phishing webpage detector." *Expert Systems with Applications*, 38(10), 12018–12027.

Ho, T. K. (1998). "The random subspace method for constructing decision forests." *IEEE transactions on pattern analysis and machine intelligence*, 20(8), 832–844.

Hochreiter, S. and Schmidhuber, J. (1997). "Long short-term memory." *Neural computation*, 9(8), 1735–1780.

Hong, J. (2012). "The state of phishing attacks." *Communications of the ACM*, 55(1), 74–81.

Hou, J. and Yang, Q. (2012). "Defense against mobile phishing attack." *Computer Security*.

Huang, H., Qian, L. and Wang, Y. (2012). "A svm-based technique to detect phishing urls." *Information Technology Journal*, 11(7), 921–925.

Huh, J. H. and Kim, H. (2011). "Phishing detection with popular search engines: Simple and effective." In *International Symposium on Foundations and Practice of Security*, Springer, 194–207.

Jagatic, T. N., Johnson, N. A., Jakobsson, M. and Menczer, F. (2007). "Social phishing." *Communications of the ACM*, 50(10), 94–100.

Jain, A. K. and Gupta, B. B. (2017). "Phishing detection: analysis of visual similarity based approaches." *Security and Communication Networks*, 2017.

Jain, A. K. and Gupta, B. B. (2018). "Two-level authentication approach to protect from phishing attacks in real time." *Journal of Ambient Intelligence and Humanized Computing*, 9(6), 1783–1796.

Jayadeva, Khemchandani, R. and Chandra, S. (2007). "Twin support vector machines for pattern classification." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5), 905–910.

Jayadeva, Khemchandani, R. and Chandra, S. (2017). *Twin Support Vector Machines*, Springer.

Jiang, M., Liang, Y., Feng, X., Fan, X., Pei, Z., Xue, Y. and Guan, R. (2018). "Text classification based on deep belief network and softmax regression." *Neural Computing and Applications*, 29(1), 61–70.

Joshi, Y., Saklikar, S., Das, D. and Saha, S. (2008). "Phishguard: A browser plug-in for protection from phishing." In *Internet Multimedia Services Architecture and Applications, 2008. IMSAA 2008. 2nd International Conference on*, IEEE, 1–6.

KasperskyLab (2014). "Kaspersky lab:spam and phishing trends and statistics report q1 2014." `https://usa.kaspersky.com/internet-security-center/threats/spam-statistics-report-q1-2014`). Accessed: 2015-07-15.

Khonji, M., Iraqi, Y. and Jones, A. (2013). "Phishing detection: a literature survey." *IEEE Communications Surveys & Tutorials*, 15(4), 2091–2121.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. and Dyer, C. (2016). "Neural architectures for named entity recognition." *arXiv preprint arXiv:1603.01360*.

Le, H., Pham, Q., Sahoo, D. and Hoi, S. C. (2018). "Urlnet: Learning a url representation with deep learning for malicious url detection." *arXiv preprint arXiv:1802.03162*.

Li, Y., Yang, Z., Chen, X., Yuan, H. and Liu, W. (2019). "A stacking model using url and html features for phishing webpage detection." *Future Generation Computer Systems*, 94, 27–39.

Lin, M.-S., Chiu, C.-Y., Lee, Y.-J. and Pao, H.-K. (2013). "Malicious url filtering—a big data application." In *big data, 2013 IEEE international conference on*, IEEE, 589–596.

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*, ACM, 1245–1254.

Ma, X. and Hovy, E. (2016). "End-to-end sequence labeling via bi-directional lstm-cnns-crf." *arXiv preprint arXiv:1603.01354*.

Mao, J., Tian, W., Li, P., Wei, T. and Liang, Z. (2017). "Phishing-alarm: Robust and efficient phishing detection via page component similarity." *IEEE Access*, 5, 17020–17030.

Marchal, S., Armano, G., Gröndahl, T., Saari, K., Singh, N. and Asokan, N. (2017). "Off-the-hook: an efficient and usable client-side phishing prevention application." *IEEE Transactions on Computers*, 66(10), 1717–1733.

Marchal, S., François, J., State, R. and Engel, T. (2014). "Phishstorm: Detecting phishing with streaming analytics." *IEEE Transactions on Network and Service Management*, 11(4), 458–471.

Marchal, S., Saari, K., Singh, N. and Asokan, N. (2016). "Know your phish: Novel techniques for detecting phishing sites and their targets." In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, IEEE, 323–333.

Menze, B. H., Kelm, B. M., Splitthoff, D. N., Koethe, U. and Hamprecht, F. A. (2011). "On oblique random forests." In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 453–469.

Mercer, J. (1909). "Functions of positive and negative type, and their connection with the theory of integral equations." *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209, 415–446.

Miyamoto, D., Hazeyama, H. and Kadobayashi, Y. (2008). "An evaluation of machine learning-based methods for detection of phishing sites." In *International Conference on Neural Information Processing*, Springer, 539–546.

Moghimi, M. and Varjani, A. Y. (2016). "New rule-based phishing detection method." *Expert systems with applications*, 53, 231–242.

Mohammad, R. M., Thabtah, F. and McCluskey, L. (2012). "An assessment of features related to phishing websites using an automated technique." In *Internet Technology And Secured Transactions, 2012 International Conference for*, IEEE, 492–497.

Mohammad, R. M., Thabtah, F. and McCluskey, L. (2014a). "Intelligent rule-based phishing websites classification." *IET Information Security*, 8(3), 153–160.

Mohammad, R. M., Thabtah, F. and McCluskey, L. (2014b). "Predicting phishing websites based on self-structuring neural network." *Neural Computing and Applications*, 25(2), 443–458.

Mohammad, R. M., Thabtah, F. and McCluskey, L. (2015). "Tutorial and critical analysis of phishing websites methods." *computer science review*, 17, 1–24.

Moore, T. and Clayton, R. (2007). "Examining the impact of website take-down on phishing." In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, ACM, 1–13.

Ndibwile, J. D., Kadobayashi, Y. and Fall, D. (2017). "Unphishme: Phishing attack detection by deceptive login simulation through an android mobile app." In *Information Security (AsiaJCIS), 2017 12th Asia Joint Conference on*, IEEE, 38–47.

Nguyen, L. A. T., To, B. L., Nguyen, H. K. and Nguyen, M. H. (2014). "A novel approach for phishing detection using url-based heuristic." In *Computing, Management*

and Telecommunications (ComManTel), 2014 International Conference on, IEEE, 298–303.

Ni, S., Qian, Q. and Zhang, R. (2018). "Malware identification using visualization images and deep learning." *Computers & Security*, 77, 871–885.

Ollmann, G. (2004). *The phishing guide*, Next Generation Security Software Limited.

Orunsolu, A. A., Alaran, M. A., Adebayo, A. A., Kareem, S. O., Oke, A. et al. (2017). "A lightweight anti-phishing technique for mobile phone." *Acta Informatica Pragensia*, 6, 114–123.

Pan, Y. and Ding, X. (2006). "Anomaly based web phishing page detection.." In *Proceedings - Annual Computer Security Applications Conference, ACSAC*, volume 6, 381–392.

Patgiri, R., Katari, H., Kumar, R. and Sharma, D. (2019). "Empirical study on malicious url detection using machine learning." In *International Conference on Distributed Computing and Internet Technology*, Springer, 380–388.

Patil, D. R. and Patil, J. (2018). "Malicious urls detection using decision tree classifiers and majority voting technique." *Cybernetics and Information Technologies*, 18(1), 11–29.

Prakash, P., Kumar, M., Kompella, R. R. and Gupta, M. (2010). "Phishnet: Predictive blacklisting to detect phishing attacks." In *INFOCOM, 2010 Proceedings IEEE*, IEEE, 1–5.

Ramesh, G., Krishnamurthi, I. and Kumar, K. S. S. (2014). "An efficacious method for detecting phishing webpages through target domain identification." *Decision Support Systems*, 61, 12 – 22.

Ranganayakulu, D. and Chellappan, C. (2013). "Detecting malicious urls in e-mail–an implementation." *AASRI Procedia*, 4, 125–131.

Rao, C. R. and Mitra, S. K. (1972). "Generalized inverse of a matrix and its applications." In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics*, The Regents of the University of California.

Rao, R. S. and Ali, S. T. (2015a). "A computer vision technique to detect phishing attacks." In *Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on*, IEEE, 596–601.

Rao, R. S. and Ali, S. T. (2015b). "Phishshield: A desktop application to detect phishing webpages through heuristic approach." *Procedia Computer Science*, 54, 147–156.

Rao, R. S. and Pais, A. R. (2017). "An enhanced blacklist method to detect phishing websites." In *International Conference on Information Systems Security*, Springer, 323–333.

Rao, R. S. and Pais, A. R. (2019a). "Detection of phishing websites using an efficient feature-based machine learning framework." *Neural Computing and Applications*, 31(8), 3851–3873.

Rao, R. S. and Pais, A. R. (2019b). "Jail-phish: An improved search engine based phishing detection system." *Computers & Security*, 83, 246–267.

Rao, R. S. and Pais, A. R. (2019c). "Two level filtering mechanism to detect phishing sites using lightweight visual similarity approach." *Journal of Ambient Intelligence and Humanized Computing*, 1–20.

Rao, R. S., Vaishnavi, T. and Pais, A. R. (2019). "Phishdump: A multi-model ensemble based technique for the detection of phishing sites in mobile devices." *Pervasive and Mobile Computing*, 60, 101084.

Rao, R. S., Vaishnavi, T. and Pais, A. R. (2020). "Catchphish: detection of phishing websites by inspecting urls." *Journal of Ambient Intelligence and Humanized Computing*, 11(2), 813–825.

Ren, Y., Zhang, L. and Suganthan, P. N. (2016). "Ensemble classification and regression-recent developments, applications and future directions [review article]." *IEEE Computational Intelligence Magazine*, 11(1), 41–53.

Rosiello, A. P., Kirda, E., Ferrandi, F. et al. (2007). "A layout-similarity-based approach for detecting phishing pages." In *Security and Privacy in Communications Networks and the Workshops, 2007. SecureComm 2007. Third International Conference on*, IEEE, 454–463.

RSA (2013). "Rsa fraud report." `https://www.emc.com/collateral/fraud-report/rsa-online-fraud-report-012014.pdf`). Accessed: 2016-07-15.

Sahingoz, O. K., Buber, E., Demir, O. and Diri, B. (2019). "Machine learning based phishing detection from urls." *Expert Systems with Applications*, 117, 345–357.

Salton, G. and McGill, M. J. (1986). "Introduction to modern information retrieval." .

Selvaganapathy, S., Nivaashini, M. and Natarajan, H. (2018). "Deep belief network based detection and categorization of malicious urls." *Information Security Journal: A Global Perspective*, 27(3), 145–161.

Sengar, P. and Kumar, V. (2010). "Client-side defense against phishing with pagesafe." *International Journal of Computer Applications*, 4(4), 6–10.

Shirazi, H., Bezawada, B. and Ray, I. (2018). "Kn0w thy doma1n name: Unbiased phishing detection using domain name based features." In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*, ACM, 69–75.

Shirazi, H., Haefner, K. and Ray, I. (2017). "Fresh-phish: A framework for auto-detection of phishing websites." In *Information Reuse and Integration (IRI), 2017 IEEE International Conference on*, IEEE, 137–143.

Srinivasa Rao, R. and Pais, A. R. (2017). "Detecting phishing websites using automation of human behavior." In *Proceedings of the 3rd ACM Workshop on Cyber-*

Physical System Security, CPSS@AsiaCCS, CPSS '17, ACM, New York, NY, USA, 33–42.

Su, K.-W., Wu, K.-P., Lee, H.-M. and Wei, T.-E. (2013). "Suspicious url filtering based on logistic regression with multi-view analysis." In *Information Security (Asia JCIS), 2013 Eighth Asia Joint Conference on*, IEEE, 77–84.

Tan, C. L., Chiew, K. L., Wong, K. and Sze, S. N. (2016). "Phishwho: Phishing webpage detection via identity keywords extraction and target domain name finder." *Decision Support Systems*, 88, 18 – 27.

Thomas, K., Grier, C., Ma, J., Paxson, V. and Song, D. (2011). "Design and evaluation of a real-time url spam filtering service." In *Security and Privacy (SP), 2011 IEEE Symposium on*, IEEE, 447–462.

Vapnik, V. N. and Vapnik, V. (1998). *Statistical learning theory*, volume 1, Wiley New York.

Varshney, G., Misra, M. and Atrey, P. K. (2016a). "Improving the accuracy of search engine based anti-phishing solutions using lightweight features." In *Internet Technology and Secured Transactions (ICITST), 2016 11th International Conference for*, IEEE, 365–370.

Varshney, G., Misra, M. and Atrey, P. K. (2016b). "A phish detector using lightweight search features." *Computers & Security*, 62, 213 – 228.

Varshney, G., Misra, M. and Atrey, P. K. (2016c). "A survey and classification of web phishing detection schemes." *Security and Communication Networks*, 9(18), 6266–6284.

Verma, R. and Dyer, K. (2015). "On the character of phishing urls: Accurate and robust statistical learning classifiers." In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, ACM, 111–122.

Wang, W. and Shirley, K. (2015). "Breaking bad: Detecting malicious domains using word segmentation." *arXiv preprint arXiv:1506.04111*.

Weiss, K. R. and Khoshgoftaar, T. M. (2017). "Detection of phishing webpages using heterogeneous transfer learning." In *Collaboration and Internet Computing (CIC), 2017 IEEE 3rd International Conference on*, IEEE, 190–197.

Whittaker, C., Ryner, B. and Nazif, M. (2010). "Large-scale automatic classification of phishing pages." In *NDSS '10*.

Wu, L., Du, X. and Wu, J. (2016). "Effective defense schemes for phishing attacks on mobile computing platforms." *IEEE Transactions on Vehicular Technology*, 65(8), 6678–6691.

Wu, M., Liu, F. and Cohn, T. (2018). "Evaluating the utility of hand-crafted features in sequence labelling." *arXiv preprint arXiv:1808.09075*.

Xiang, G., Hong, J., Rose, C. P. and Cranor, L. (2011). "Cantina+: A feature-rich machine learning framework for detecting phishing web sites." *ACM Transactions on Information and System Security (TISSEC)*, 14(2), 21.

Xiang, G. and Hong, J. I. (2009). "A hybrid phish detection approach by identity discovery and keywords retrieval." In *Proceedings of the 18th international conference on World wide web*, ACM, 571–580.

Xu, L., Zhan, Z., Xu, S. and Ye, K. (2013). "Cross-layer detection of malicious websites." In *Proceedings of the third ACM conference on Data and application security and privacy*, ACM, 141–152.

Yi, P., Guan, Y., Zou, F., Yao, Y., Wang, W. and Zhu, T. (2018). "Web phishing detection using a deep learning framework." *Wireless Communications and Mobile Computing*, 2018.

Yu, B., Pan, J., Hu, J., Nascimento, A. and De Cock, M. (2018). "Character level based detection of dga domain names." In *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 1–8.

Zhang, D., Yan, Z., Jiang, H. and Kim, T. (2014). "A domain-feature enhanced classification model for the detection of chinese phishing e-business websites." *Information & Management*, 51(7), 845–853.

Zhang, H., Liu, G., Chow, T. W. and Liu, W. (2011). "Textual and visual content-based anti-phishing: a bayesian approach." *IEEE Transactions on Neural Networks*, 22(10), 1532–1546.

Zhang, J., Porras, P. A. and Ullrich, J. (2008). "Highly predictive blacklisting.." In *USENIX Security Symposium*, 107–122.

Zhang, L. and Suganthan, P. N. (2014). "Random forests with ensemble of feature spaces." *Pattern Recognition*, 47(10), 3429–3437.

Zhang, L. and Suganthan, P. N. (2015). "Oblique decision tree ensemble via multisurface proximal support vector machine." *IEEE transactions on cybernetics*, 45(10), 2165–2176.

Zhang, W., Jiang, Q., Chen, L. and Li, C. (2017). "Two-stage elm for phishing web pages detection using hybrid features." *World Wide Web*, 20(4), 797–813.

Zhang, Y., Hong, J. I. and Cranor, L. F. (2007). "Cantina: a content-based approach to detecting phishing web sites." In *Proceedings of the 16th international conference on World Wide Web*, ACM, 639–648.

Zhao, J., Wang, N., Ma, Q. and Cheng, Z. (2018). "Classifying malicious urls using gated recurrent neural networks." In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Springer, 385–394.

Zuhair, H., Selamat, A. and Salleh, M. (2016). "New hybrid features for phish website prediction." *International Journal of Advances in Soft Computing & Its Applications*, 8(1).

PUBLICATIONS BASED ON THE RESEARCH WORK

1. Rao, R. S. and Pais, A. R. (2019). Detection of phishing websites using an efficient feature-based machine learning framework. *Neural Computing and Applications (Springer)*, 1-23. [**DOI**: https://doi.org/10.1007/s00521-017-3305-0]

2. Rao, R. S. and Pais, A. R. (2019). Jail-Phish: An improved search engine based phishing detection system. *Computers & Security (Elsevier)*, 83:246-267. [**DOI**: https://doi.org/10.1016/j.cose.2019.02.011]

3. Rao, R. S., Vaishnavi T. and Pais, A. R. (2020). CatchPhish: Detection of Phishing websites by inspecting URLs. *Journal of Ambient Intelligence and Humanized Computing (Springer)*, 1-13 [**DOI:** https://doi.org/10.1007/s12652-019-01311-4]

4. Rao, R. S., Vaishnavi T and Pais, A. R. (2019). PhishDump: A Multi-model ensemble based technique for the detection of phishing sites in mobile devices. *Pervasive and Mobile Computing (Elsevier)* [**DOI:** https://doi.org/10.1007/s12652-019-01311-4]

5. Rao, R. S. and Pais, A. R. (2017). An Enhanced Blacklist Method to Detect Phishing Websites. *In International Conference on Information Systems Security (ICISS-2018)* [**DOI:** https://doi.org/10.1007/978-3-319-72598-7_20]

6. Rao, R. S. and Pais, A. R. (2017). Detecting Phishing Websites Using Automation of Human Behavior. *In Proceedings of the 3rd ACM Workshop on Cyber-*

Physical System Security, (CPSS@AsiaCCS-2018) [**DOI:** `https://doi.org/10.1145/3055186.3055188`]

7. Rao, R. S. and Pais, A. R. (2019). A heuristic technique to detect phishing websites using TWSVM classifier. *Neural Computing and Applications (Springer)* [**Under review**]

8. Rao, R. S. and Pais, A. R. (2019). Two level filtering mechanism to detect phishing sites using lightweight visual similarity approach. *Journal of Ambient Intelligence and Humanized Computing (Springer)*, 1-15 [**https://doi.org/10.1007/s12652-019-01637-z**]

BIO-DATA

Name:	Routhu Srinivasa Rao
Date of Birth:	10/08/1990
Gender:	Male
Marital Status:	Single
Father's Name:	Kamu Naidu
Mother's Name:	Kantha
Email Id:	routh.srinivas@gmail.com
Present Address:	481/2, 3rd floor, old oil mill road, 1st Main Rd, Shubodaya layout, Near Royal Enclave Apartments, Marathahalli, Bengaluru - 560037
Educational Qualifications:	B.Tech (CSE) - SRKR Engineering college, Chinnamiram, Bhimavaram, West Godavari, Andhra Pradesh 534204 M.Tech (Computer Engineering) - National Institute of Technology Kurukshetra, Kurukshetra, Haryana 136119
Areas of Interest:	Information Security, Cyber Security, Phishing, Machine Learning.