A Mini Project Report on

# PHISHING WEBSITE DETECTION SYSTEM

Submitted to
The Department of Information Technology
In partial fulfillment of the academic requirements of
Jawaharlal Nehru Technological University
For
The award of the degree of

Bachelor of Technology
in
Information Technology

By


| | |
|---|---|
| G.MAHIRATHAN | 18311A1214 |
| M.ABHINAV | 18311A1233 |
| P.SAKETH | 18311A1242 |


Under the Guidance of
Dr.Sunil Bhutada



Sreenidhi Institute of Science and Technology
Yamnampet, Ghatkesar, R.R. District, Hyderabad - 501301


Affiliated to
Jawaharlal Nehru Technology University
Hyderabad - 500085

Department of Information Technology

Sreenidhi Institute of Science and Technology

The Department of Information Technology


# CERTIFICATE

This is to certify that this Project report on **"PHISHING WEBSITE DETECTION SYSTEM**", submitted by G.MAHIRATHAN(18311A1214), M.ABHINAV (18311A1233) ,P.SAKETH(18311A1242) in the year 2022 in partial fulfillment of the academic requirements of Jawaharlal Nehru Technological University for the award of the degree of Bachelor of Technology in Information Technology, is a bonafide work that has been carried out by them as part of their Mini project (2021), under our guidance. This report has not been submitted to any other institute or university for the award of any degree.

Dr. Sunil Bhutada
Prof & Head,
Department of IT
Internal guide

Dr. Sunil Bhutada
Prof & Head,
Department of IT

,

Dr. Sunil Bhutada
Prof & Head,
Department of IT
Project Coordinator

External Examiner

# DECLARATION

We, **G.MAHIRATHAN(18311A1214), M.ABHINAV(18311A1233) and P.SAKETH(18311A1242),** students of **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY, YAMNAMPET, GHATKESAR, of INFORMATION TECHNOLOGY** solemnly declare that the project work, titled **"PHISHING WEBSITE DETECTION SYSTEM"** is submitted to **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY** for partial fulfillment for the award of degree of Bachelor of technology in **INFORMATION TECHNOLOGY**.

It is declared to the best of our knowledge that the work reported does not form part of any dissertation submitted to any other University or Institute for award of any degree

# ACKNOWLEDGEMENT

I would like to express my gratitude to all the people behind the screen who helped me to transform an idea into a real application.

I would like to express my heart-felt gratitude to my parents without whom I would not have been privileged to achieve and fulfill my dreams. I am grateful to our principal, **Dr. T. Ch. Siva Reddy,** who most ably run the institution and has had the major hand in enabling me to do my project.

I profoundly thank **Dr. Sunil Bhutada,** Head of the Department of Information Technology who has been an excellent guide and also a great source of inspiration to my work.

I would like to thank my internal guide **Dr. Sunil Bhutada** for technical guidance, constant encouragement and support in carrying out my project at college.
I would like to thank my coordinator Dr. Sunil Bhutada **,** for his technical guidance, constant encouragement and support in carrying out my project at college.

The satisfaction and euphoria that accompany the successful completion of the task would be great but incomplete without the mention of the people who made it possible with their constant guidance and encouragement crowns all the efforts with success. In this context, I would like thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased my task.

**G. MAHIRATHAN 18311A1214**
**M. ABHINAV 18311A1233**
**P. SAKETH 18311A1242**

# ABSTRACT

Online phishing is one of the most epidemic crime schemes of the modern Internet. A common countermeasure involves checking URLs against blacklists of known phishing websites, which are traditionally compiled based on manual verification, and is inefficient. Thus, as the Internet scale grows, automatic URL detection is increasingly important to provide timely protection to end users. In this thesis, we propose an effective and flexible malicious URL detection system with a rich set of features reflecting diverse characteristics of phishing webpages and their hosting platforms, including features that are hard to forge by a miscreant. Using Random Forests algorithm, our system enjoys the benefit of both high detection power and low error rates. Based on our knowledge, this is the first study to conduct such a large-scale websites/URLs scanning and classification experiments taking advantage of distributed vantage points for feature collection.One of the extra feature added to this is the listing of various aspects of the website such as keywords,provider,author,e.t.c.Users can also visit the website if it's a legitimate website.

# Table of Contents

# 1. INTRODUCTION

## 1.1 Phishing Definition

Phishing is an online crime that tries to trick unsuspected users to expose their sensitive (and valuable) personal information, for example, usernames, passwords, financial account details, personal addresses, SSN, and social relationships, to the miscreant, of- ten for malicious reasons. Phishing is normally perpetrated by disguising as a trustworthy entity in Internet communication which is achieved by combining both social engineering and technical tricks. The instruments frequently exploited by attackers includes sending out spoofing emails and putting up deceiving websites to entice users to expose information. The spoofing emails usually purport to be from legal businesses, intended to lead users to counterfeit websites that lure the user to input sensitive information.

## 1.2 Current Phishing Landscape

Currently, a lot of existing tools, encapsulated in browsers, search engines or applications, such as Safe Browsing from Google and SmartScreen from Microsoft, try to inform a user that a specific URL the user is about to visit has been identified as unsafe or malicious. This is realized by matching the URL being visited with blacklists constructed by the security community. Those blacklists are accumulated using various techniques, ranging from user reporting to web crawlers with site content analysis to automatic classification based on heuristics or machine learning classifiers. However, many malicious websites can still sneak through such protection systems, which can be the consequence of a number of reasons:

(1) The website is too new and thus has not been scanned or analyzed by any mechanisms yet.

(2) The website has been incorrectly analyzed, either due to the imperfection of mechanisms or the countermeasures against detection taken by the attackers,
e.g. “cloaking”, or abusing the legal short URL services.

There do exist some systems aiming at addressing the issue of incomplete blacklists by using real-time client-side evaluation against the content or behavior of the website when an end user visits it. However, those systems suffer from run-time overhead. Be- sides, depending on the nature of the attack, clients may have already been exposed to the threats of such malicious websites since the contents have been downloaded before the analysis begins.

The most critical component of defense against phishing is accurate detection of phishing websites in a timely manner. Successful recognition and blacklisting of phishing URLs would result in end users receiving a warning while being deceived to visit the phishing site. Once a striking warning such as the one presented in Figure 1.3 is displayed, it is highly likely that users would decline the login/data-input requests or malicious payload- downloading popups in phishing sites.

## 1.3.Our Work

Hence, in this thesis, we propose an effective detection system that crawls websites and

automatically discovers malicious pages. We intend our system to be used by a blacklist provider who can automatically compile and maintain an up-to-date blacklist of malicious URLs. Our system is equipped with a plentiful set of features that reflect various types of essential characteristics of the webpage content or behavior, which are impossibleor difficult to be camouflaged by the miscreants. This system can proactively crawl and evaluate a given URL, labeling it to be phishing/malicious or legitimate, based on a trained classifier. Further, crawling is done from distributed vantage points, which allows the system to collect novel features and achieve higher accuracy and quicker recognition speed. By avoiding manual analysis, the blacklist can achieve better cover- age and timeliness. Also, client-side analysis is avoided, thus removing both the runtime overhead and risk of downloading suspicious content to personal device.

Specifically, in this thesis, we propose a novel phishing detection system viewing the phishing detection task as a binary classification problem where the positive class is phishing and negative class is legitimate, which utilizes supervised learning algorithm employing a rich set of features, some of which are hard or impossible to collect from

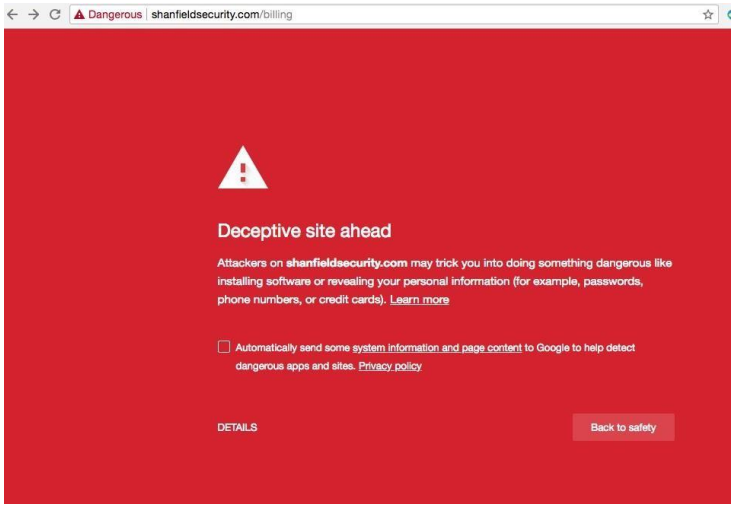


Figure 1.3.1. Deceptive Site Warning of Chrome

the client side. Our end-to-end system empowers the blacklist provider to proactively detect and update phishing URL blacklist promptly with high accuracy. The set of features utilized by the machine learning algorithm is derived from a broad scope of information crawled by a distributed system of widely dispersed vantage points. Those features are extracted from raw probing data including the URL string, HTTP headers, DNS records, server status, network traces, WHOIS information, and redirection process. We demonstrate that by taking advantage of carefully selected features that indicate

diverse aspects of the phishing content distributor, including physical characteristics, network characteristics and programming implementation characteristics, our system is robust to evasion attempts of miscreants

# 2. LITERATURE SURVEY

## 2.1 Problem Statement:

Our work in this thesis focuses mainly on detecting phishing websites with machine learning. There has been quite some effort regarding similar topics such as malicious domain blacklisting and email spam filtering. Furthermore, it is increasingly popular to utilize machine learning in these areas. Existing malicious websites detection approaches can be mainly divided into two categories based on the features leveraged: static feature based approaches and dynamic feature based approaches. Static feature based approaches relyon features extracted from the URL, page content, HTML DOM structure, domain-based information(such as WHOIS and DNS records) and so on. Alternatively, dynamic feature based solutions primarily focus on analyzing behaviors captured when the page is loaded and rendered, or investigating system logs when some scripts are executed. In this thesis, we concentrate on exploiting static features.

## 2.2 Machine Learning Based Methods

Dongetal have investigated a comprehensive list of features that can be extracted from X.509 certificates. With using top 100,000 websites from Alexa top one million web- sites as their legitimate examples and phishing URLs downloaded from Phish Tank as phishing examples, precision and recall of 95.5% and 93.7% are attained for the phishing category with a Random Forests classifier. However, these statistics are doubtful when it comes to the current Internet environment, as will be discussed . Our feature list, in contrast to their work, leverages information from not only certificates, but also several other dimensions of a given website, including server characteristics, DNS responses, network performance and so on.

There are some severe drawbacks for the methods relying on DOM or lexical information. First of all, lexical characteristics of URL or HTML content can be easily manipulated by the attacker to deceive the classification system. Meanwhile, in order to perform the classification, the page con- tent needs to be downloaded, during which the malicious component might already be introduced and downloaded to the user machine if those methods are used within the user's browser. Additionally, as has been shown by several studies, cloaking technology(cloaking means the website serves different content to different visitors based on pre-defined criteria for the seek of bypassing the security system or deceiving the crawler) is used by a non- negligible portion of malicious websites for the purpose of misleading crawler services like Google to put them on higher rank or getting better exposure in the search result. This phenomenon makes the PageRank or keyword search results which CANTINA+ depends on less reliable.

Metal attempted to circumvent this issue by not including any page content related features for classification. Beyond that, they also took advantage of tens of thousands of features generated from IP address, WHOIS records and domain names, with a majority of features being lexical features constructed using "bag of words" technique. One drawback of their method is that the resulting model is hard to interpret due to the massive number of algorithmically generated lexical features. On the contrary, features within our system are all comprehensible, making the resulting classifier explainable.

## 2.3 Importance Of Machine Learning:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and "more items to consider" and "get yourself a little something" on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today's data-rich world.

Machines can aid in filtering useful pieces of information that help in major

advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that's in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works

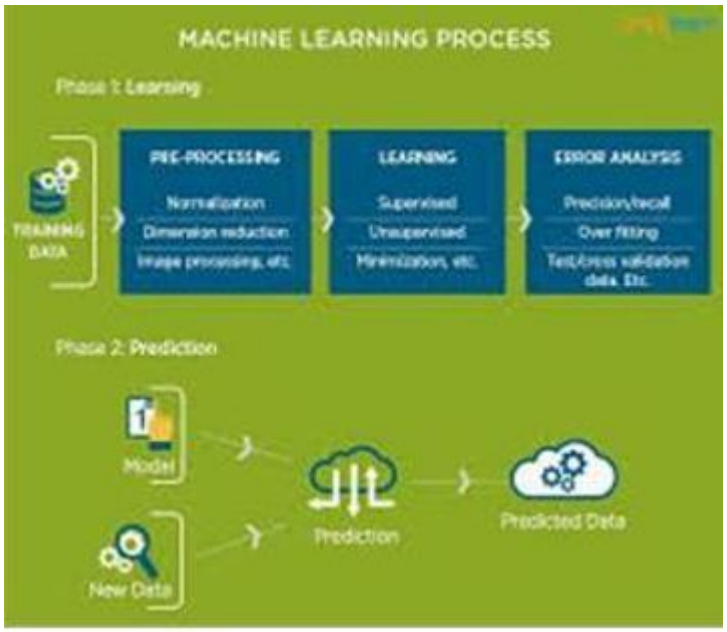


Figure 2.3.1 the Process Flow

## 2.4 TYPES OF LEARNING ALGORITHMS:

**2.4.1** Supervised Learning
**2.4.2** Unsupervised Learning
**2.4.3** Semi Supervised Learning

## 2.5 Purpose and other aspects

### 2.5.1 Purpose:

Phishing is a form of fraud in which the attacker tries to learn sensitive information such as login credentials or account information by sending as a reputable entity or personal infomation or other communication channels. Typically a victim receives a message that appears to have been sent by a known contact or organization.

### 2.5.2 Adavantages:

The message contains malicious software targeting the user's computer or has links to direct victims to malicious websites in order to trick them into divulging personal and financial information, such as passwords, account IDs or credit card details.We can take measurements for our safety.

### 2.5.3 Non-Functional Requriments:

Lets check the URL structure for the clear understanding of how attackers think when they create a phishing domain.Uniform Resource Locator (URL) is created to address web pages. The figure below shows relevant parts in the structure of a typical URL.

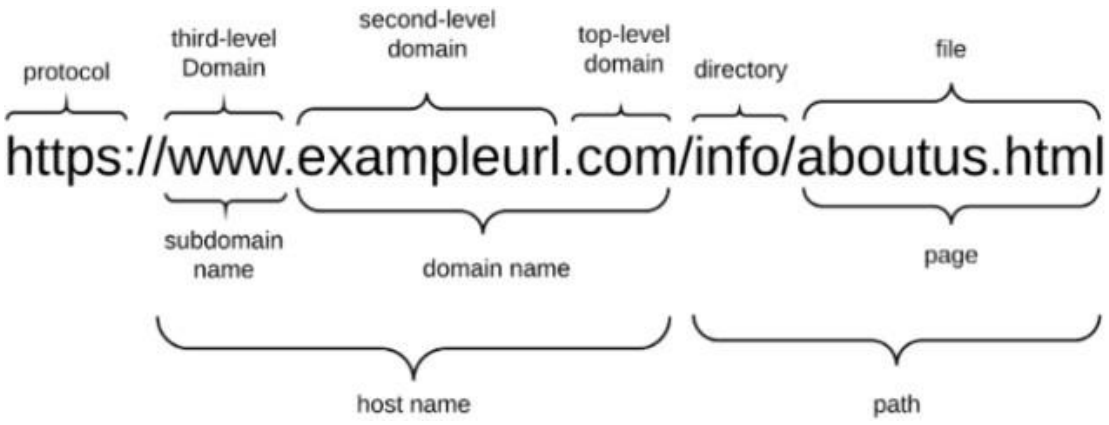


Figure 2.5.3.1 Characteristics of Phishing Domains

## 2.6 Technologies Used:

**Hardware requirements:**

Processer : Any Update Processer
Ram : Min 4 GB
Hard Disk : Min 100 GB

**Software requirements:**

| | | |
|---|---|---|
| Operating System | : | Windows family |
| Technology | : | Python 3.6 |
| IDE | : | Visual Studio |

# 3 ANALYSIS

In this section, we present the features we selected for the machine learning algorithm. Effective machine learning highly relies on the discerning of an informative set of features. Our feature selection starts with observations of the natural behavioral differences between phishing websites and legitimate websites, selected carefully from both application and infrastructure level. We also absorb useful features and concepts from previous works on related topics such as phishing detection, proactive blacklist generation and email spam detection. Eventually, our feature set can be divided into eight groups based on their origin. The intuition and meaning behind each feature will be discussed in this chapter.

## 3.1 Certificate Features

As mentioned above, X.509 digital certificates are extensively used in modern Internet communication to provide end-to-end identity authentication. However, except for the traditional client-side verification of the certificate itself, the information embedded in the certificate can provide a large number of useful features to deduce the nature of the server. An extensive list of features that can be extracted solely from the X.509 digital certificate has already been proposed by Dong et al. Our certificate feature set is gathered by reusing most of their features and logical groups, while adding some new ones. Meanwhile, we have also removed some of their features due to their deficiencies in practice. For example, is Highly Trusted CA(Issuer), which checks if the issuer of the certificate is VeriSign or Thawte, is removed since it is hard to judge which CA can be indeed more highly trusted than others in the root store, and also because this feature would throw off the classifier if one of these highly trusted CAs is compromised.

## 3.2 Certificate Extensions

Certificate extensions are a list of fields that are optional for a valid certificate. Although a clear standard does not exist, extensions can still potentially reveal critical underlying patterns of fraudulent websites potentially. Certificate extensions could include extra information such as alternative names for the issuer and subject, certificate and public key usage policies, Certificat e Revocation List (CRL), CRL distribution points, extended validated certificate, and so on. The features are retrieved in the following way. Firstly, the total number of extensions is recorded as a numerical feature. Then the existence of 13 most frequently used extensions is checke d, each kept as a Boolean feature. The following is the list of features retrieved from certificate extensions:

Authority info access: If the authority Info Access field exists in the certificate's extension.

authority_key_identifier: If the authority Key Identifier field exists in the certificate's extension.

subject_key_identifier: If the subject Key Identifier field exists in the certificate's extension.

basic constraints: If the basic Constraints field exists in the certificate's extension.

certificate_policies: If the certificate Policies field exists in the certificate's extension.

extended_key_usage: If the extended Key Usage field exists in the certificate's ex tension.

CRL_distribution_points: If the crl Distribution Points field exists in the certificate's extension.

freshest_CRL: If the freshest CRL field exists in the certificate's extension.

is_extended_validation: If the is Extended Validation field exists in the certificate's extension.

key_usage: If the key Usage field exists in the certificate's extension.
issuer_alt_name: Ifthe issuer Alt Name field exists in the certificate's extension.
subject_alt_name: If the subject Alt Name field exists in the certificate's extension.
subject_directory_attributes: If the subject Directory Attributes field exists in the certificate's extension.
extension_count: The total number of extension fields contained in this certificate.

## 3.3 Issuer and Subject/Domain Information

The two most important fields within a certificate are the Issuer and Subject. Issuer refers to the CA who signs and certifies the identity of the current domain using the certificate. Subject refers to the domain or organization who isbeing certifiedby thedigital certificate. Both Issuer and Subject share the same schema, known as 'Distinguished Names' (DNs), containing the detailed name, location, and contact information of the issuer and the subject, respectively. They each can provide a set of subfields including country, state, city, common name, organization name, organization unit, and email. Three groups of features are developed from the subfields of Issuer and Subject.

**Whitelisting and Blacklisting for Issuer and Subject.** This group contains three features, is_trusted_issuer, is_prohibited_issuer and is_prohibited_subject. is_trusted_issuer indicates whether the issuer of this certificate is one of the CAs trusted by Mozilla certificate store. As mentioned above, in X.509 PKI, all CAs are trusted equally, allowing any CA to issue a digital certificate to any website. Hence including clues of the certificate issuer's reputation could be useful for phishing detection. A list of CAs trusted by Mozilla,one of the primary browser providers,is used as the set of trusted issuers, considering that itrepresents the state of the art certificate validation experience. We argue that is_trusted_issuer is more feasible than is Highly Trusted CA(Issuer) because, to the best of our knowledge, there is no previous work showing that the two "highly trusted" CAs (i.e., VeriSign and Thawte) chosen by Dong et al. are superior to other CAs existed in Mozilla trust store. Moreover, if any accident led to those highly trusted CA being compromised, such as the Digi Notar and Comodo case, is Highly Trusted CA(Issuer) would be harmful to the classification.

**Relationship Between Subfields.** There is no clear standard published regarding the content and relationship that a CA should follow for the subfields inside Issuer and Subject section of a certificate when signing a digital certificate. However, checking the patterns revealed by the relationships between subfields can still be helpful to discover different CA characteristics. Moreover, those patterns could even serve as evidence to detect technically valid certificate that is signed by unethical or compromised CAs, as is mentioned in a report from Microsoft in 2014.

For example, the similarity between the common name of Issuer and Subject and the similarity between the common name of Issuer and domain name of website is good indication for whether this is a self-signed certificate or not, even if the trust chain of the certificate could be validated. High similarities insinuate a self-signed certificate is being used. Nevertheless, we do need to note that large-scale organizations might operate their own CA. For example, Microsoft and Google both run intermediate CAs and issue themselves certificates. Figure 3.3.1 shows the trust chain of a certificate belonging to Microsoft, in which the Issuer and subject are both Microsoft, generated via the Chrome browser.

Another good differentiator between legitimate and suspicious sites is the similarity between the domain name and name of the certificate Subject. In accordance with RFC6125 and common practice, the certificate presented by a domain, in order to certify its identity, must include the corresponding domain name either in the Subject field or the Subject Alternative Name extension. Therefore, if the domain name matches with the common name of the Subject or Subject Alternative Name in the extension could be inferred as a direct hint of trustworthiness.

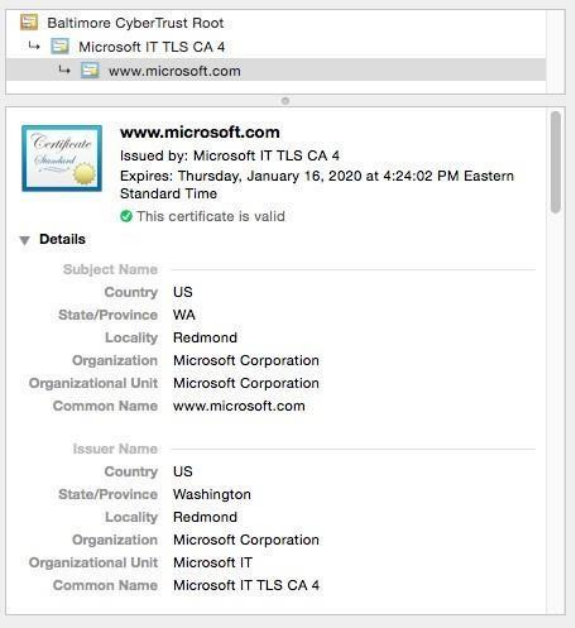


Figure 3.3.1. Microsoft's Own Certificate Authority

In addition, a high similarity between the organization unit and common name im- plies a legitimate site as well, since the two fields are supposed to refer to the same entity. We measure similarity between string-valued fields using matching ratios, which in this thesis are all

Calculated using the default sequence matching function provided by diff lib of python2.7. In the following we present the list of features corresponding to the relationship be- tween subfields.

match_issuer_o_cn: Matching ratio of organization and common name of certificate issuer.

match_issuer_o_ou: Matching ratio of organization and organization unit of certificate issuer.

match_subject_o_cn: Matching ratio of the organization and common name of certificate subject.

match_subject_o_ou: Match ratio of the organization and organization unit of certificate subject.

match_issuer_subject_cn: Matching ratio of the common name of certificate is- suer and subject.

match_website_issuer_cn: Matching ratio of website domain with issuer's com- mon name.

match_website_altname: Highest matching ratio of website domain with the each of the alternative name in the list of alternative names field of certificate extension.

match_website_subject_cn: Matching ratio of website domain and subject's com- mon name.

## 3.4 Chronological Information

Not Before and Not After are two fields in a certificate specifying its validity period. As a common practice, a certificate with a valid period that is too long is considered as less secure. Hence the time difference between Not Before and Not After is used as a feature. In addition to that, the time differences between the time when we obtain the certificate (referred to as "downloading time" of the certificate below) and those two fields are also calculated. The difference between the downloading time and Not Before, to some extent, could serve as an indicator for suspiciousness of the domain, with the intuition that certificates which have been used for a while are more credible than a newly issued one. We also check if the certificate has expired, that is, being captured after the time specified in its Not After field, as this behavior is illegal and highly suspicious.

## 3.5 Server Status Features

Whatever methodologies are used for phishing websites to deceive end users, the miscreant needs to host them on some physical servers. Those physical servers, which are used for short- live malicious purposes, are possibly botnets or hosts that are not maintained professionally. Thus, features reflecting the status of the physical server could be extracted to detect malicious sites. These features are especially desirable because they are hard for a miscreant to circumvent.

## 3.6 Port Status

The status of each port (i.e., whether it is open or closed) is recorded as a binary feature. Generally, legitimate servers could be expected to have well-defined pattern of which ports they open and a limited number of open ports overall for the sake of security, while malicious websites would be held on servers or botnets with inferior or no maintenance at all, thus leaving more ports open by default. Besides, those malicious servers could potentially host a lot of different malicious sites and applications at the same time, leaving more ports open for different services or domains, due to budget limitation or other reasons. On the other hand, we can speculate that legitimate servers are usually dedicated to a small number of tasks in production, restricting the total number of open ports. Thereby the total number of open ports is also kept as a feature.

## 3.7 Average Distance Measurement

As modern websites are paying more and more attention to high availability and scalability, domain administrators have greater motivation on migrating to a distributed hosting infrastructure, e.g., CDN or Cloud Service. This trend, as a consequence, could be used for distinguishing malicious websites, as miscreants may not have much motivation to set up distributed hosting infrastructure for a short-lived phishing website, because of both the effort and costs. We mainly recruit three features with the purpose of measuring both network and physical distances between our vantage points and the servers. The features included are average geographical distance, average RTT and average hop distance.

## 3.8 HTTP Header Features

The HTTP header is returned by an HTTP server to the client along with the requested contents, which allows additional information corresponding to the requests and responses to be passed to the client. The HTTP header is a list of key-value pairs of which the possible keys and values are registered with IANA[46]. The content of the HTTP headers can be used to induce some clues regarding the underlying server implementation details, for example, cache control policy of the server, web framework used by the server, access control policy of the server, content type of the response. Out of the HTTP header content, three features that can be easily parsed are extracted.

Firstly, we counted the total number of different header fields in the server response. The intuition behind this feature is to acquire differences in underlying patterns in the HTTP response of malicious and legitimate websites. Considering that malicious web- sites are usually maintained with less effort, their HTTP header could be the default values of the web framework used by the miscreant. However, headers returned by a legitimate server could be developed with clear purposes.

## 3.9 Network Features

Network traffic and network-level characteristics between the vantage points and the target server can be a good indication of the underlying connection quality as well as the ability of the server to handle user requests. These features were investigated by Ouyang , my collaborator on our overall effort to detect malicious URLs. Thus, we only present a brief description of the network features here. Readers who are interested in learning more about the network features can refer to Ouyang's previous work for details.

ttl: TCP TTL value retrieved from the SYN/ACK packet accepted by the local vantage point.

advertised_window_size: Size of the TCP advertised window field retrieved from SYN/ACK packet accepted by the local server.

3_way_handshake: Time interval between sending out SYN packet and sending out ACK packet for the SYN/ACK packet received.

fins_local: Totalnumber ofpackets with TCP FIN flag set emitted by local vantage point.

fins_remote: Total number of packets with TCP FIN flag set accepted by local vantage point.

max_idle_time: The Longest time interval between two consecutive reception of packets on the local vantage point.

variance_packet_arrival_time: The variance of time intervals captured between consecutive receptions of remote packets.

## 3.10 URL Lexical Features

Recruiting URL lexical features for phishing and malicious website detection has been extensively discussed in many previous studies. We reused those lexical features from previous studies since their feasibility in phishing detection has been proved. Note that we only consider the features for domain names, since our negative examples (legitimate websites) only consist of top million domains.

## 3.11 Is IP Address Used as Domain

Sometimes the miscreants try to establish their malicious websites without being able to set up corresponding DNS entries, because of either the effort or the cost needed. The most straightforward solution, in this case, is using the IP address directly as the domain. On the contrary, a legitimate website would never want to do so because this would degrade the user experience, e.g., users would highly prefer visiting www.google.com rather than

172.217.6.14, which is much harder to remember. Thus, using IP addresses the domain name is recorded as a feature.

## 3.12 Characters in Domain

Besides the IP address checking, we also calculate the number of dots, the number dashes, and the length of the domain as features since, according to previous studies, those numbers have different distribution patterns in phishing and legitimate domains.
URL lexical features are presented as follows.

IP_address_in_domain: If the IP address is used directly as the domain name.
num_of_dot: The number of dots (“.”) in the domain.
num_of_dash: The number of dashes (“-”) in the domain.
len_domain: Length of the domain.

## 3.13 WHOIS Features

WHOIS records are the registration details of a given domain, including the creation date, updated date, expiry date, registrar of the domain, registrant of the domain, abuse contact of the domain, DNS servers of the domain and so on.

### 3.13.1 WHOIS Chronological Information

First of all, the time difference in seconds between the downloading timestamp and domain creation time, the downloading timestamp and updated date, the downloading timestamp and expiry date is used as domain_age, diff_update_time and diff_exipry_time, respectively. Considering the fast-iteration nature of phishing websites, the patterns of those chronological values could be distinct from legitimate websites.

### 3.13.2 WHOIS Registrar

Meanwhile, previous research has shown that attackers tend to abuse specific registrars intensively. Thus, the registrar field of WHOIS record is also utilized as a feature.

## 3.14 Is Entry Locked

Lastly, whether the domain is locked or not is recorded as a feature. Locked domains are indicated by "Registrar lock" or "Client Transfer Prohibited" status in WHOIS entry. Domain names are normally locked to prevent from unauthorized changes or accidental transfers. A locked status indicates the domain is safer against misconduct than those that are not.

WHOIS features are presented in the following list.

domain_age: Time difference in seconds between the domain creation time and the time when the WHOIS entry is downloaded.

diff_update_time: Time difference in seconds between the update time and the time when the WHOIS entry is downloaded.

diff_expiry_time: Time difference between the domain expiry time and the time when the WHOIS entry is downloaded.

is_entry_locked: Whether or not the WHOIS entry forth is domain is locked.

who_is_registrar: The Registrar name specified in the WHOIS entry.

## 3.15 DNS Features

DNS plays a vital role in the Internet architecture. To deceive end users to visit phishing websites, a miscreant normally attempt to set up a page that looks like a legitimate website which then asks for sensitive personal information such as username and pass- word, as is presented. In this case, the attacker needs to configure assorted DNS entries, e.g., NS and A records, for the phishing website. Otherwise, the attacker need to use IP address as the domain name which has been discussed. If the corresponding DNS structure is set up for phishing websites, then the latent pattern difference between phishing websites and legitimate websites regarding the number of DNS records, type of records and TTL of records could be discovered through analysis of DNS content. Previous attempts of extracting features from DNS records have been discussed in prior works. We reuse some of the previously proposed features while adding new features at the same time. Some of the recursive name servers are known to strip off additional information of a DNS response. Thus, for each URL, we first perform a DNS NS query to get the domain's authoritative name server.

## 3.16 HTTP Redirection Features

While both legitimate domains and phishing domains utilize redirection in practice, different purposes and patterns lie behind it. For example, a legitimate domain, say, a.com, could redirect a user who is visiting http://a.com to https://a.com to ensure the security of the communication. On the contrary, a phishing website might ensconce itself behind an online redirection service to avoid being detected, as pointed out by Chhabra et al. Hence for each URL or domain, we first perform a redirection checking via Chromium browser running in headless mode driven by Selenium, a browser automation framework.

During the redirection, the program remembers all the intermediate URLs passed and the final landing URL, from which the length of the redirection chain and the number of different domains crossed are calculated as features. These two features could be recruited to reflect underlying redirection pattern differences between legitimate domains and phishing domains. For example, a long redirection chain with

multiple different domains crossed could be suspicious. We use both redirection features because one could imagine that a legitimate website might be more likely to use redirection within the same domain.
The list of features extracted from redirection is presented as follows.

len_redirection_chain: Length of the redirection chain from the initial URL/domain to the final destination.
different_domains_crossed: The number of different domains crossed during the redirection.

# 4 DESIGN

In this section, we first introduce the design of the data collection and classifier training system, going through each of the components. Next, we discuss the setup of the experiment, including the data collection process, legitimate and phishing website data source, machine learning algorithms used, and evaluation methods for the classification results.

## 4.1 Classification System Design

The design of our system for crawling data and generating classifier complies with the following criteria:

(1) The classifier should work with low false positives, since the misclassification of a legitimate website might lead to serious consequences such as lawsuits and financial loss. In a real commercial application of such systems, the administrator might even be willing to sacrifice the true positive rate for a low false positive rate.

(2) The classifier should be resilient to evasion techniques taken by miscreants.
This requires the feature set to possess a substantial portion of features which are challenging enough to be forged.

(3) Thefeature collection system should be flexible regarding changes in the feature set and data collection process. Considering the fast-changing nature of phishing websites, the feature set, which plays the most crucial role in classification, might need adjustments from time to time. Hence our system needs to be amenable to changes in an effortless style.

Based on the above objectives, the overall structure and components of the data collection and classifier training system are expressed . The detail of each component is discussed sequentially.

## 4.2 Classifier Training Workflow

Our data collection and classifier training system is constituted of the following parts: a set of disjoint feature collection and extraction pipelines, the training/testing data file generator, and the machine learning classifier.

## 4.3 Feature Crawling and Raw Data Processing Pipelines

The very first step of our workflow, givenwe already have the list of labeled websites, is to collect different sets of raw data corresponding to each group of features. As shown in Figure 4.3.1, a feature crawling and raw data processing pipeline comprises a Crawler, a Raw Data Database, a Raw Data Processor, and a Feature Database, yielding the set of feature values corresponding to the current group of features for later usage. Here each pipeline could correspond to one or more feature groups,

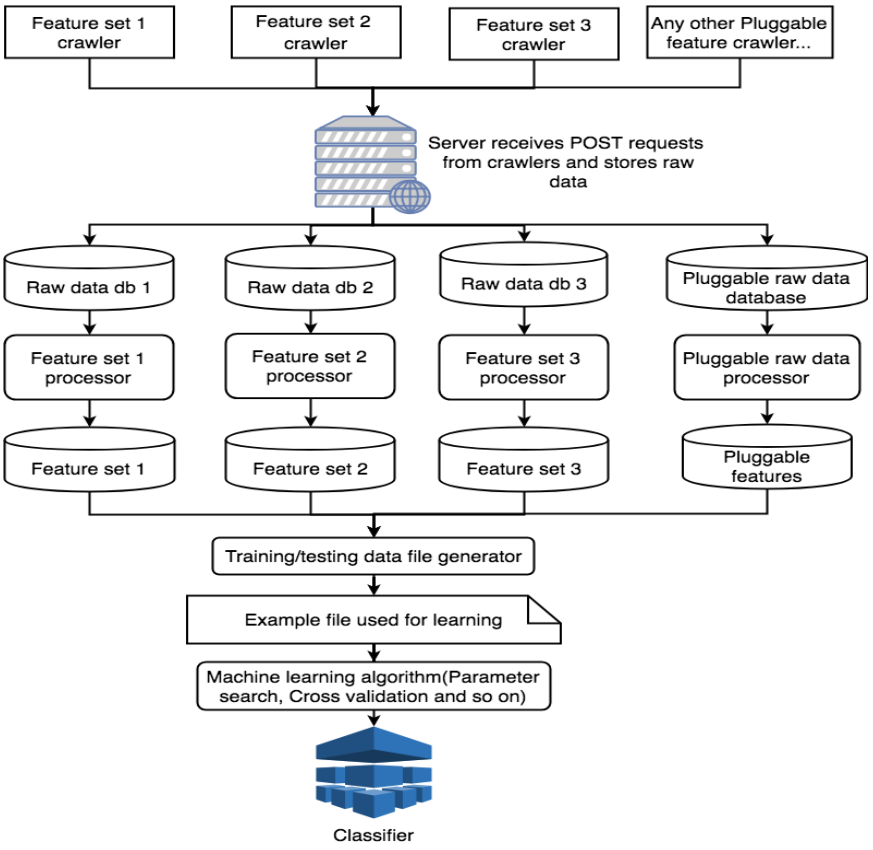


Figure 4.3.1. Classifier Training System with Decoupled Components

to achieve convenient and mutually-independent collecting and parsing processes. The main advantage of this design is low coupling between different pipelines, that is to say, low entanglement exists between the processes for collecting and parsing different groups of features. This design, therefore, enjoys the benefits of a plugin-and- play pattern for any new feature groups to be added in the future. Similarly, removal of old feature groups is as simple as unplugging the relevant pipeline from the system. In our case, we divide the features based on the collection position (i.e., distributed col- lection or not) and the raw data contents (i.e., certificates, network trace, and others) for ease of deployment, resulting in four sets: certificate-distributed, others-distributed, network-local and others-local.

Given a pipeline for a feature group, it starts with obtaining raw data used for extracting the features. For example, the pipeline of certificate features downloads the X.509 certificates for each website in our labeled website list, and the pipeline of server status features performs server scanning using tools like nmap and traceroute. The down- loaded raw data for each group is stored in separate databases for subsequent parsing.

As the next step, the corresponding process or of each group retrieves records from the database, transforming raw data into the set of feature values that belong to that feature group based on the meaning of features discussed . Extracted features are persisted in the corresponding database of each group of features.

## 4.4 Training/Testing Data File Generator

The training/testing data file generator assembles features from each of the separate feature database and produces the *csv* file used as machine learning input in which each of the examples is represented as a feature vector.

## 4.5 Machine Learning Classifier

After the feature vectors of URLs/websites are stored in a *csv* file, this file is fed to the machine learning algorithms to learn the classifier used for predicting the label of new URLs/websites, i.e., labeling them as positive or negative. Random Forests and Decision Tree algorithms are applied to the feature vectors in this case. Decision Tree is selected because of its understandability which conduces to the comprehension of the classification process and phishing characteristics, while Random Forests is selected because:

(1) Random Forests algorithm is robust against noise and outliers in the training data.

(2) Random Forests algorithm can be easily parallelized.

(3) Random Forests algorithm gives a useful internal estimate of error and feature importance, whichcan be used for analysis later on.

## 4.6 Practical Usage

Our proposed system is naturally suitable for being applied by large-scale service providers such as Google or Bing. Considering they already have well-established distributed infrastructures for Internet-wide data crawling, collecting features should be relatively easy integration.

Once the classifier is trained using the above-described system, users can query the system for the classification result of a URL to determine its legitimacy.

The query, however, can result in two scenarios. One is that the system has already performed a classification on this queried URL before and the results are preserved in the constructed blacklist. In this case, the result could be returned to the user directly. Another is that the queried URL has never been seen by the system previously. In this case, the system needs to start a new distributed feature collection process for the given URL, producing its feature vector. This feature vector is fed into the machine learning classifier to get a prediction result, which is conserved in the blacklist database for future reference.

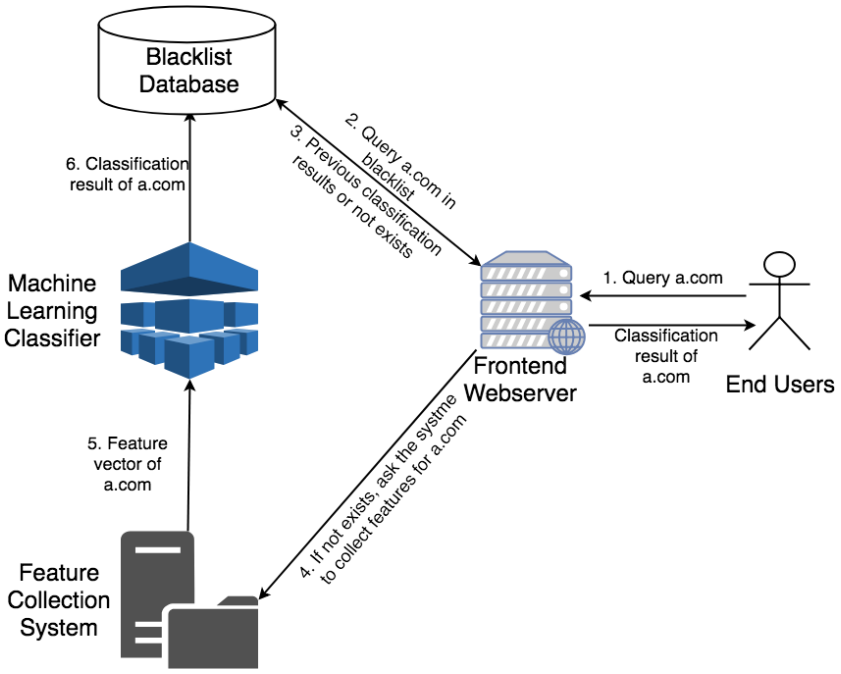


Figure 4.6.1. User Query Handling

In the second case,the first user who submitted the initial query for the URL might not be able to get the desired classification result for the URL. However, any user querying for this URL subsequently will benefit from the query of the first user. Thus, we argue that this system, once integrated with existing service providers, is advantageous for a majority of the users, protecting them from phishing attacks.

## 4.7Data Collection

### 4.7.1Data Sets

Our phishing list is obtained from Phish Tank, an opensource blacklist which contains phishing URLs that have been manually verified by community members. Our script downloaded the blacklist hourly fromApril17th, 2018untilAugust 15th, 2018.
The benign list comes from a snapshot of the Majestic top one million list on July 16th as this could be a good representative of legitimate websites, considering their relatively higher popularity among all websites. Note that using top ranked websites as benign examples is a common practice in relevant studies.
For all the URLs, we follow their redirection chain to get the landing URL as the actual target for crawling.

# 5 IMPLEMENTATION

## 5.1 Machine Learning Algorithms

In this paper, we utilize Decision Tree and Random Forests for classification.

### Data PreProcessing

**Data is in two data frames so we merge them to make one dataframe**

Note: two dataframes has same column names

```
In [9]: urls = legitimate_urls.append(phishing_urls)
```

```
In [10]: urls.head(5)
```

Out[10]:

| | Domain | Having_@_symbol | Having_IP | Path | Prefix_suffix_separation | Protocol | Redirection_//_symbol | Sub_domains | URL |
|---|---|---|---|---|---|---|---|---|---|
| 0 | www.liquidgeneration.com | 0 | 0 | / | 0 | http | 0 | 0 | |
| 1 | www.onlineanime.org | 0 | 0 | / | 0 | http | 0 | 0 | |
| 2 | www.ceres.dti.ne.jp | 0 | 0 | /~nekoi/senno/senfirst.html | 0 | http | 0 | 1 | |
| 3 | www.galeon.com | 0 | 0 | /kmh/ | 0 | http | 0 | 0 | |
| 4 | www.fanworkrecs.com | 0 | 0 | / | 0 | http | 0 | 0 | |

```
In [11]: urls.columns
```

```
Out[11]: Index(['Domain', 'Having_@_symbol', 'Having_IP', 'Path',
```

Figure 5.1.1 Data Preprocessing

**Removing Unnecessary columns**

```
In [12]: urls = urls.drop(urls.columns[[0,3,5]],axis=1)
```

**Since we merged two dataframes top 1000 rows will have legitimate urls and bottom 1000 rows will have phishing urls. So if we split the data now and create a model for it will overfit so we need to shuffle the rows before splitting the data into training set and test set**

```
In [13]: # shuffling the rows in the dataset so that when splitting the train and test set are equally distributed
urls = urls.sample(frac=1).reset_index(drop=True)
```

**Removing class variable from the dataset**

```
In [14]: urls_without_labels = urls.drop('label',axis=1)
urls_without_labels.columns
labels = urls['label']
```

**splitting the data into train data and test data**

```
In [15]: from sklearn.model_selection import train_test_split
data_train, data_test, labels_train, labels_test = train_test_split(urls_without_labels, labels, test_size=0.20, random_state=10
```

Figure 5.1.2 Data Preprocessing

```
In [16]: print(len(data_train),len(data_test),len(labels_train),len(labels_test))

         1612 403 1612 403

In [17]: labels_train.value_counts()

         #labels_train[labels_train == 0].count()
         #labels_train[labels_train == 1].count()

Out[17]: 0    814
         1    798
         Name: label, dtype: int64

In [18]: labels_test.value_counts()

Out[18]: 0    203
         1    200
         Name: label, dtype: int64
```

Figure 5.1.3 Data splitting testing and training

## 5.2 Decision Tree

Scikit-learn implements an optimized version of the CART decision tree algorithm. CART algorithm is similar to the well-known C4.5 algorithm. Based on CART decision tree algorithm, each node is constructed by selecting the feature or threshold that generates the most substantial improvement using "Gini measure of impurity"(see Equation below). By default, it constructs binary trees only, that is, every parent node is split into two child nodes. The authors of CART argue that binary splits are preferable com- pared with multiway splits for the following reasons:

(1) The data is fragmented slowly.

(2) Repeated splits on the same attribute are possible. In this case, any attribute can be partitioned for as many times as possible.

The trees generated by CART are grown to a maximal size without using any stopping rules until no further splits are possible. Afterward, the trees are pruned such that the splits contributing the least to the overall performance are removed.

The "Gini measure of impurity" used by CART, given a tree node t, can be calculated using equation , of which $j$ is the different class labels in the problem and $p(j)$ is the probability of the class $j$.

$$G(t)=1-\sum_{j} p(j)^2$$

Once a parent node $P$ is split into two child node $L$ and $R$, the improvement, or gain, generated by the split is calculated using equation 4.2.

$$I(P)=G(P)-q\cdot G(L)-(1-q)\cdot G(R)$$

Here, $q$ is the portion of instances that goes to the left subtree.
The reason why Gini is favored over entropy, by the authors of CART, is that:

(1) Gini can be computed more quickly.

(2) Using Gini can make the tree to generate"pure nodes",i.e.,nodes with only one class label that do not require further split, more likely.

```
from sklearn.tree import DecisionTreeClassifier
```

```
model = DecisionTreeClassifier()
```

```
model.fit(data_train,labels_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

**predicting the result for test data**

```
pred_label = model.predict(data_test)
```

Figure 5.2.1 Decision Tree Classifier

## 5.3 Random Forests

Different from Decision Tree, Random Forests[55] grows an ensemble of asymmetric trees with the purpose of adding randomness to the constructed Decision Trees. This approach improves classification performance considerably in most cases.

A random group of input variables, or features, and training data are selected to grow a series of decision trees. As a result, the best split among the subset of features randomly chosen from all features , rather than the best split among all features, is selected for splitting the node in each tree. The predicted class of a given input is the most popular vote among all trees in the forest. In contrast to the original publication, Random Forests implemented by scikit-learn combines individual classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class

```
from sklearn.ensemble import RandomForestClassifier

random_forest_classifier = RandomForestClassifier()
```

```
random_forest_classifier.fit(data_train,labels_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

**Predicting the result for test data**

```
prediction_label = random_forest_classifier.predict(data_test)
```

Figure 5.3.1 Random Forest Classification

## 5.4 GUI Screen Approach

**Website code:**

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknL
PMO" crossorigin="anonymous">
    <title>Phishing Url Detection</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <style>
@keyframes dot-keyframes {
  0% {
    opacity: 0.4;
    transform: scale(1, 1);
  }
  50% {
    opacity: 1;
    transform: scale(1.2, 1.2);
  }
  100% {
    opacity: 0.4;
    transform: scale(1, 1);
  }
}
.loading-dots {
  text-align: center;
  width: 100%;
}
.loading-dots--dot {
  animation: dot-keyframes 1.5s infinite ease-in-out;
  background-color: #fff;
  border-radius: 10px;
  display: inline-block;
  height: 10px;
  width: 10px;
  margin: 15px;
}
.loading-dots--dot:nth-child(2) {
  animation-delay: 0.5s;
}
.loading-dots--dot:nth-child(3) {
  animation-delay: 1s;
```

```
}

    </style>
  </head>
  <body style="background-image:
url('https://www.it.ucla.edu/sites/g/files/yaccgq956/f/february-security-article-image-phishing-blank992x500.jpg');">

    <center style="margin-top: 90px;">
     <h1 class="text-white">Phishing Url Detection</h1>
      <div class="single-url">
      <div class="input-group input-group-lg" style="width: 800px;margin-top: 50px">
        <div class="input-group-prepend">
          <span class="input-group-text" id="inputGroup-sizing-lg">Enter a
URL:</span>
        </div>
        <input type="text" class="form-control" id="url" value="">
          <button type="submit" class="btn getResult btn-primary btn-lg mb-2"
style="margin-left: 10px">Submit</button>
            <div>
      <span class="text-white">If you want see the result of multiple URL's. You can
upload a file containing URL's </span><span><button type="button" class="btn text-
white btn-primary multi">Upload File</button></span>
    </div>

      </div>
    </div>

    <div class="multiple-urls" style="display: none;margin-top: 50px">
  <form method='POST' action="http://localhost:5000" enctype=multipart/form-data>
      <input type=file name=file class="btn text-white">
      <input type=submit value=Upload class="btn btn-primary">
    </form>
          <div>
      <span class="text-white">If you want see the result of single URL. You can enter
a single URL </span><span><button type="button" class="btn text-white btn-primary
single">Enter URL</button></span>
    </div>
  </div>
      <div class="result"></div>

    </center>



  </body>
  <script>
```

```
document.getElementsByClassName("getResult")[0].addEventListener("click",result)
;
  function result(){

    var div = document.querySelector(".result");
     var urlname = document.getElementById('url').value;
                $.ajax({
                        data : {
                                name : urlname
                        },
                        type : 'GET',
                        url : '/result'
                }).always(div.innerHTML= '<div class="loading-dots mt-4"><div
class="loading-dots--dot"></div><div class="loading-dots--dot"></div><div
class="loading-dots--dot"></div></div>')
                .done(function(data) {

                        if (data.error) {
                                alert(data.error);
                        }
                        else {

      var newHTML = '<h3 class="text-white mt-4">This is a '+data+'</h3>';
      div.innerHTML= newHTML;
     }

                });


  }

$(document).ready(function(){

$(".multi").on('click',function(){

$(".multiple-urls").show();
$(".single-url").hide();
});

$(".single").on('click',function(){

$(".multiple-urls").hide();
$(".single-url").show();
});

});

 </script>
</html>
```

**Validation Code for checking URL Legamiate or Phishing:**
**Server.py**

```
import os
import phishing_detection
from flask import Flask
from flask import (
   Blueprint, flash, g, redirect, render_template, request, session, url_for
)
from flask import jsonify
from werkzeug.utils import secure_filename
app = Flask(_name_)

UPLOAD_FOLDER= '/files'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif','py'])
def allowed_file(filename):
   return '.' in filename and \
        filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


@app.route('/result')
def result():
   urlname = request.args['name']
   result = phishing_detection.getResult(urlname)
   return result

# @app.route('/upload')
# def upload():
#         return 'yes'

@app.route('/', methods = ['GET', 'POST'])
def hello():
        if request.method == 'POST':
                if 'file' not in request.files:
                        flash('no file part')
                        return "false"
                file = request.files['file']
                if file.filename == '':
                        flash('no select file')
                        return 'false'
                if file and allowed_file(file.filename):
                        filename = secure_filename(file.filename)
                        contents = file.read()
                        with open("files/URL.txt","wb") as f:
                                f.write(contents)
                        file.save = (os.path.join(app.config['UPLOAD_FOLDER'],
filename))
                        return render_template("getInput.html")
```

```
        return  render_template("getInput.html")


if__name__== '_main_':
    app.run(debug=True)
```

# 6 TESTING AND VALIDATION

In **machine learning**, model **validation** is referred to as the process where a trained model is evaluated with a testing data set. The testing data set is a separate portion of the same data set from which the training set is derived. Model **validation** is carried out after model training.

## 6.1 Validation Dataset:

The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.

## 6.2 Test Dataset:

The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

## 6.3 Understanding Model Validation for Classification:

- Define the problem : Predict whether it will rain tomorrow or not.
- Data Set description : Rainfall data contains 118 features and one dependent variable (y_test) whether it will rain or not.
- Metric is a technique to evaluate the performance of the **model**.

# 7 RESULT ANALYSIS

## 7.1 creating confusion matrix and checking the accuracy for decision tree algorithm

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(labels_test,pred_label)
print(cm)
accuracy_score(labels_test,pred_label)
```

```
[[203   9]
 [  1  93]]

0.9673202614379085
```

Figure 7.1.1 Decision Tree Confusion matrix and Accuracy

## 7.2 Creating confusion matrix and checking the accuracy for Random Forest Classification

```
from sklearn.metrics import confusion_matrix,accuracy_score
cpnfusionMatrix = confusion_matrix(labels_test,prediction_label)
print(cpnfusionMatrix)
accuracy_score(labels_test,prediction_label)
```

```
[[287   6]
 [  2 164]]

0.9825708061002179
```

Figure 7.2.2 Random Forest Confusion matrix and Accuracy

## 7.3 Checking by Decision tree algorithm using GUI Screen

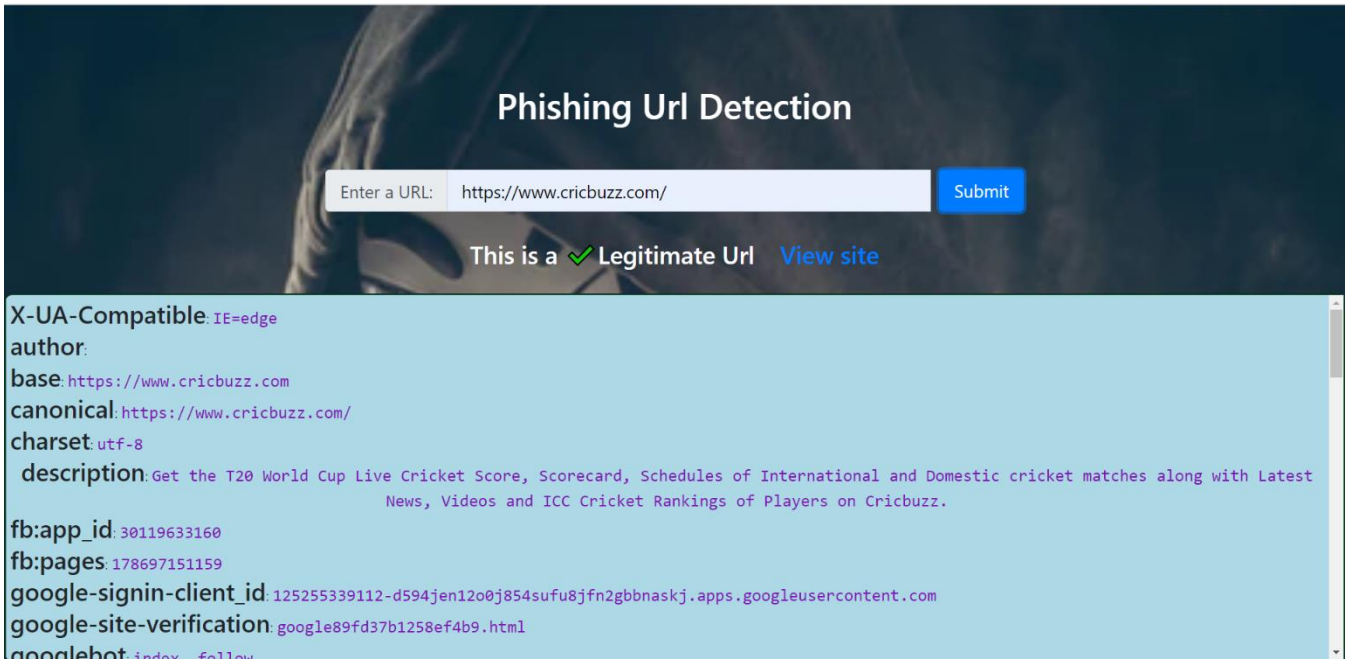


Figure 7.3.1 URL is Legitimate

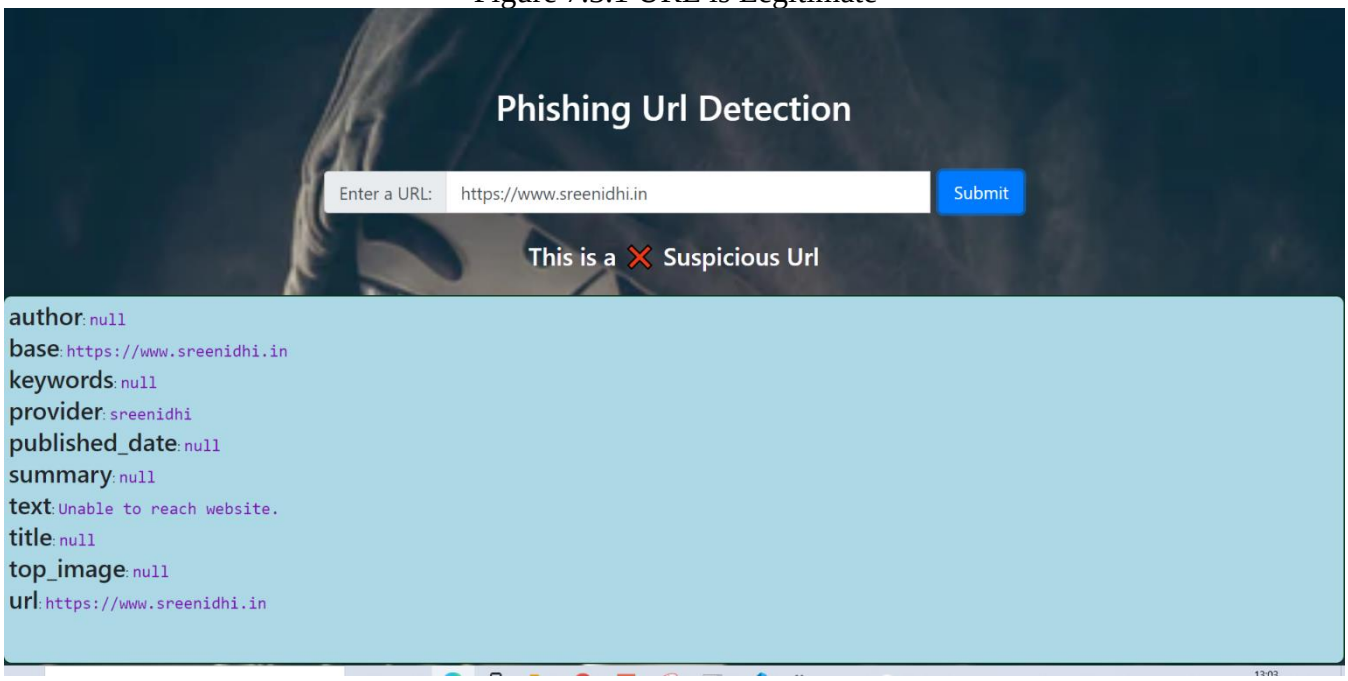


Figure 7.3.2 URL is Phishing

# 8 CONCLUSION

In this thesis, an automatic phishing detection system is presented. The system utilizes a rich set of features, including hard-to-forge features, captured from many different aspects of the corresponding URLs, as well as from distributed vantage points. With using the Majestic top one million list as legitimate URLs and blacklists downloaded hourly from Phish Tank as phishing URLs, data collection and machine learning experiments are conducted to evaluate the proposed methodology. In addition, a previous study is analyzed based on our contemporary data set.

Experiment results show that our approach achieves good accuracy for phishing detection, indicating the effectiveness of the proposed mechanism. The previous study that is analyzed, on the other hand, exposes a performance decline due to the evolution of the phishing ecosystem, while our proposed methodology and feature set demonstrates significant superiority. Meanwhile, differences between legitimate and phishing websites are revealed based on the case study of some features.

Nevertheless, there are some limitations in our study as well. First of all, the legitimate and phishing examples used as ground truth in the experiments are downloaded from single sources, i.e., Majestic top one million list and phishtank.com. Further evaluation of the mechanism against different sources of URLs is needed. Besides, as is described, our data collection process fails to gather information for some of the URLs we have, due to the reasons mentioned. Based on our best knowledge, no bias is found within the dataset because of data loss. However, further evaluation of the mechanism with more complete data set is preferable.

Besides, there are more than 140 features used by the machine learning process currently. Further data analysis focused on understanding the classification process and reducing the total number of features used would be preferable. For example, answering the question, what is the smallest possible set of features that could lead to the same level of performance, would be interesting. This could significantly reduce the effort needed

for data collection. Nonetheless, decreasing the total number of features has the side effect of facilitating attackers by reducing the effort needed to deceive the system. Itisworthwhile to point out that the battle between attackers and guardians of the cyber- security is endless. Even though our proposed features present high classification power against state-of-the-art phishing practice, they could be stale and ineffective someday as a result of the evolution of phishing ecosystem. Hence constant analysis and update of the features is required to maintain the high detection power of the system.

# REFERENCES


[1] https://tools.ietf.org/html/rfc5280.

[2] https://tools.ietf.org/html/rfc5246.

[3] https://tools.ietf.org/html/rfc2818.

[4] ZakirDurumeric,JamesKasten,Michael Bailey,andJ.AlexHalderman.Analysisof the https certificate ecosystem. In Proceedings of the 2013 Conference on Internet Measurement Conference, IMC'13, pages 291–304, NewYork,NY,USA,2013. ACM.

[5] 2013 microsoftcomputingsafetyindex(mcsi) worldwideresultssummary.

[6] https://apwg.org/.

[7] http://docs.apwg.org/reports/apwg_trends_report_q4_2017.pdf.

[8] Aquarter of phishingattacksare nowhosted onhttps domains: Why?

[9] https://safebrowsing.google.com/.

[10] https://en.wikipedia.org/wiki/microsoft_smartscreen.

[11] Luca Invernizzi, Kurt Thomas, Alexandros Kapravelos, Oxana Comanescu, Jean- Michel Picod, and Elie Bursztein. Cloak of visibility: detecting when machines browse a different web. In Security and Privacy (SP), 2016 IEEE Symposium on, pages 743–758. IEEE, 2016.

[12] Sidharth Chhabra, Anupama Aggarwal, Fabrício Benevenuto, and Ponnurangam Kumaraguru. Phi.sh/$ocial: the phishing landscape through short urls. In CEAS, 2011.

[13] https://www.phishtank.com/index.php.

[14] https://www.opendns.com/.

[15] https://majestic.com/reports/majestic-million.